International Journal of Latest Research in Engineering and Technology (IJLRET)
ISSN: 2454-5031
www.ijlret.com || Volume 09 - Issue 11 || November 2023 || PP. 23-35

A Most Efficient Hybrid Algorithm for Transcendental and Non-Linear Equations

Chaman Lal Sabharwal
Missouri University of Science and Technology
Rolla, Missouri - 65409, USA

Abstract: Finding roots of an equation is a fundamental problem in diverse fields. Mostly optimization problems lead to solving non-linear equations for optimizing calculation of the value of a parameter, that is the root of the equation. Numerical techniques are used when analytic solution is no available. We design and implement a new algorithm that is proficient hybrid of Quadrisection and Regula Falsi methods. The implementation results validate that the new algorithm outperforms the existing hybrid algorithms. Thus, we contribute an essential hybrid algorithm to the repertoire of root finding algorithms**.**
Keywords: Bisection, Trisection, Quadrisection, Regula Falsi, Newton-Raphson, Secant, Hybrid Algorithm.

I. INTRODUCTION

Finding the roots of an equation is a fundamental problem in engineering fields. All disciplines in physical and social sciences, including Computer Science, Engineering (biological, civil, electrical, mechanical), and social sciences (psychology, economics, businesses) etc,, require the optimal solution to recurring non-linear problems. The problems such as minimization, Target Shooting, Orbital Motion, Plenary Motion, Social Sciences, Financial Stock Analysis etc, lend themselves to finding roots of non-linear functional equations [1]. There is a thorough study by Sapna and Mohan in the financial sector away from mathematics [2].

There are classical root-finding algorithms: Bisection, False Position, Newton-Raphson, Secant, Modified Secant methods for finding the roots of an equation f(x) = 0. Every text book on Numerical Methods has details of these methods [3], [4], [5]. Even though classical methods have been developed and used for decades, yet enhancements are progressively made to improve the performance of these methods, barring new methods.

Recently papers are making differing claims on their performance and seeking better performing methods. In response, better algorithms that are hybrid of classical methods Bisection, Trisection, with False Position, Newton Raphson methods been developed, namely Hybrid1 [6],Hybrid2 [7], and Hybrid3 [8]. Inspired by these three algorithms, we have created a new adept algorithm Hybrid4, that is hybrid of Quadrisection and false Position that is at par or better than Hybrid1, Hybrid2, and Hybrid3 in terms of computational efficiency, solution accuracy (less error).and reduced iteration count required to terminate within the specified error tolerance. We present Hybrid4 that further optimizes these algorithms one step further by reducing the computing time and increasing the efficiency of the algorithm. The new hybrid algorithm is described in Section3 and simulations validating the performance of the new algorithm are in Section 4.

The paper is organized as follows. Section II is brief description of classical methods Bisection, Regula Falsi, Newton-Raphson, Secant; their strengths and pitfalls. In addition, Trisection and new Quadrisection methods are included. Section III describes hybrid algorithms and new algorithm that outpaces the previous hybrid algorithms. Section IV presents experimental results comparing the performance of previous hybrid algorithms with the new Algorithm. Section V is conclusion.

II. BACKGROUND

The classical algorithms Bisection, False Position, Newton-Raphson, Secant methods are readily found in any text book in detail and iterated in most articles[2]. For this reason, their stand alone derivations are passed over to an appendix and skipped in this background section. Thus, for the sake of completeness, these algorithms are delegated to an appendix for reference. To enhance the performance of Bisection method, Bader et.al [7] designed a Trisection method that supersedes Bisection method in iterations performed, computation time, and error of approximation at a small cost on number of function computations. We extend Trisection algorithm to a Quadrisection algorithm that is at par with Trisection algorithm or better at iterations performed, computation CPU time, and error in approximate root. These algorithms are used in conjunction with False Position and Newton-Raphson methods to create hybrid algorithm. The effectiveness and efficacy of root approximation is measured by number of iterations in root calculation and the accuracy of the termination of an algorithm. The metrics for measuring error in the number iterations and stopping criteria are given first.

2.1 Metric for Error Measurement

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

The iterated *root* approximation error can be

RelativeRootError = $\left|\frac{r_n - r_{n-1}}{r_n}\right|$

or

AbsoluteRootError = $|r_n - r_{n-1}|$

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 test. Another reason for discarding relative error this is that if $r_n = 2^{-n}$, then$\left|\frac{r_n - r_{n-1}}{r_n}\right|$is always 1, it can never be less than 1, so the root-error tolerance test cannot be effectively satisfied, that is, this test does not work.

Since 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 instead of r_n. There are three versions for this concept, they are

RelativeValueError =$\left|\frac{f(r_n) - f(r_{n-1})}{f(r_n)}\right|$

or

AbsoluteVlaueError = $|f(r_n) - f(r_{n-1})|$

or

TrueValueError = $|f(r_n)|$

for comparison criteria.

Since $f(r_n)$ is to be close to zero near the root, in order to avoid divide by small numbers, we discard using $\left|\frac{f(r_n) - f(r_{n-1})}{f(r_n)}\right|$. Further, since $|f(r_n) - f(r_{n-1})|$can be close to zero without $|f(r_n)|$ being close to zero, we discard using $|f(r_n) - f(r_{n-1})|$also in favor of using only$|f(r_n)|$, trueValue error. For example, $f(r_n)= (n-1)/n$ is such an example. We avoid using the first two criteria for this reason and exploit the last one, $|f(r_n)|$. Now we are left with two options$|r_n - r_{n-1}|$and $|f(r_n)|$to consider forerror analysis. Again, since r_n and r_{n-1} can be closer to each other without $f(r_n)$ being closer to zero. For example$r_n= 1 +1/n$, $f(r_n) = r_n$. 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 the approximation error. Hence, we use,$|f(r_n)|$, as the criteria for comparing with tolerance error analysis for all the methods uniformly.

2.2 Metric for Stopping Criteria, Halting Condition

Stopping criteria plays a major role in simulations. The iteration termination (stopping) criteria for False Position method is different from Bisection method. Tradeoff between accuracy and efficiency is accuracy of the outcome. In order to obtain n significant digit accuracy [3], let$\in_s$ be stopping error and let$\in_a$ be the approximation error at any iteration. If$\in_a < \in_s$, the algorithm stops iterations. With$\in_s = 5/10^{n-1}$, we have n significant digit accuracy in the outcome. The bisection algorithm is trivial, here trisection and quadrisection algorithm's stopping criteria are described.

Here we describe two enhancements to bisection algorithm. Trisection algorithm [7] and Quadrisection, new algorithm, each algorithm has four comparison tests and seven function evaluation references in implementation. Thus, the Quadrisection algorithm uses the same amount of computation resources as Trisection algorithm.

But the number of iterations bn (Bisection),tn (Trisection),qn (Quadrisection) required by the Bisection, Trisection, and Quadrisection algorithms on [a,b] with stopping tolerance epsare

bn =log{(b-a)/tol} tn=(0.63) log{(b-a)/tol} qn= (0.5) log{(b-a)/tol}.

Trisection algorithm is 37% faster than Bisection algorithm, Quadrisection algorithm takes 13% *fewer* iterations than Trisection algorithm to converge within the desired tolerance.

In addition, as observed below in each algorithm, in each iteration, there is no change in the computation time: seven references to function evaluation and four references to compare test between Trisection and Quadrisection algorithms..

2.3 Algorithms

2.3.1 Trisection Algorithm [7].

- Input: Function *f*(*x*), Initial approximations [*a*,*b*] and absolute **error** *eps*.
- Output: Approximate root ***r***, *enclosing interval*, and number of **iterations** *k*

```
for k=1 to n
    p := (2*a +b)/3; q := (a +2*b)/3;
    if |f(p)| < |f(q)|
        r := p
    else
        r := q
    endif;
    if |f(r)| < eps
        return r,a,b, k;
    else if f(a)*f(p) < 0
        b:=p;
    else if f(p)*f(q) < 0
        a:=p;
        b:=q;
    else
        a:=q;
    end if;
 end for
```

2.3.2Quadrisection Algorithm (new)

- Input: function *f*(*x*), Initial interval [*a*,*b*] and absolute **error** *eps*.
- Output: Approximate root ***r***, *enclosing interval*, and number of **iterations** *k*

```
for k=1 to n
p:= (3*a +b)/4; m := (2*a +2*b)/4; q := (a +3*b)/4;
   r=m;
    if f(a)*f(m)< 0
        b=m; r=p;
        if f(a)*f(p)<0
            b=p
        else
            a=p
         endif
    else
        a=m;r=q;
        if f(a)*f(q)<0
            b=q
        else
            a=q
        endif
    endif
    if |f(r)| < eps
     return r, a,b, k;
    endif
endfor
```

The Trisection algorithms [7],[8] and a Quadrisection algorithm are computationally equivalent in each iteration, but Quadrisection algorithm requires fewer iterations to converge, see Table1,2,3. It shows that in the sequence of innovations, Quadrisection algorithm is ahead of the other algorithms with respect to loop iteration and accuracy in approximation of the root. These benchmark functions appear in recent papers [7],[8], see Tables 1,2,3,4. The functions and the interval of definition are generic unbiased of any algorithm.

Table 1 Summary for Comparison of Methods Iteration Counts
Non-Linear Equation
Function f(x) = x^2 - x - 2, Interval [1, 6]
Max Iterations = 40 Error tolerance = 0.000001000000

Method	Iterations	Error	Root	TimeUsed
FalsePos	28	0.0000007049318	1.999999765023	0.00875641700000
Bisection	22	0.0000007152558	2.000000238419	0.00317154200000
Trisection	14	0.0000009408381	1.999999686387	0.00561808300000
Quadrisection	11	0.0000007152558	2.000000238419	0.00425629100000

Table2Summary for Comparison of Methods Iteration Counts
Non-Linear Equation
Function f(x) = 0.986*x.^3 -5.181* x.^2 +9.067*x-5.289, Interval [0, 2]
Max Iterations = 40 Error tolerance = 0.000001000000

Method	Iterations	Error	Root	TimeUsed
FalsePos	40	0.0013275239622	1.943958024512	0.00745204200000
Bisection	16	0.0000005319648	1.929840087891	0.00250129200000
Trisection	10	0.0000007183544	1.929837931210	0.00333941700000
Quadrisection	8	0.0000005319648	1.929840087891	0.00306541700000

Table3Summary for Comparison of Methods Iteration Counts
Transcendental Equation
Function f= @(x) m*exp(x).*(x-1)- Tanscendental, Interval [-2, 13]
Max Iterations = 40 Error tolerance = 0.000001000000

Method	Iterations	Error	Root	TimeUsed
FalsePos	40	6.4962923172994	-1.999954114040	0.01695079200000
Bisection	28	0.0000004860667	0.999999988824	0.01603791700000
Trisection	18	0.0000005051780	0.999999988385	0.00994154100000
Quadrisection	14	0.0000004860667	0.999999988824	0.01140166700000

Table4 Summary for Comparison of Methods Iteration Counts
Transcendental Equation
Function x-cos(x), Interval [0, 6]

Method	Iterations	Error	Root	TimeUsed
FalsePos	6	0.0000005254348	0.739084819263	0.00571970800000
Bisection	21	0.0000001075021	0.739085197449	0.00284170900000
Trisection	15	0.0000002517506	0.739085283639	0.00391850000000
Quadrisection	12	0.0000004910282	0.739084839821	0.00382875000000

III. HYBRID ALGORITHMS

First we describe the original hybrid algorithm, namely, Hybrid1 [6] based on classical Bisection and False Position algorithms. Since the classical algorithms can be found in any text book, those algorithms are not described here. For reference in hybrid algorithm, and for the sake of completeness, these algorithms are delegated to an appendix.

In Hybrid1 algorithm, at each iteration, more promising root between the Bisection and False Position approximate roots is selected for the next iteration. This curtails the unnecessary iterations in either method. It was succeeded by more efficient Hybrid algorithms [7],[8] following it, leading to our new most proficient algorithm Hybrid4. For convenience in notation, the parameters are prefixed by the name of the algorithm: b for bisection, t for trisection, n for Newton-Raphson, q for quadrisection, and p for false Position

Hybrid1: Bisection and False Position Algorithm [6]

Input: f , [a, b], $\in_s$, maxIterations
Output: root r, k-number of iterations, error of approximation $\in_a$, 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
$pa_{k+1}=ba_{k+1}=a_1$;$pb_{k+1}=bb_{k+1}=b_1$
repeat

$pa_{k+1}=ba_{k+1}=a_k$; $pb_{k+1}=bb_{k+1}=b_k$
compute the mid point the error
m= $\frac{a_k+b_k}{2}$, and $\in_m = |f(m)|$
compute the False Position point and error,
s = $a_k - \frac{f(a_k)(b_k-a_k)}{f(b_k)-f(a_k)}$and $\in_p = |f(s)|$
if $|f(m)| < |f(s)|$,
f(m) is closer to zero, Bisection method determines bracketing interval $[ba_{k+1}, bb_{k+1}]$
r= m
$\in_a = \in_m$
if $f(a_k)\cdot f(r) > 0$,
$ba_{k+1} = r$; $bb_{k+1} = b_k$;
else
$ba_{k+1} = a_k$; $bb_{k+1} = r$;
endif
else
f(s) is closer to zero, False Position method determines bracketing interval $[pa_{k+1}, pb_{k+1}]$
r= s
$\in_a = \in_p$
if $f(a_k)\cdot f(r) >0$,
$pa_{k+1} = r$; $pb_{k+1} = b_k$;
else
$pa_{k+1} = a_k$; $b_{k+1} = r$;
endif
endif
Since the root is bracketed by both $[ba_{k+1}, bb_{k+1}]$ and $[pa_{k+1}, pb_{k+1}]$, set
$[a_{k+1}, b_{k+1}] = [ba_{k+1}, bb_{k+1}] \cap [pa_{k+1}, pb_{k+1}]$ or
$a_{k+1} = \max(ba_{k+1},pa_{k+1})$;
$b_{k+1} = \min(bb_{k+1},pb_{k+1})$;
outcome: iteration complexity, root, and error of approximation
iterationCount = k
$r = r_k$
error = $\in_a = |f(r)|$
k = k + 1
until $|f(r)| <\in_s$ or k >maxIterations

Hybrid2: Trisection and False Position Algorithm[7]

This function implements a blend of trisection and false position methods.
Input: The function f; the interval [a, b] where f(a)f(b) < 0 and the root lies in[a, b],
The absolute error (eps).
Output: The root (x), The value of f(x), Numbers of iterations (n), the interval [a, b] where the root lies in

```
n = 0; a1 := a; a2 := a; b1 := b, b2 := b
while true do
    n := n + 1
    xT1 := (b + 2*a)/3
    xT2 := (2*b + a)/3
    xF := a − (f(a)*(b − a))/(f(b) − f(a))
    x := xT1
    fx := fxT1
if |f(xT2)|< |f(x)|
   x := xT2
if |f(xF)| < |f(x)|
    x := xF
if |f(x)| <= eps
    return x, f(x), n, a, b
if fa * f(xT1) < 0
    b1 := xT1
else if f(xT1) * f(xT2) < 0
```

```
    a1 := xT1
    b1 := xT2
else
    a1 := xT2
if fa*f(xF) < 0
    b2 := xF;
else
    a2 := xF;
a := max(a1, a2) ; b := min(b1, b2)
end (while)
```

Hybrid 3: Trisection and Newton-Raphson Algorithm [8]

This algorithm is along the same lines as Hybrid 2, but with (1) instead of false position method, it uses Newton-Raphson algorithm which requires differentiability of the function, (2) improved iteration count and accuracy in the hybridization step: namely, the common interval in each iteration is computed by analyzing the five function values and then mapped to parameter values for the optimal interval.

The Algorithm is as follows.

- Input: Function $f(x)$, an Initial approximations x_0 and absolute error *eps*.
- Output: Root x and number of iterations n

```
df(x):=f'(x); k:=0;
for k=1:n
  p := (2*a + b)/3;
  q := (a + 2*b)/3;
  if |f(p)| < |f(q)|
     then r := p - f(p)/df(p);
     else r := q - f(q)/df(q);
  end if;
  if |f(r)| < eps
  then
      return r, k;
  else
     find fv:={f(a),f(b),f(r),f(p),f(q)};
     a := xa where fv max -ve;
     b := xb where fv min +ve
  end if;
end.
```

As indicated, this algorithm equires differentiability used in Newton-Raphson algorithm step. In addition, this algorithm uses search from five function values for two values to determine the parameter interval enclosing the root.

In the previous algorithms, two steps are used to coordinate two algorithms to hybridize. At each iteration, determine (1) the promising approximation root (2) the common interval enclosing the approximate root. In Hybrid1 and Hybrid2 algorithms, this simply reduces to intersection of two intervals so that common interval contains the approximate root. No function value is involved in the search for common interval to contain the predicted approximate root. In the Hybrid3 algorithm, it searches among five function values used to determine two function values for the common interval. From these two function values, the function parameters are determined to create the common parameter interval.

Note. Hybrid3 requires differentiability of the function as required by Newton Raphson algorithm.

New Hybrid Algorithm

Hybrid4 provides a more efficient approach to optimization: (1) Quadrisection algorithm is used instead of bisection or trisection, (2) it eliminates the computation of common interval required by the foregoing algorithms used to hybridize, This leads to more efficient algorithm for optimal root approximation and readily available common interval without the calculations required by Hybrid1, Hybrid 2, and Hybrid3.Otherwise it takes computing time, to calculate the common interval. It is based on Occam's razor principle [9], Fig 1.The Occam's razor principle is a heuristic, not a proof. That is, when faced with competing choices, the simplest is the accurate one. It will be shown that Occam's Razor Principle works quite well in this case.

Figure1https://conceptually.org/concepts/occams-razor

Hybrid 4 uses Quadrisection and False Position methods. This is based on Occam's razor principle [9].
Input a_0,b_0,r_o, eps, imax, f
Output k, a_k,b_k,r_k
for k=1:imax
quadrisection iteration step determines
a_k,b_{k-},r_{k-} from a_{k-1},b_{k-1},r_{k-1}
relable
a_k,b_k,r_ktoa,b,r_
False-position iteration step
input isa,b,r instead of olda_{k-1},b_{k-1},r_{k-1}
false position iteration step determines
a_k,b_k,r_{k-} from a,b,r
This makes $[a_k,b_k]$ as the common interval readily available without any computation.
At the same time, using a,b,r instead of old a_{k-1},b_{k-1},r_{k-1}, makes this step more optimal
if $f(r_k)$<eps
returnk,a_k,b_k,r_k
end
endFor

Summarizing the foregoing algorithms, succinctly the *iteration step* in the algorithms are:
Hybrid1
$[ba_k,bb_k,br_k]$= Bisection(a_{k-1},b_{k-1},r_{k-1},f)
$[pa_k,pb_k,pr_k]$=FalsePosition(a_{k-1},b_{k-1},r_{k-1},f)
The results of hybridization step are:
r_k is better of br_k, pr_k,
$[a_k,b_k]$ is common to $[ba_k,bb_k]$, $[pa_k,pb_k]$,
$r_k$belongs to $[a_k,b_k]$

Hybrid 2
$[ta_k,tb_k,tr_k]$=Trisection(a_{k-1},b_{k-1},r_{k-1},f)
$[pa_k,pb_k,pr_k]$=FalsePosition(a_{k-1},b_{k-1},r_{k-1},f)
The outcomes of hybridization are:
r_k is better of tr_k, pr_k,
$[a_k,b_k]$ is common to $[ta_k,tb_k]$, $[pa_k,pb_k]$,
$r_k$belongs to $[a_k,b_k]$

Hybrid3
$[ta_k,tb_k,tr_k]$=Trisection(a_{k-1},b_{k-1},r_{k-1},f)
$[na_k,nb_k,nr_k]$=NewtonRaphson(a_{k-1},b_{k-1},r_{k-1},f)
The upshot of hybridization is:
r_k as better of tr_k, nr_k,
$[a_k,b_k]$ is common from $[ta_k,tb_k]$, $[na_k,nb_k]$ and nr_k by analyzing { $f(ta_k)$, $f(tb_k)$, $f(na_k)$,$f(nb_k)$, $f(nr_k)$})}then finding $[a_k,b_k]$ from { ta_k,tb_k,na_k,nb_k, nr_k}.
$r_k$belongs to $[a_k,b_k]$

Hybrid4 (new contribution)

$[a,b,r]$=Quadrisection(a_{k-1},b_{k-1},r_{k-1},f)
$[pa_k,pb_k,pr_k]$=FalsePosition(a,b,r,f)
The conclusion of hybridization is:
The interval $[a_k,b_k]$ is readily available, because $[pa_k,pb_k]$ is itself the desired interval $[a_k,b_k]$ containing r_k, pr_k is r_k, the desired root. This algorithm is optimal in the number of iterations and the accuracy of approximate root. There is no need to do any work for calculations.

IV. 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 new hybrid algorithm outperforms the other algorithms.

A. Empirical Evidence Testing

We have tested our new algorithm side by side with other hybrid algorithms on diverse examples found in article in then literature to validate that new algorithm outperforms the conclusion made by them. Tables 5-11give a synopsis of some functions.

B. Experiments in Matlab

Some researchers used Mirosoft visual C++ to find roots, we used MatlabR2022A 64 bit (maci64) on MacBook Pro MacOS Sonoma 14.1.116GB Apple M1Pro .Along with several different functions, we use the same quadratic equation for root of x^2-x-2 = 0. Overall we have validated that the new algorithm performs better than all the hybrid algorithms. These tests indicate that hybridization with false position algorithm is still preferable to hybridization with Newton-Raphson possibly because the NR method requires that the function be differentiable.

Tables 5-11 are from the functions used in the literature. In all the function in this table the upperbound on the number of iterations is 40 and error bound is 10^{-6}. These are some benchmark test functions from recent hybrid algorithms.

Table 5 Function f= @(x) m*exp(x).*(x-1)- Transcendental, Interval [-2, 13]

Method	Iterations	Error	Root	TimeUsed
Hybrid1: Bisection-FalsePos	10	0.0000003170838	0.999999992709	0.02568770800000
Hybrid2: Trisection FalsePos	8	0.0000000395461	0.999999999091	0.02214395800000
Hybrid3: Trisection-NewtonRaph	10	6.4960935953574	-2.000000000000	0.02972425000000
Hybrid4: Quadrisection-FalsePos	6	0.0000001375199	0.999999996838	0.01611329200000

Table 6 Function f(x) = 0.986*x.^3 -5.181* x.^2 +9.067*x-5.289 Non-Linear, Interval [0, 2]

Method	Iterations	Error	Root	TimeUsed
Hybrid1: Bisection-FalsePos	7	0.0000008959635	1.929835876074	0.00868341600000
Hybrid2: Trisection FalsePos	5	0.0000002142064	1.929843764485	0.00650604200000
Hybrid3: Trisection-NewtonRaph	5	0.0000001094874	1.929847509586	0.01198429200000
Hybrid4: Quadrisection-FalsePos	4	0.0000000494278	1.929845670976	0.00769437500000

Table 7 Function x-cos(x), Interval [0, 1] -Transcendental

Method	Iterations	Error	Root	TimeUsed
Hybrid1: Bisection-FalsePos	5	0.0000000059872	0.739085129638	0.02441987500000
Hybrid2: Trisection FalsePos	4	0.0000001695627	0.739085031900	0.02608075000000
Hybrid3: Trisection-NewtonRaph	4	0.0000002673808	0.739085292978	0.02653025000000
Hybrid4: Quadrisection-FalsePos	2	0.0000005367912	0.739084812477	0.02592866700000

Table 8 Function f(x) = 1./(x-3)-6, Interval [3.100000e+00, 4] -Non-linear

Method	Iterations	Error	Root	TimeUsed
Hybrid1: Bisection-FalsePos	7	0.0000004800000	3.166666680000	0.00750858300000
Hybrid2: Trisection FalsePos	3	0.0000000000000	3.166666666667	0.00563233400000
Hybrid3: Trisection-NewtonRaph	6	0.0000001279518	3.166666663112	0.01658004200000
Hybrid4: Quadrisection-FalsePos	3	0.0000000034154	3.166666666762	0.00622525000000

Table 9 Function f(x) = x^2 - 2 - Non-Linear, Interval [1, 3] - quadratic

Method	Iterations	Error	Root	TimeUsed
Hybrid1: Bisection-FalsePos	6	0.000000021751	1.414213554683	0.00548650000000
Hybrid2: Trisection FalsePos	5	0.000000049645	5 1.414213544821	0.00401850000000
Hybrid3: Trisection-NewtonRaph	6	0.000000200028	7 1.414213633094	0.01707441700000
Hybrid4: Quadrisection-FalsePos	3	0.000000144004	5 1.414213511460	0.00423837500000

Table 10 Function f(x) = x^2 - x - 2, Interval [1, 5] - quadratic

Method	Iterations	Error	Root	TimeUsed
Hybrid1: Bisection-FalsePos	5	0.000000002095	5 1.999999999302	0.01563458300000
Hybrid2: Trisection-FalsePos	5	0.000000028479	2 1.999999990507	0.01726020800000
Hybrid3: Trisection-NewtonRaph	6	0.000009078811	2.000000302627	0.04992987500000
Hybrid4: Quadrisection-FalsePos	1	0.000000000000	2.000000000000	0.00655712500000

Table 11 Function f(x) = x^2 - x - 2, Interval [1, 6] - quadratic

Method	Iterations	Error	Root	TimeUsed
Hybrid1: Bisection-FalsePos	6	0.000000065862	8 1.999999978046	0.00301462500000
Hybrid2: Trisection FalsePos	2	0.000000000000	2.000000000000	0.00264712500000
Hybrid3: Trisection-NewtonRaph	6	0.000000636530	9 2.000000212177	0.00593533300000
Hybrid4: Quadrisection-FalsePos	2	0.000000011351	9 1.999999996216	0.00254025000000

V. CONCLUSION

We have designed and implemented a new algorithm, a proficient hybrid of Quadrisection and Regula Falsi methods. The algorithm was implemented in Matlab R2022A 64 bit (maci64) on MacBook Pro MacOS Sonoma 14.1.116GB Apple M1Pro. The implementation tests in Tables 5-11 indicate that it outperforms all the above cited algorithms all the time by a considerable margin. The experiments on numerous datasets used in the literature justify that the new algorithm is effective both conceptually and computationally. Thus, this paper provides a fastest algorithm to the repertoire of hybrid algorithms.

VI. APPENDIX

There is no universal algorithm optimal for root approximation that works on all the functions on all the domain intervals. We provide a summary of classical methods here for reference. In the paper, we have provided a new hybrid algorithm that is based on the classical methods and outperforms both the classical and hybrid methods.

There are four classical methods for finding roots of non-linear equations: Bisection, Regula Falsi, Newton-Raphson, Secant. For completeness, we describe these methods briefly for (1) root approximation, (2) error calculation, and (3) termination criteria. Then we use Occam's razor principle to select the optimal method for error calculation and termination criteria.

We constrain 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.

A. Bisection Method

The Bisection method is (1) based on binary chopping of irrelevant subintervals, (2) virtually binary search, and (3) guaranteed to converge to the root. Bisection method is static, the *length* of the subinterval at each iteration is independent of the function. No 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 the number of iterations can be predetermined quickly [10].

Problem: If a function f:[a, b]→R (1) is continuous and (2) f(a) and f(b) are of opposite signs, i.e., $f(a)\bullet f(b) < 0$, then there exists a root $r \in [a, b]$ such that $f(r) = 0$.

Let $[a_1,b_1] = [a, b]$ be the initial interval of continuity of f. The first approximate root is

$r_1 = \frac{a_1+b_1}{2}$, middle point of the interval $[a_1,b_1]$

the actual root lies in the interval $[a_1, r_1]$ or $[r_1,b_1]$,

if $f(r_1)=0$, r_1 is the root.

if $f(a_1)$ and $f(r_1)$ are of the opposite sign, $f(a_1)\bullet f(r_1) < 0$, true root lies in $[a_1, r_1]$

if $f(r_1)$ and $f(b_1)$ are of the opposite sign, $f(r_1)\bullet f(b_1) < 0$, true root lies in $[r_1,b_1]$,

The new interval is denoted by $[a_2,b_2]$

At each iteration, new root and next sub-interval is generated.
In general, for each iteration k, the approximation
$r_k=\frac{a_k+b_k}{2}$ is middle point of $[a_k, b_k]$,
Either r_k is the root or
if $f(a_k)\bullet f(r_k) < 0$, the root lies in $[a_k, r_k]$
else if $f(r_k)\bullet f(b_k) < 0$, the root lies in $[r_k, b_k]$

Then the new interval is denoted by $[a_{k+1},b_{k+1}]$ with r_k as one of its end points

A.1 Advantages of Bisection Method

Since the method brackets the root, at each iteration the length of root-bracketing interval is scaled to half the length. Thus, the method guarantees the decrease in the error in the approximate root at each iteration. The convergence of Bisection method is certain as it is simply based on halving the bracketing interval containing the root.

A.2 Drawbacks of Bisection Method

Though the convergence of Bisection method is guaranteed, its rate of convergence is too slow and as such it is quite difficult to extend to use for systems of equations. If one of the initial endpoints is closer to the root, Bisection method does not take advantage of this information, it will take larger number of iterations to reach the root [10].

B. False Position (Regula Falsi) Method[11], [12]

Motivation for innovative methods arises from the poor performance of Bisection method. The False Position method is known by various names: Double False Position, Regula Falsi method, linear interpolation method. It is a very original method for solving equations in one unknown. This method differs from Bisection method in the way the estimates are calculated. False Position method is a dynamic method, it takes advantage of the location of the root to make a conceivably better appropriate selection. Unfortunately, this method is not satisfactory as expected, for all functions, see Figure 2,3,4,5.

Here also the function f:[a, b]→R (1) is continuous and (2) f(a) and f(b) are of opposite signs, i.e., $f(a)\bullet f(b)<0$. The algorithm uses a, b as the two initial estimates a_1, b_1 for the root. The False Position method uses two start values $r_o = a_1$, $r_1 = b_1$, to compute successive values

$$r_n = r_{n-1} - \frac{f(r_{n-1})(r_{n-1}-r_{n-2})}{f(r_{n-1})-f(r_{n-2})} \text{ for natural number } n \geq 2$$

the approximations, r_n , are ensured to be bracketed in $[a_n, b_n]$ depending on $f(a_{n-1})\bullet f(r_n)<0$ for $[a_{n-1},r_n]$ and $[r_n, b_{n-1}]$ depending on as in the case of Bisection method,

B.1 Justification: Error in Bisection method is straight forward; in each iteration root approximation error is halved, i.e. root approximation error in the nth step is no larger than $\frac{b-a}{2^n}$. This is not the case for *False Position* method. If a is sufficiently close to the root r, then f(a) is close to f(r) = 0 due to continuity of f. The slope of the secant line slope, $\frac{f(b)-f(a)}{b-a}$, is approximately equal to $\frac{f(b)}{b-r}$. The closer b is to r, the closer the secant line is to tangent, f'(r). Though, it is not required that f be differentiable. The closer b is to r, faster the convergence of iterations [12].

B.2 Advantages of False Position Method

It is guaranteed to converge due to decreasing length of root-bracketing interval. It is fast when you know the linear nature of the function.

B.3 Drawbacks of False Position Method

For False Position method, there is no way to tell the number of iterations needed for convergence. The False Position method is expected to be faster than Bisection method. If we cannot ensure that the function can be interpolated by a linear function, then applying the False Position method can result *in worse* results than the Bisection method. The problem occurs when the function is convex, concave up or concave down According to [10], for concave down function, left end point remains stationary and right end point updates in each iteration. For concave up function, right end point remains stationary and left end point updates in each iteration. This is not an accurate statement, it works some of the functions, not all the functions. Figures 2, 3, 4, 5 contradict this statement. It depends on the convexity of the function, not concavity up or down. When the

root is very close to the end points of the interval, convergence can become extremely slow. Visuals are helpful insight to comprehend the behavior of algorithms.

In these examples, four functions are used with same tolerance and the upper limit of 10 iterations for display purposes. The purpose is to show how the algorithms work for False Position method. In the interest of simplicity of plots, we terminated the algorithms before reaching the error-tolerance.

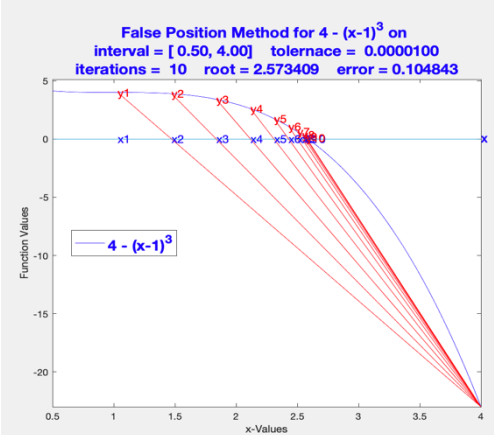

Figure 2. Convex Function Concave up, right end point fixed

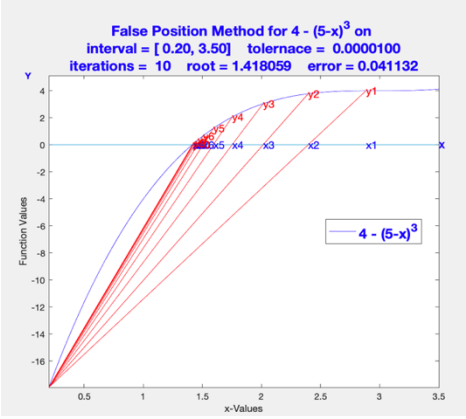

Figure 3. Convex Function Concave up, right end point fixed

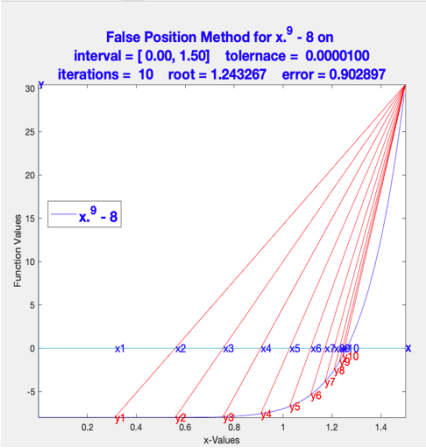

Figure 4. Convex Function Concave down, right end point fixed

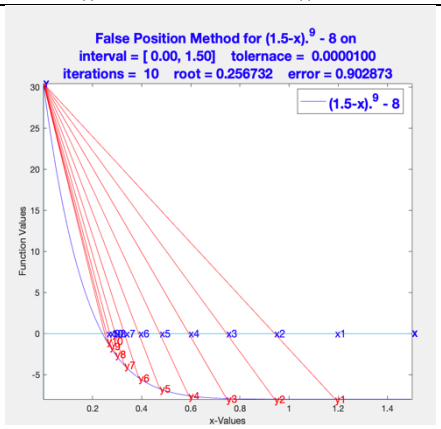

Figure 5. Convex Function Concave down, left end point fixed

C. Newton-Raphson Method [3]

Newton-Raphson method is also called a fixed point iteration method. 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 initial guess, then the first approximation r_1 is defined by

$r_1 = r_0 - f(r_0)/f'(r_0)$

and successive approximations are

$r_n = r_{n-1} - f(r_{n-1})/f'(r_{n-1})$ for natural numbers $n \geq 2$.

This method converges provided $|x - f(x)/f'(x)| < 1$ for x in the domain of definition. For functions where there is a singularity and it reverses sign at the singularity, Newton-Raphson method may converge on the singularity.

C.1 Advantages of the Newton-Raphson method

The convergence rate is linear and this method is very fast as compared to Bisection and False Position methods. If we know the *multiplicity*, m, of the root, it can be further improved with faster convergence to the root [3]. The updated iteration formula becomes,

$r_n = r_{n-1} - m\, f(r_{n-1})/f'(r_{n-1})$ for natural numbers $n \geq 1$.

Note. If $f^{(k)}(x) = 0$, for all integers $k < m$, then multiplicity of root is m.

C.2 Disadvantages of the Newton-Raphson method;

The only pitfall is that it fails if the derivative, f'(x), is near zero at some iteration. For example, Newton-Raphson method fails to compute r_1 for $f(x) = x^2 - 1$ where $r_0 = 0$. But it does not create a problem in some cases where singularity in f'(x) cancel with f(x). for example, $f(x) = x^3$ with $r_0 = 0$ does result in $r_1 = 0$.

D. Secant Method, Modified Secant Method[13]

The Secant method also requires that the function be differentiable. When it is not easy to compute the derivative of the function, the Secant method approximates the derivative with the slope of a secant line. That is, in the absence of derivative of the function, Secant method is a modification of the Newton-Raphson method. In fact, it does not need differentiability as well as bracketing. It is similar to False Position method. The only difference is that False Position method ensures that approximations are bracketed and Secant method simply uses the last two values to approximate the tangent. Thus, the Secant method is not guaranteed to converge.

The convergence rate of the Secant method is superlinear. Thus, the convergence rate is between that of the Bisection method and the Newton-Raphson's method. The Secant method requires two initial values, whereas Newton-Raphson method required only one start values. If the function f(x) is differentiable, r_1 can be computed from r_0. Now r_0, r_1 are two is initial guesses, then the approximations of the secant method can be written as

$$r_n = r_{n-1} - \frac{f(r_{n-1})(r_{n-1} - r_{n-2})}{f(r_{n-1}) - f(r_{n-2})}$$ for natural numbers $n \geq 2$

For still better estimate of the slope of the Secant line, we can define a small constant delta, δ, so that it can uniformly replace r_{n-2} with $r_{n-1} - \delta$ as

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

The success of this method is questionable if δ is not chosen sufficiently small. One way will be set $\delta < |r_{n-2}-r_{n--1}|$

D.1 Advantages and Shortcomings

It has the advantage for finding a bracketing-interval quickly where the root lies, but choice of delta must be made adaptively, else the algorithm runs the risk of missing the root.

Some researchers, experimented on f(x) = x - cos(x) on a close interval [0,1] and concluded that secant method is better than Bisection and Newton-Raphson method [14], [15]. It is not accurate to make a general conclusion statement from one function.

VII. REFERENCES

[1]. Donna Calhoun
[2]. https://math.boisestate.edu/~calhoun/teaching/matlab-tutorials/lab_16/html/lab_16.html accessed December 2023
[3]. S. Sapna, Biju R. Mohan, Comparative Analysis of Root Finding Algorithms for Implied Volatility Estimation of Ethereum Options, Computational Economics, https://doi.org/10.1007/s10614-023-10446-8, springer, August, 2023, pp.1-37.
[4]. Steven C Chapra and Raymond P Canale, Numerical Methods for Engineers, 7th Edition, McGraw-Hill Publishers, 2015.
[5]. C. Thinzar, and N. Aye, "Detection the storm movement by sub pixel registration approach of Newton Raphson method" International Journal of e-Education, e-Business, e-Management and e-Learning, Vol. 4, No. 1, 2014.
[6]. Sivanandam S., Deepa S. (2008) Genetic Algorithm Implementation Using Matlab. In: Introduction to Genetic Algorithms. Springer, Berlin, Heidelberg, pp. 211-262, DOIhttps://doi.org/10.1007/978-3-540-73190-0_8.
[7]. Sabharwal, C. L. (2019). Blended root finding algorithm outperforms bisection and regulafalsi algorithms. Mathematics, 7(11), 1118.
[8]. Badr, E., Almotairi, S., & Ghamry, A. E. (2021). A comparative study among new hybrid root finding algorithms and traditional methods. Mathematics, 9(11), 1306.
[9]. Srinivasarao, Thota A Blended Root-Finding Algorithm for Solving Transcendental Equations SNAPP, International Journal of Applied and Computational Mathematics, https://doi.org/10.21203/rs.3.rs-2956091/v1, May 2023, pp.1-9
[10]. Occam'Razor: https://conceptually.org/concepts/occams-razor
[11]. John H. Mathews and Kurtis K. Fink Numerical Methods Using Matlab, 4th Edition, 2004 ISBN: 0-13-065248-2 Prentice-Hall Inc. Upper Saddle River, New Jersey, USA
[12]. Wikipedia https://en.wikipedia.org/wiki/False_position_method#Two_historical_types accessed December 2023
[13]. Douglas Wilhelm Harder, https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/10RootFinding/falseposition/, accessed Jun 2019.
[14]. Ehiwario, J.C., Aghamie, S.O.; Comparative Study of Bisection, Newton-Raphson and Secant Methods of Root- Finding Problems; IOSR Journal of Engineering (IOSRJEN) www.iosrjen.org; ISSN (e): 2250-3021, ISSN (p): 2278-8719; Vol. 04, Issue 04 (April. 2014), ||V1|| PP 01-07.
[15]. Tanistha Nayak ,Tirtharaj Dash Solution To Quadratic Equation Using Genetic Algorithm , Proceedings Of National Conference On AIRES-2012, Andhra University Https://Arxiv.Org/Pdf/1306.4622
[16]. Srivastava, R.B and Srivastava, S (2011), Comparison of Numerical Rate of Convergence of Bisection, Newton and Secant Methods. Journal of Chemical, Biological and Physical Sciences. Vol 2(1) pp 472-479, 2011.