International Journal of Latest Research in Engineering and Technology (IJLRET)
ISSN: 2454-5031
www.ijlret.com || Volume 05 - Issue 12 || December 2019 || PP. 01-15

Hybrid Algorithm Improving Bisection, Regula Falsi, Dekker, Brent Algorithms for Roots of Non-linear Equations

Chaman Lal Sabharwal
Computer Science Department, Missouri University of Science and Technology, Rolla, MO 65409, USA;

Abstract: Bisection Method for continuous functions and Newton-Raphson method for differentiable function are widely used for finding zeros of non-linear equations. Numerical techniques are explored when an analytic solution is not obvious. There is no single algorithm that works best for every function. We designed and implemented a new hybrid algorithm that is a dynamic blend of theBisectionand RegulaFalsialgorithms. The experimental results validate that the new algorithm outperforms Bisection,RegulaFalsi, Dekker's and Brent'salgorithms with respect to computational iterations. The new algorithm is guaranteed to find a root and requires fewer computational iterations. It is also observed that the new hybrid algorithm performs better that the Secant algorithm and the Newton–Raphson algorithm.The new algorithm guarantees the reliability of Bisection method and speed of Secant method. The theoretical and empirical evidence shows that the computational complexity of the new algorithm is considerably less than that of the classical algorithms.
Keywords-Algorithm , Bisection,Brent, Dekker, Hybrid, Newton–Raphson, RegulaFalsi,Secant

1. Introduction

Finding zeros of non-linear equations is a fundamental problem in optimization methods. The root finding algorithms can be classified into two categories: (1) continuous functions defined on an interval [a, b] such that f(x) is of opposite sign at the end points of [a, b], meaning f(a)f(b) < 0; (2) differentiable functions with a start point close to the zero of the function. Themethods of first category include Bisection, False Position (1697), Dekker (1969), Brent (1973); while methods of second category include Newton-Raphson, Secant, Reverse Quadratic Interpolation (RQI), Halley. Non-linear equationsarise in most disciplines including computer science, natural sciences (civil engineering, electrical engineering, mechanical engineering), biological engineering and social sciences (psychology, economics), etc.Problems such as minimization, target shooting, orbital plenary motion, etc., often lead to finding roots of non-linear functional equations [1],[2], [3].

The classical root-finding algorithmssuch as Bisection, False Position, Newton–Raphson,and Secant methodare ubiquitous. There are improvements of these algorithms for speedup, more elegant and efficient implementations are emerging. The variations of Bisection and False Position method are due to Dekker [4], [5], and Brent[6], [7] and alternate versions of Newton–Raphson and Secant methods. From these algorithms, the developer has to explore and exploit the algorithm suitable under specified constraints on the function and the domain.There is not a single algorithm that works best for every function[8], [9]. We present a new hybrid algorithm that outperforms all the existing algorithms. See Section 4 for empiricaloutcomes validating the performance of the new algorithm

Sometimes, in applications, we may need to determine when will two functions g(x) and h(x) be equal, i.e., g(x) =h(x). This translates into finding roots of the classical equation f(x) =g(x) − h(x) =0.Similarly, the optimal value of g(x) can be determined by finding a root of the derivative equation $\frac{dg(x)}{dx}$ =0, which implies finding roots of the equation $\frac{dg(x)}{dx}$ =f(x) =0.

Even though classical methods have been developed and used for decades, enhancements are progressively made to improve the performance of these methods. A method may outperform other methods on one dataset, and may produce inferior results on another dataset. Different methods have their own strengths/weaknesses. A dynamic new hybridalgorithm (blend of Bisectionand False Position) is presented here by taking advantage of the best in Bisectionand False Position methods to locate the roots independent of Dekker and Brent approach.Its computational efficiency is validatedby comparing itwith the existing methods via complexity analysis and empirical evidence.The new hybrid algorithm isa tested technique that outperforms other existing methods by curtailing the *number of iterations* required for approximations (Table 1). Table 1is also a listing of functions from literature used to test the validity of results comparing with the new algorithm.

Carpentry Example: Given overall dimensions w and h, and the material dimension, b, what is the value of θ to achieve this picnic table configuration, see Fig.1[10]. The problem becomes that of determining the angle θ such that

$$w \sin\theta = h \cos\theta + b \quad (1)$$

An analytical solution for θ exists, but is not obvious; in fact,θ =atan2(w, -h) ± atan2($\sqrt{h^2 + w^2 - b^2}$, b). Such examples, where analytic solution exists, are used to confirm that the numerical solution is an accurate approximation. We used this example as one of test cases for numerical approximate solutions. This leads to finding a numerical value of θ that satisfies the equation

$$f(\theta) = h \cos\theta + b - w \sin\theta = 0 \quad (2)$$

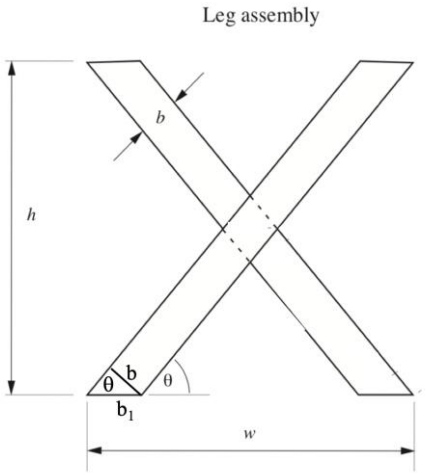


Figure 1. Given the configuration in terms of h,w,b, to determine the angle θ.[10]

This paper is organized as follows.Section 2 is a brief description of the classical methods Bisection, RegulaFalsi,Dekker, Brent, Newton–Raphson, Secant; their strengths and pitfalls.Section 3describes the new hybrid algorithm. Section 4 gives a theoretical justification for the hybridalgorithm. Section 5 presents empirical results validating the performance of the new algorithm. Section 6 is the conclusion.

2. Background

The classical methods for finding roots of non-linear equations includeBisection, False Position, Newton–Raphson and Secant methods. For completeness,we brieflydescribethese methods including (1) root approximation, (2) error calculation and (3) termination criteria. Since there is no single optimal algorithm for root approximation[8], [9], we will design a new hybrid algorithm that outperforms these methods.

We constrict this discussion to finding a single root instead of all the roots of an equation. In case an equation has several roots, we can delineate an interval where the desired root is to be found. No matter which method is used to find zeros ofan equation f(x)=0, it amounts to constructing a sequence $\{r_k\}$of approximations that gradually converges to accurate solution.

A sequence $\{ r_n \}$ converges to r if there exists a positive constant c, a positive integer N, and a positive real number p, such that $\lim_{n\to\infty} r_n = r$ and $|r_{n+1} - r| < c\,|r_n - r|^p$ for n>N.The sequence of approximations converges *linearly*, *super linearly*, *quadratically* and *cubically* depending on the value of p: p=1, 1<p<2, p=2, and p=3 respectively.Convergence checking (1 considers if the two approximations r_n and r_{n-1} are close to each other, (2) tests if $f(r_n)$ is close to zero, and (3) avoids the unnecessary search.

2.1. Bisection Method(1817)

The Bisectionmethod (1) is based on binary slashing of irrelevantsubintervals, (2)is a virtual binary search,and (3) is guaranteed to converge to the root, see Algorithm 1. It does not matter what the function is, the root-error upper bound is fixed at each iteration and can be determined a priori. By specifying the root-error tolerance, the upper bound on thenumber of iterations can be predetermined quickly.

Problem If a function f:[a, b]→R, is 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)·f(b) < 0, then there exists a root r ∈ [a, b] such that f(r) =0.

Algorithm 1: Bisection Method

1. Input: f and [a,b], $\in$, maxIterations
2. Output: root r, k-number of actual iterations, bracketing interval $[a_k,b_k]$
2. //initialize
3. k =1; $[a_k,b_k]$ = [a,b];
4. repeat
5. //compute the mid-point
6. $r=a_k + \frac{b_k - a_k}{2}$
7. if $f(a_k)\cdot f(r) >0$,
8. $a_{k+1} = r; b_{k+1} = b_k$;
9. else
10. $a_{k+1} = a_k; b_{k+1} = r$;
11. endif
12. $r_k = r$;
13.k =k+1;
14. until $|f(r)| < \in$ or k> maxIterations

2.1.1. Merits and Drawbacks of Bisection Method

Since the Bisectionmethod brackets the root, at each iteration, the length of the root-bracketing interval is halved. Thus, the method guarantees the decline in error of approximate root at each iteration, because the interval enclosing the root shrinks in half. The convergence of the Bisectionmethod is certain because it is simply based on halving the length of the interval containing the root, at each iteration.

Though the convergence of the Bisection method is guaranteed, its rate of convergence is slow, and as such it is quite difficult to extend its usability for systems of equations. If for any iteration, root approximation is very close to an endpoint of bracketing interval, Bisection does not take advantage of this information [5], it will take same pre-determined number of iterations to reach the root [11].

2.2. False Position (Regula Falsi) Method

The motivation for innovative methods arises from the poor performance of the Bisectionmethod. One such method is finding zeros by *linear interpolation*. The False Position is a dynamic method; it takes advantage of the location of the root to make a conceivably better appropriate selection. Unfortunately, this method does not perform as well as expected, for all functions, see Fig. 2.

Here also, the function f:[a, b]→R is such that (1) f(x) is continuous, where R is the set of all real numbersand (2) f(a) and f(b) are of opposite signs, i.e., $f(a)\cdot f(b)<0$. The algorithm uses a, b as the two initial estimates of the root. The False Position method uses both endpoints as two starting values $r_o = a_1 = a$, $r_1 = b_1 = b$.The secant line through points $(r_0, f(r_0))$ and $(r_1, f(r_1))$intersects the x-axisto yield thenext estimate, r_2. For successive estimates, the False Position method intersects the secant line through $(a_n, f(a_n))$ and $(b_n, f(b_n))$with the x-axis for the approximation,r_{n+1},

$$r_{n+1} = a_n - \frac{f(a_n)(a_n - b_n)}{f(a_n) - f(b_n)} \qquad (3)$$

for $n \geq 1$.

The approximations,r_n, are ensured to be bracketed as in the case of Bisectionmethod. And one should note that the length of the brackets do not necessarily tend to zero. The pseudo-code of the Algorithm 2 is as follows.

Algorithm 2: False Position Method

1. Input: f and [a, b], $\in$, maxIterations
2. Output: root r, k-number of actual iterations, bracketing interval $[a_k, b_k]$
3. initialize
4. $r_0 = a$; $r_1 = b$
5. k =1; $[a_k,b_k]$=[a,b]
6. repeat
7. compute next point the secant line intersection with x-axis, note that $f(b_k) - f(a_k) \neq 0$
8. $r=a_k - \frac{f(a_k)(b_k - a_k)}{f(b_k) - f(a_k)}$
9. if $f(a_k)\cdot f(r) >0$,

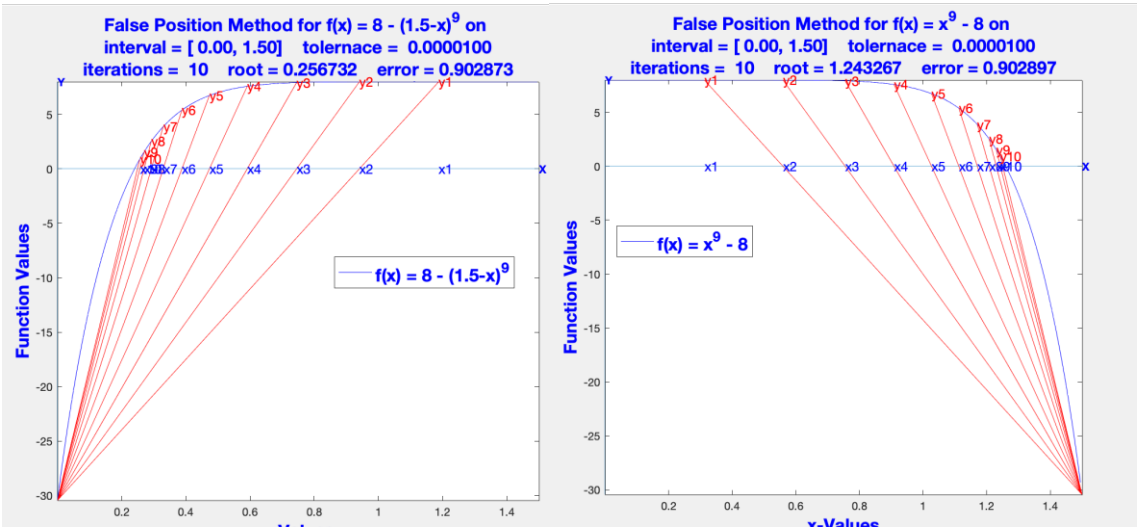
10. a_{k+1} =r;b_{k+1} = b_k;
11else
12. a_{k+1} = a_k;b_{k+1} =r;
13. endif
14. r_k = r;
15. k =k+1;
16. until |f(r)| <∈ or k>maxIterations

2.2.1. Merits and Drawbacks ofFalse Position Method

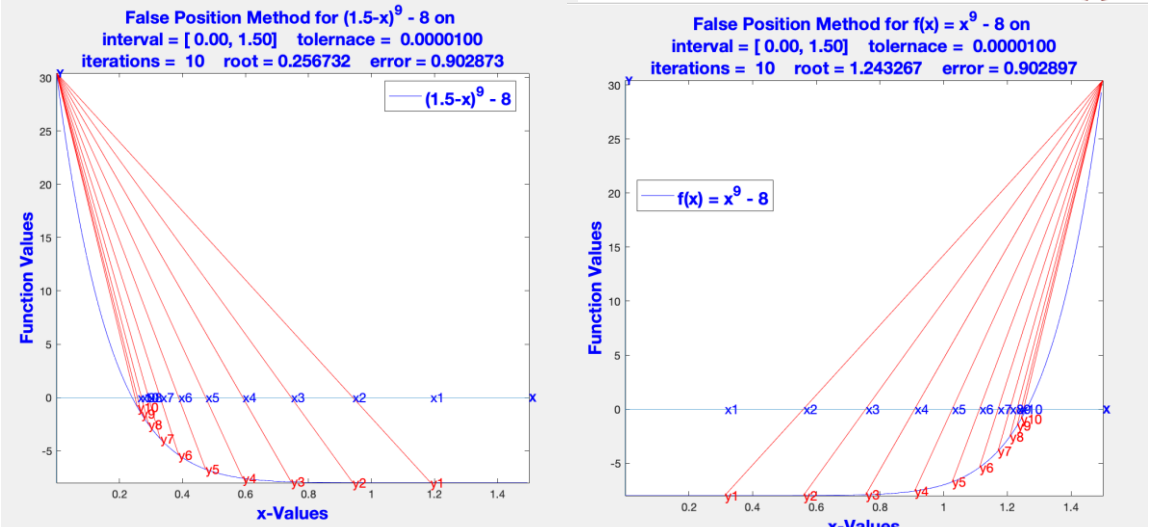
The root approximation error calculation in the Bisection method is straight forward (see Section 4); in each iteration root approximation error is halved, i.e., root approximation error in the *n*-th step is no more than $\frac{b-a}{2^n}$. This is not the case for the False Position method.It is guaranteed to converge due to the gradually decreasing of distance between the approximation and the rootin the root-bracketing interval, though the length of the interval may not become zero. It is fast when you know the linear nature of the function.

Unlike Bisection method, for the False Position method, there is no way to determine a priori the number of iterations needed for convergence. If we cannot ensure that the function can be interpolated by a linear function, then applying the False Position method can resultinworseresultsthantheBisectionmethod.A problem occurs when the function is convex, concave up or concave down.According to [12], for a concave down function, the left endpoint remains stationary and the right endpoint updates in each iteration. For concave up function, the right endpoint remains stationary and the left endpoint updates in each iteration. This is *not an accurate* statement, it works for some of the functions, not all the functions. Fig. 2 examples*contradict* their statement.

In the examples in Fig. 2 , four functions are used with the same tolerance $0.1^{-6}$and 10 as the upper limit of the number of iterations for visualization purposes. The purpose is to show how the False Position algorithmbehaves in these four cases. For visualization, in the interest of simplicity of plots, the algorithms are terminated before reaching the error-tolerance.



(a) (b)



(c) (d)

Figure 2. (a) Convex function concaves 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 (1969) algorithm was designed to circumvent the shortcomings of slow speed of Bisectionand uncontrolled iteration count of False Position method.It is a combination of two methods: Bisection, False Position [5].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 "best" estimate of the approximate root. The algorithm maintains that $\{b_k\}$ is the sequence of best estimates, the root enclosing interval is always $[a_k, b_k]$ or $[b_k, a_k]$.Dekker's algorithm maitain three values: b is the current best approximation, c is the previous previous approximation b_{k-1} and a is the contrapoint so that root lines in the interval $[a_k, b_k] \cup [b_k, a_k]$. Both secant point, s, and midpoint, m, are computed, b_k is s or m whichever is closest to b_{k-1}. This method is described in Algorithm 3.

Algorithm 3: Dekker's Method

1. Input: f and [a, b], $\in$, maxIterations
2. Output: root r, k-number of actual iterations, bracketing interval $[a_k, b_k]$
3. initialize
4. $b_0 = a$; $b_1 = b$
5. $[a_1, b_1] = [a, b]$, $c = b_1$;
6. If $(f(a_1)<f(b_1))$, swap (a_1,b_1) endif
7. k=1;
8. repeat
9. compute the Secant line at the point-point
10. if $f((b_k) \neq f(b_{k-1})$
$$s = b_k - \frac{f(b_k)(b_k - b_{k-1})}{f((b_k) - f(b_{k-1})}$$
11. else
$$s = b_k - \frac{f(b_k)(b_k - a_k)}{f((b_k) - f(a_k)}$$
12. endif
13. Secant method may take longer than Bisection method.
14. $m = (a_k+b_k)/2$;
15. b_{k+1} is s or m whichever is closer to $b_k$to shrink the root enclosing interval
16. if $(b_k < s < m$, $b_{k+1} = s$; else $b_{k+1} = m$;
17. if $f(a_k)f(b_{k+1})<0$ $a_{k+1}= a_{k+1}$else $a_{k+1} = c = b_k$
18. Ensure that $f(b_{k+1})$ is smaller value

19. If ($|f(a_{k+1})|<|f(b_{k+1})|$), swap ($a_{k+1}$,$b_{k+1}$); endif
20. r = b_{k+1}; c= b_{k+1};
21. k = k+1;
22. until $|f(r)| <\in$ or k >maxIterations

2.3.1 Merits and Drawbacks of Dekker's Method

Dekker's method performs better if the function f is reasonably well-behaved. However, this mehod is not devoid of shortcomings. There are pathological cases in which every iteration employs the secant method, and the iterates b_k converge very slowly. For example, (1)this can run into problem when $| b_k - s| < | b_k - m|$, but $|f(s)|>|f(m)|$, (2) $|b_k - b_{k-1}|$ may be arbitrarily small, Fig. 2. In such cases, Dekker's method requires far more iterations than the Bisection method, thus it is no better than even Bisection method. Another problem is that this criteria creates interval [a,b] where f(a)•f(b)>0, the algorithm terminates with out completion.

2.4 Brents Algorithm

Brent's(1973) algorithm evolved from the failures of Dekker's algorithm, however it is a step towards 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) . Reverse quadratic interpolation is based on the method of Lagrange polynomials and it leads to extra calcualtions.Thus, it may take more iterations to circumvent pathological cases. It uses three estimates to derive the next estimate. It is implemented in Matlab as fZero algorithm. It has the same conditions (1)continuity on f(x), and (2) f(a)f(b)<0 at the end points.

2.4.1 Detour to Reverse Quadratic Interpolation

Let a, b, 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)). 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 directformula for reverse quadratic polynomial that can be evaluated from values of y.

$$x = \frac{(y-f(b))(y-f(c))}{(f(a)-f(b))(f(a)-f(c))}a + \frac{(y-f(a))(y-f(c))}{(f(b)-f(a))(f(b)-f(c))}b + \frac{(y-f(a))(y-f(b))}{(f(c)-f(a))(f(c)-f(b))}c \qquad (4)$$

For example, for each iteration we need to compute new root x_{k+1} from three values.For simplicty, let the three valuesx_{k-1}, a_k, $x_k$be c, a, b (b being the best estimate, c the previous estimate value, a for the end point of root enclosing interval [a,b]∪[b,a])and to determine $b_{k+1} = x_{k+1}$we set $y(x_{k+1}) \cong 0$. This leads to

$$x_{k+1} = \frac{(0-f(b))(0-f(c))}{(f(a)-f(b))(f(a)-f(c))}a + \frac{(0-f(a))(0-f(c))}{(f(b)-f(a))(f(b)-f(c))}b + \frac{(0-f(a))(0-f(b))}{(f(c)-f(a))(f(c)-f(b))}c \qquad (5)$$

$$x_{k+1} = \frac{af(b)f(c)}{(f(a)-f(b))(f(a)-f(c))} + \frac{bf(c)f(a)}{(f(b)-f(c))(f(b)-f(a))} + \frac{cf(a)f(b)}{(f(c)-f(a))(f(c)-f(b))} \qquad (6)$$

2.4.2 Brent's algorithm

Whereas Dekker's algorithm uses *linear interpolation* as in False Position method in orderto determine the next estimate based on previous two values, the Brent's algorithm, uses *reverse quadratic estimate*from previous three estimates and additional tests.If$f(b_{k-1})$, $f(a_k)$ and $f(b_k)$ are distinct, it usesreverse quadratic approximation to estimate x_{k+1}. This increases the efficiency slightly. As a consequence, the condition for selecting $b_{k+1}$from linear interpolation and inverse quadratic interpolation is that b_{k+1} has to be closer to b_k. This method is described in algorithm 4 and is implemented in Matlab as a function fZero.

2.4.3. Merits and Drawbacks of Brent's algorithm

Brent proved that his method requires at most N^2 iterations, where N denotes the number of iterations for the Bisection method [6]. Due to the complxity of this algorithm, it may be slower than Dekker's algorithm and use more iterations to resolve Dekker's issues. We wil present a new algortithm whose iterationa are less than N.

Algorithm 4: Brent's Method-- fZero

1. Input: f and [a, b], $\in$, maxIterations
2. Output: root r, k-number of actual iterations, bracketing interval [a_k, b_k]
3. //initialize

4. $b_0 = a$; $b_1 = b$
5. $k = 1$; $[a_k, b_k] = [a, b]$
6. if ($|f(a_k)| < |f(b_k)|$), swap (a_k,b_k); endif
7. k=k+1;
8. repeat
9. $a = a_k$, $b = b_k$, $c = b_{k-1}$,
10. if ($f(a_k) \neq f(b_{k-1})$ and $f(b_k) \neq f(b_{k-1})$)
11. use reverse quadratic estimate
12. $s = \frac{af(b)f(c)}{(f(a)-f(b))(f(a)-f(c))} + \frac{bf(c)f(a)}{(f(b)-f(c))(f(b)-f(a))} + \frac{cf(a)f(b)}{(f(c)-f(a))(f(c)-f(b))}$
13. else if $f(b_k) \neq f(b_{k-1})$
14. $s = b_k - \frac{f(b_k)(b_k - b_{k-1})}{f((b_k) - f(b_{k-1})}$
15. else
16. $s = b - \frac{f(b)(b-a)}{f((b)-f(a)}$
17. end
18. endif
19. if $f(a_k) \cdot f(s) > 0$,
20. $a_{k+1} = s$; $b_{k+1} = b_k$
21. elseif $f(s) \cdot f(b_k) > 0$,
22. $a_{k+1} = a_k$; $b_{k+1} = s$
23. endif
24. To shrink interval,Inverse quadratic interpolation is used,otherwise
25. Midpoint method if Inverse quadratic or Secant method may take longer than Bisection method.
26. $m = (a_k + b_k)/2$;
27. additional tests are performed here to ensure accuracy see [6]
28. if ($b_k < s < (3a_k + b_k)/4$, $b_{k+1} = s$ else $b_{k+1} = m$; it may lead to more iterations
29. ensure that $f(b_{k+1})$ is the smaller value
30. If $|f(a_{k+1})| < |f(b_{k+1})|$, swap ($a_{k+1}$,$b_{k+1}$); endif
31. r = s
32. k = k+1;
33. until $|f(r)| < \in$ or k >maxIterations

Next methods are based on differentiable functions and a start point. Newton-Raphson method assumes that the function is differentiable to ensure that accurate tangent line exists for faster approximation of the root. It speedsup the Bisection method from linear to quadratic convergence.

2.5. Newton–Raphson Method

Newton–Raphson method (1697) is also called a fixed-point iteration method and is due to Isaac Newton and Joseph Raphson. This method requires that the function be differentiable. If the function f(x) is differentiable on the domain of the function, and r_0 is the initial approximation, calleda guess, then by expanding the function by Taylor series,

$$f(x) = f(r_0) + (x - r_0)f'(r_0) + O((x - r_0)^2)$$

if x is closer to root r, f(x) closer to zero, therefore

$$0 = f(r_0) + (r - r_0)f'(r_0)$$

has quadratic error.If the initial guess, r_0, is close to the real root, it is rewarding with an efficient solution, else it can be devastatingly disappointing. This method approximates the function by a linear function. Besides iterating for fixed point, we check that f'(r) does not vanish.This equation gives r_1, the value of r, as the next approximation r_1

$$r_1 = r_0 - \frac{f(r_0)}{f'(r_0)} \quad (7)$$

The Newton-Raphson method assumes that the root is simple, so the derivative $f'(r_0)$ does not vanish. So, the root is not a stationary point.

The first approximation $r_1$can also be obtained by intersection of the tangent line at $(r_0, f(r_0))$ with the x-axis, and is defined by

$$r_1 = r_0 - \frac{f(r_0)}{f'(r_0)} \quad (8)$$

The successive approximations are

$$r_n = r_{n-1} - \frac{f(r_{n-1})}{f'(r_{n-1})} \quad (9)$$

for $n \geq 2$.

This method converges provided $|g'(x)| < 1$ where $g(x) = x - \frac{f(x)}{f'(x)}$ for x in the neighborhood of the root.This is always possible because at the root

$$g'(x) = 1 - \frac{f'(x)^2 - f''(x)f(x)}{f'(x)^2} = \frac{f''(x)f(x)}{f'(x)^2} = 0 \quad (10)$$

2.5.1. Improvements to Newton–Raphson Method

2.5.1.1 Improvement1

The convergence rate is quadratic, and this method is very fast as compared to the Bisection and False Position methods. If $f^{(k)}(x) = 0$ for $x = r$ and integers $k < m$, then m is the multiplicity of the root. If we know the multiplicity, m, of the root, it can be further improved with faster convergence to the root [11]. The updated iteration formula becomes,

$$r_n = r_{n-1} - m\,\frac{f(r_{n-1})}{f'(r_{n-1})} \quad (11)$$

for $n \geq 1$.

For example, if $f(x) = x^3$, it converges in one iteration.

2.5.1.2 Improvement2

It requires that the function be at least twice differentiable. If $f(x) = (x-r)^m h(x)$ has root of multiplicity m, then $f'(x) = m(x-r)^{m-1} h(x) + (x-r)^m h'(x)$. Now $f(x)/f'(x)$ has a *simple* root, so we can use $u(x) = \frac{f(x)}{f'(x)}$

$$x_n = x_{n-1} - \frac{u(x_{n-1})}{u(x_{n-1})} \quad (12)$$

or

$$x_n = x_{n-1} - \frac{f(x_{n-1})\,f'(x_{n-1})}{f'^2(x_{n-1}) - f(x_{n-1})\,f''(x_{n-1})} \quad (13)$$

for $n \geq 1$.

2.5.1.3 Improvement 3 Halley's Method

Another improvement to Newtons method is Halley's method [16], named after Edmond Halley.Itrequires 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 [17]

$$0 = f(x_{k+1}) = f(x_k) + (x_{k+1}-x_k)f'(x_k) + \frac{(x_{k+1}-x_k)^2}{2} f''(x_k) + O((x_{k+1} - x_k)^3) \quad (14)$$

$$0 = f(x_{k+1}) = f(x_k) + (x_{k+1} - x_k)f'(x_k) + O((x_{k+1} - x_k)^2) \quad (15)$$

Multiply the second by $(x_k - r)f''(r)$, first by $2f'(r)$, and subtract second from first, we get

$$x_{k+1} = x_k - \frac{2\,f(x_k)df(x_k)}{2\,df(x_k)^2 - f(x_k)ddf(x_k)} \quad (16)$$

Algorithm 5

```
1.  Input: f , initial guess a, ∈, maxIterations
2.  Output: root r, k-number of actual iterations
3.      // initialization
4.      x=zeros(1,maxit);
5.      k = 1;
6.      x_k=a;
7.      x_{k+1} =x_k - (2 f(x_k)df(x_k)) / (2 df(x_k)^2 - f(x_k)ddf(x_k))
8.      roots(k) = x_k;
9.      ea(k) =f(x_k);
10.     k=2;
11.     while ((abs(x_k-x_{k-1}) >= eps)&&(k<maxIterations))
12.     if abs(abs(x_k-x_{k-1})< eps
13.     root = x_k;
14.             return
15.         end
16.     x_{k+1} = x_k - (2 f(x_k)df(x_k)) / (2 df(x_k)^2 - f(x_k)ddf(x_k))
17.         ea(k) =f(x_k);
```

```
18.             roots(k) = x_k;
19.         k=k+1;
20.         end
21.         root=x_k;
22.         iter=k;
23.     end
```

2.5.3. Merits and Drawbacks of Newton–RaphsonMethod

The *firstpitfall* is that it may fail if the derivative, f′(x), is near zero at some iteration, the approximation value will overshoot. For example, Newton–Raphson method fails to compute r_1 forf(x) =x^2-1 where r_0=0. But in some cases where the singularity in f'(x) cancels with f(x), it does not create a problem. For example, f(x) =x^3 with r_0 =0, does result in r_1 =0.

The *secondpitfall*is that if the root multiplicity, m, has to be known apriori or calculated, its calculation is laborious and is not practical.

2.6. Secant Method, and Enhancements

The *Secant method*also requires that the function be differentiable, though it does not use the derivative explicitly. Thus, in the absence of a derivative of the function, theSecantmethod replaces the Newton–Raphson method derivative by divided difference. Indeed, it does not useeither differentiability or bracketing.

The convergence rate of the Secantmethod is *super linear*.Thus, the convergence rate is between the convergence rate of the Bisectionmethod and the Newton–Raphson's method. The Secantmethod requires two initial values, whereas Newton–Raphsonrequires only one starting value. If the function f(x) is differentiable, and r_0, r_1 are two initial guesses, then the approximations of the Secantmethod can be written as

$$r_n = r_{n-1} - \frac{f(r_{n-1})(r_{n-1}-r_{n-2})}{f(r_{n-1})-f(r_{n-2})} \quad (17)$$

for $n \geq 2$.

2.6.1Modified Secant Method

If the secant line is closer to the tangent line, we can get a better estimate. The closer the points defining the Secant line, the better the approximation of the secant line is to the tangent line. accomplish this improvement, we replace r_{n-2} by a value even closer to r_{n-1}.Thus, for a better estimate of the slope of the Secant line, we can define a *small positive* constant, δ,and replace r_{n-2} with$r_{n-1}-\delta$ (or even $r_{n-1} + \delta$) before computing the next iteration as

$$r_n = r_{n-1} - \frac{f(r_{n-1})((r_{n-1}-r_{n-2})}{f((r_{n-1})-f(r_{n-2})} \quad (18)$$

$$r_n = r_{n-1} - \frac{f(r_{n-1})\delta}{f(r_{n-1})-f(r_{n-1}-\delta)} \quad (19)$$

Or

$$r_n = r_{n-1} - \frac{f(r_{n-1})\delta}{f(r_{n-1}+\delta)-f(r_{n-1})} \quad (20)$$

The success of this method depends on the choice of sufficiently small real positive δ. For practical reasons, we choose positive δ, such that $0 < \delta < |r_{n-1} - r_{n-2}|$.

2.6.2. Merits and Drawbacks of Secant Method

It has the advantage of finding a root quickly, but the choice of delta must be made adaptively, otherwise, the algorithm runs the risk of missing the root.

Some researchers, experimented onf(x) =x − cos(x) on a closed interval [0,1] and concluded that the Secant method is better than the Bisectionand Newton–Raphson methods[13], [14]. It is not accurate to make a conclusion from one function. We show that this statement is *not accurate*, see Tables1,2.

2.7. Effectiveness and Efficacy of Root Approximation Iterations

2.7.1. Methods for Root-Error Calculation for Termination Criterion

There are various ways to measure error when approximating a root of an equation at successive iterations to continue to a more accurate approximate root. To accurately determine r_n for which $f(r_n) \cong 0$, we proceed to analyze as follows.

The iterated root approximation error can be

$$\text{AbsoluteRootError} = |r_n - r_{n-1}| \quad (21)$$

Or

$$\text{RelativeRootError} = \frac{|r_n - r_{n-1}|}{|r_n|} \quad (22)$$

There are various reasons that relative error is not the best way in numerical methods.Since a root can be zero, in order to avoid division by small numbers, it is preferable to use absolute error $|r_n - r_{n-1}|$for convergence testing. Another reason is that it does not work all the time. For example,if $r_n = 2^{-n}$,then $\frac{|r_n - r_{n-1}|}{|r_n|}$ is always 1.It can never be less than 1.

Since the function value is expected to be zero at the root, an alternate, cognitively more appealing error test, is to use $|f(r_n)|$for error consideration. There are three versions of this concept.They are

$$\text{TrueValueError} = |f(r_n)| \quad (23)$$

Or

$$\text{AbsoluteValueError} = |f(r_{n-1}) - f(r_n)| \quad (24)$$

Or

$$\text{RelativeValueError} = \frac{|\mathbf{f(r_n) - f(r_{n-1})}|}{|\mathbf{f(r_n)}|} \quad (25)$$

for comparison criteria.Since $f(r_n)$ is to be close to zero, in order to avoid division by small numbers, we discard using $\frac{|\mathbf{f(r_n) - f(r_{n-1})}|}{|\mathbf{f(r_n)}|}$. Since $|f(r_{n-1}) - f(r_n)|$ can beclose to zero without $|f(r_n)|$ being close to zero, we discard using $|f(r_{n-1}) - f(r_n)|$ in favor of using only $|f(r_n)|$, trueValue error. For example, $f(r_n) = (n - 1)/n$ is such an example.To avoid using the latter two criteria for this reason, we exploit the first one, $|f(r_n)|$, see Fig.3.

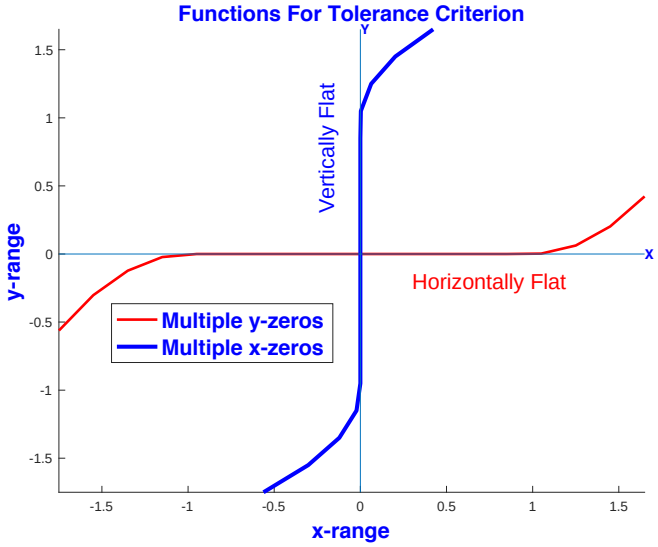


Figure3. Horizontally and vertically flat functions defy natural functions being flat at crossing points.

Now we are left with two options for error analysis $|r_n - r_{n-1}|$ and $|f(r_n)|$. Since r_n andr_{n-1} can be closer to each other without $f(r_n)$ being closer to zero, between the options $|r_n - r_{n-1}|$ and $|f(r_n)|$, we find that $|f(r_n)|$is the only reliable metric for analyzing approximation error becausewe can have a sequence $\{r_n\}$ where$|r_n - r_{n-1}|$ goes to zero, but $|f(r_n)|$ does not go to zero. For example, if $r_n = 1 + 1/n$, and $f(r_n) = r_n$, then$|r_n - r_{n-1}| \to 0$, but $|f(r_n)| \to 1$.Hence, we use this *absolute*criterion, $|f(r_n)| < \epsilon_s$for error analysis for all the methods uniformly in order to circumvent pathological cases. It is not uncommon to use *absolute relative* criteria, $|f(r_n)| < \epsilon_s \max(|f(r_0)|, |f(r_1)|)$, to optimize the number of iterations [10]. See Fig.3, if $f'(x) = 0$ near the root, $f(x)$ can be zero for range of value, the tolerance on $|r_n - r_{n-1}|$ is preferable. If $f'(x) =$ infinite near the root, x can be approximately zero for a range of values without the function $f(x)$ being close to zero, the tolerance on $|f(x)|$ is most desirable. Either way, it is preferable to use both for a reliable conclusion.

2.7.2. Stopping Criteriafor Halting Condition

Stopping criteria plays a major role in simulations. We start with an initial guess(es), and by iteratively applying the selected criteria, we get a sequence of root values of an increasingly accurate estimation of the root, hopefully, converging to a real root. The tradeoff between accuracy and efficiency is the accuracy of the outcome.In order to obtain n significant digit accuracy [11], let $\epsilon_s$be the stopping error and let ϵ_a be the approximation error at any iteration.If $\epsilon_a < \epsilon_s$, the algorithm stops iterating.With $\epsilon_S = 5/10^{n-1}$, we have n significant digit accuracy in the outcome.We have seen that in pathological cases, ϵ_a may never be useful and we have to specify upper bound on

the number of iterations. In all our experiments, we test both the error tolerance and iteration bound to curtails unnecessary iteration.The algorithm terminates whichever test is satisfied first.

3. Hybrid Algorithm

The new algorithm has reliability of Bisection method and speed of False Position method which works all the time. 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 method solely, we selectively apply the most relevant method at each step to redefine the approximate root and bracketing interval.Thus, we construct a new hybrid algorithm, Algorithm 5, that performs better than the Bisection and False Position methods.It also outperforms the Newton–Raphson and Secantmethods (Tables1,2) 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 mid pointand secant point, and the better of the two is selected for the next approximation, in addition the common enclosing interval is defined as the new interval. This prevents the unnecessary iterations performed in either method (see Tables 1, 2). This method does not require differentiability of the function as required by Newton–Raphson and Secant methods.

Algorithm 5: Hybrid Bisection and False Position

1. Input: f and [a, b], $\in_s$, maxIterations
2. Output: root r, k-number of iterations, bracketing interval $[a_{k+1}, b_{k+1}]$
3. //initialize
4. $k = 0$; $a_1 = a$, $b_1 = b$
5. Initialize bounded interval for bisection and false position
6. $fa_{k+1}=ba_{k+1}=a_1$;$fb_{k+1}=bb_{k+1}=b_1$
7. repeat
8. $fa_{k+1}=ba_{k+1}=a_k$;$fb_{k+1}=bb_{k+1}=b_k$
9. compute the mid point
10. $m= \frac{a_k+b_k}{2}$, and $\in_m = |f(m)|$
11. compute the False Position Secant line point,
12. $s = a_k - \frac{f(a_k)(b_k-a_k)}{f(b_k)-f(a_k)}$and $\in_f = |f(s)|$
13. if $|f(m)| < |f(s)|$,
14. $f(m_k)$ is closer to zero, Bisection method determines bracketing interval $[ba_{k+1}, bb_{k+1}]$
15. r= m
16. $\in_a = \in_m$
17. if $f(a_k)\cdot f(r) > 0$,
18. $ba_{k+1} = r$; $bb_{k+1} = b_k$;
19. else
20. $ba_{k+1} = a_k$; $bb_{k+1} = r$;
21. Endif
22. Else
23. $f(s_k)$ is closer to zero, False Position method determines bracketing interval $[fa_{k+1}, fb_{k+1}]$
24. r= s
25. $\in_a = \in_f$
26. if $f(a_k)\cdot f(r) >0$,
27. $fa_{k+1} = r$; $fb_{k+1} = b_k$;
28. Else
29. $fa_{k+1} = a_k$; $fb_{k+1} = r$;
30. endif
31. endif
32. Since the root is bracketed by both $[ba_{k+1}, bb_{k+1}]$ and $[fa_{k+1}, fb_{k+1}]$, set
33. $[a_{k+1}, b_{k+1}] = [ba_{k+1}, bb_{k+1}] \cap [fa_{k+1}, fb_{k+1}]$
34. $a_{k+1} = \max(ba_{k+1},fa_{k+1})$;
35. $b_{k+1} = \min (bb_{k+1},fb_{k+1})$;
36. outcome: iteration complexity, root, and error of approximation
37. IterationCount = k
38. $r = r_k$
39. error = $\in_a = |f(r)|$

40. k = k + 1
41. until |f(r)| <∈ or k >maxIterations

4. Simplicity and Complexity ofHybrid Algorithm

The new hybrid algorithm is a blend ofthe Bisection algorithm,False Position algorithm, independent of Dekker and Brent hybrid algorithms. The new algorithm differs from all the previous algorithms by tracking the best bracketing interval in addition to the best root approximation. The number of iterations to finda root depends on the criteria used to determinethe root accuracy.The complexity of the new algorithm is batter that the bisection. In other words, it uses fewer iterations than the bisection algorithm whereas Brent's algorithms uses more iterations than Bisection method [6].If f(x) is used, then complexity depends on the function as well as the method. If relative error is used, it does not work for every function, as seen in Section 2.7. Most of the time, absolute error, $\in_s$, is used as the stopping criteria.For functions on interval [a, b] with the Bisectionmethod, the upper bound $n_b(\in)$ on the number of iterations can be predetermined from $\frac{b-a}{2^n}<\in_s$ andis lg $((b-a)/\in_s)$.For the False Position method, it depends on the convexity and location of the root near the endpoint of the bracketing interval.The bound $n_f(\in_s)$for the number of iterations for theFalse Position method cannot be predetermined, it can be less,$n_f(\in_s) < n_b(\in)$ =lg $((b-a)/\in_s)$ (see Table 1)or can be greater, $n_f(\in_s) > n_b(\in_s)$ =lg $((b-a)/\in_s)$(see Fig. 2).The number of iterations, $n(\in_s)$,in the new algorithm is less than $\min(n_f(\in_s),n_b(\in_s))$.Brent's algorithm has complexity of order N^2 if bisection is of order N. The new algorithm complexity of order less that N. This is confirmed from the algorithm and is validated with the empirical computations in Section 5. Newton–Raphson and Secantmethods have quadratic convergence, but are not guaranteed to converge unless additional constraints are imposed on the functions and the initial guess is close to the root.

5. Discussion

Many researchers focused their attention toward using such methods to solve their problems. The roots are calculated, along with the number of iterations within a specified tolerance. All the existing methods are compared. Error analysis is performed. It is determined that the hybridalgorithm (blend of Bisectionand False Position) outperforms the other algorithms.Empirical Testing Evidence shows that we have validated our new algorithm against other methods on examples tested in the literature (Table 1) and validated that the new algorithm outperforms these methods.

5.1. Experiments Using MATLAB

Moaazzam used Microsoft Visual C++ to find roots [9],we used MATLAB R2018b64bit on MacMojave2.2GHz16GB. Overall,the experiments havevalidated that the new algorithm outperforms all the classical algorithms in terms of the number of iterations required to find a root. Table 1is a listing of (1) functions, (2) the domain of the function, (3) algorithm and (4) number of iterationson functions used in the literature[13, 15-20].We have demonstrated that the new algorithm is an improvement on all of these equations.Interestingly, Ehiwario [13], [15] compared Bisection, Newton–Raphson and Secant method, and claimed the Secant method was better. That is also *false* as Table 1 shows that among the classical algorithms, Newton–Raphson outperforms them.

Though the root approximation is almost the same in all methods, as expected, yet the number of iterations used to reach the root shows the improvement by the new algorithm. Tables 1, 2 are detailed MATLAB output tables to support Table 1. Since it is hard to interpret the numbers in the tables of the algorithm outputs, graph plots are a preferred way to visualize the algorithm behavior. The plots of the execution of the algorithms are given in Fig. 4. The Tables 1, 2 show that the new hybridalgorithm is faster than other algorithms.

In Tables 1, 2 below, the functions found in the literature and tested by other researchers are used to authenticate the new algorithm [13,15-20]. This table describes algorithms, the function tested by each algorithm, the function interval or start point defining interval and error-tolerance used. The derivative based algorithms can fail if the initial guess in not sufficiently close the to the real root. The new algorithm is reliable, it never fails as long as the function is atleast continuous and has opposite signs at the end points of the root-enclosing interval. The error-tolerance is the same in each case, the root approximations similar as is expected, the only difference is in the complexity of iterations.Brent's algorithm has complexity of order N^2 if bisection is of order N. The new algorithm complexity of order less that N.

Table 1. Summary of Interval-based Algorithms, Functions, and number of iterations to find root.

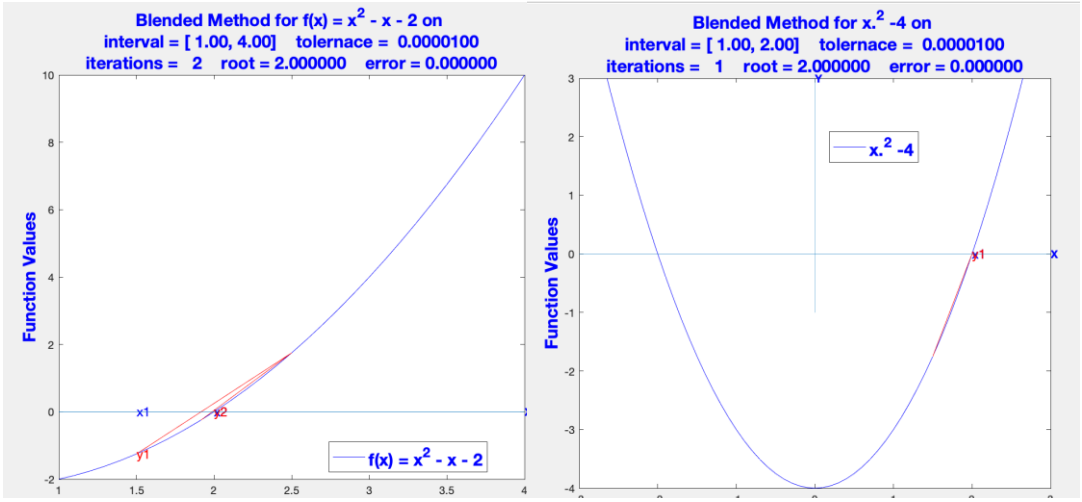
	Number of Iterations			Error Tolerance = 0.00000000001					
	[0.1,1.5]			[-2, 1]			[3.1, 4]		
Intervals	[1, 4]			[0.2, 2]			[0, 1]		
		[1, 2]			[0.2, 4]			[0.3, 2]	
Functions →	$8 - x^9$			x^3-x+3			1/(x-3)-6		
	$x^2 - x - 2$			x^3-x^2-x-1			x-cos(x)		
Algorithms ↓		$x^2 - 4$			$4x^3-16x^2+17x-4$			x+log(x)	
Bisection	21	40	19	40	40	40	40	39	40
FalsePosition	30	32	1	20	15	40	34	10	17
Dekker	37	9	2	10	8	10	11	8	7
Brent	17	14	1	11	17	17	14	12	9
Hybrid	8	2	1	9	8	10	9	7	7

The derivative based algorithms can fail if the start point is not appropriately chosen. To test theminterval-based functions, if differentiable, we can select the start points from the end points of the interval. Table 2 shows the comparison of all the ten algorithms and it shows that the hybrid algorithm still succeeds on these methods.

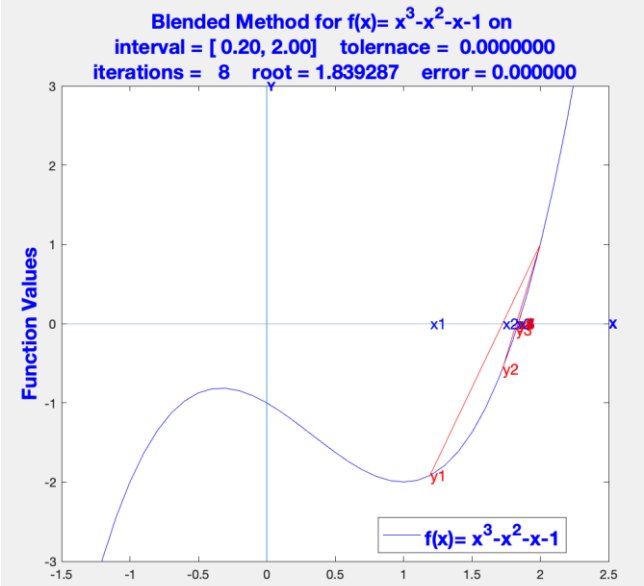
Table 2 Comparisons of iterations for ten algorithmson three functions

Number of Iterations Error Tolerance = 0.00000000001			
Intervals	[1, 4]	[1, 2]	[0.2, 2]
Functions →	$x^2 - x - 2$		
		$x^2 - 4$	
Algorithms ↓			$4x^3-16x^2+17x-4$
Bisection	40	19	40
FalsePosition	32	1	15
Dekker	9	2	8
Brent	14	1	17
Hybrid	2	1	8
Newton	6	4	18
RQI	4	3	8
Halley	5	4	10
Secant	9	2	9
SecantModified	7	5	24

The sample of executions of thesealgorithms with new algorithm are given in Fig. 4 for illistration purpose.



(a) (b)



(c)

Figure4 . Sample exections confirming the exection of Hybrid algorithm on functions in table, x^2- x - 2 = 0 [1,4],x^2- 4 = 0 [1,4], x^3 - x^2– x- 1 = 0 [0.2, 2]

6. Conclusions

We have designed and implemented a new hybrid algorithm, a dynamic blend of the Bisectionand RegulaFalsimethods. It is benchmarked using Bisection, False Position, Dekker, Brent algorithms on various functions cited in the literature.The algorithm was implemented in MATLAB R2018B 64 bit (maci64) on a MacBook Pro MacOS Mojave2.2GHz intel Core i716 GB2400MHz DDR4 Radeon Pro555X 4GB. The number of iterations required to determine the root clearly shows the improvement of the new algorithm (Tables 1,2).

New reliable efficient algorithms are emerging, this was an attempt for interval based continuous functions, there is scope to improve point passed algorithms for minimally differentiable functions. Clearly, asdesired and expected, the root approximation accuracy is almost the same in all algorithms.The implementation validates that the new algorithm outperforms both the Bisection and False Position algorithms all the time in terms of number of *computation iterations*. It is also observed that the new algorithm outperforms Secantmethod and Newton–Raphson method (Tables 2).

References

[1]. Datta, B.N. Lecture Notes on Numerical Solution of Root Finding Problems. 2012. Available online: www.math.niu.edu/~dattab (accessed on 15 January 2019).

[2]. Calhoun, D. Available online: https://math.boisestate.edu/~calhoun/teaching/matlab-tutorials/lab_16/html/lab_16.html (accessed on 13 June 2019).

[3]. Thinzar, C.; Aye, N. Detection the storm movement by sub pixel registration approach of Newton Raphson method. *Int. J. E Educ. E Bus. E Manag. E Learn.* 2014, *4*, 28–31.

[4]. T. J. Dekker and W. Hoffmann (part 2), *Algol 60 Procedures in Numerical Algebra, Parts 1 and 2*, Tracts 22 and 23, Mathematisch Centrum Amsterdam, 1968.

[5]. Dekker, T. J. (1969), "Finding a zero by means of successive linear interpolation", in Dejon, B.; Henrici, P. (eds.), *Constructive Aspects of the Fundamental Theorem of Algebra*, London: Wiley-Interscience, ISBN 978-0-471-20300-1https://en.wikipedia.org/wiki/Brent%27s_method#Dekker's_method

[6]. Brent's Method https://en.wikipedia.org/wiki/Brent%27s_method#Dekker's_method (accessed 12 December 12,2019

[7]. Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. (2007). *"Van Wijngaarden–Dekker–Brent Method"*. Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. *ISBN 978-0-521-88068-8*.

[8]. Sivanandam, S.; Deepa, S. Genetic algorithm implementation using matlab. In *Introduction to Genetic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 211–262, doi: 10.1007/978-3-540-73190-0_8.
[9]. Moazzam, G.; Chakraborty, A.; Bhuiyan, A. A robust method for solving transcendental equations.*Int. J. Comput. Sci. Issues* 2012, *9*, 413–419.
[10]. Gerald W. Recktenwald, Finding the Roots of f x) = 0 - Computer Action Team web.cecs.pdx.edu/~gerry/nmm/course/slides/ch06Slides.pdf (accessed 1 December 2019)
[11]. Chapra, S.C.; Canale, R.P. *Numerical Methods for Engineers*, 7th ed.; McGraw-Hill Publishers: Boston, MA, USA, 2015.
[12]. Mathews, J.H.; Fink, K.D. *Numerical Methods Using Matlab*, 4th ed.; Prentice-Hall Inc.: Upper Saddle River, NJ, USA, 2004; ISBN 0-13-065248-2.
[13]. Nayak, T.; Dash, T. Solution to quadratic equation using genetic algorithm. In Proceedings of the National Conference on AIRES-2012, Vishakhapatnam, India, 29–30 June 2012.
[14]. Srivastava, R.B.; Srivastava, S. Comparison of numerical rate of convergence of Bisection, Newton and Secant methods. *J. Chem. Biol. Phys. Sci*. 2011, *2*, 472–479.
[15]. Ehiwario, J.C.; Aghamie, S.O.; Comparative study of Bisection, Newton-Raphson and Secant methods of root-finding problems. *IOSR J. Eng.* 2014, *4*, 2278–8719.
[16]. Halley at Wikipedia https://en.wikipedia.org/wiki/Halley%27s_method#Cubic_convergence (accessed 12 November 2019)
[17]. Halley's method - Wikipediahttps://en.wikipedia.org/wiki/Halley's method(accessed 22 December 2019)
[18]. Iwetan, C.N.; Fuwape, I.A.; Olajide, M.S.; Adenodi, R.A. Comparative study of the Bisection and Newton methods in solving for zero and extremes of a single-variable function. *J. NAMP* 2012, *21*, 173–176.
[19]. Azizul Hasan, R.B.Srivastava, Najmuddin Ahmad, *An Improved Method Based on Cubic Spline Functions for solving Non-linear Equations*, Journal of Chemical, Biological, and Physical Sciences,p.528-537, Jan 2014. (accessed on 11 June 2019).
[20]. Harder, D.W. Numerical Analysis for Engineering. https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/10RootFinding/falseposition/(accessed on 11 June 2019).