Scientific Programming
Winter semester 2022-23

Project
based on Scientific Paper
"Finding Roots of Non-Linear Equations"

Comparing Different Methods and Tools

(Newton's method, Regula Falsi, Secant)

Student name	***Rafay Ahmed***
Student ID	***32077***
Student E-mail	***32077@students.hsrw***

Contents

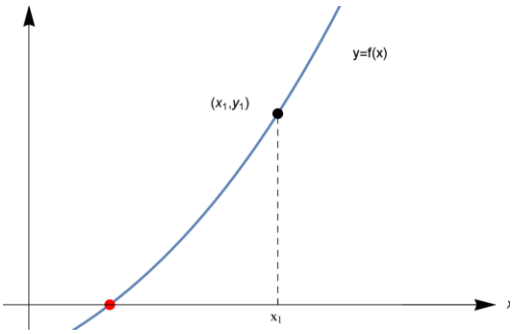
Table of Figures

Abstract

Mathematical approaches have been found so feasible for human to prosper in today's world, to connect with and to study many other various aspects. Likewise, finding roots of non-linear equations has been so obliged to solve problems of mechanics, chemicals, aerospace, astronomy and so on. In this paper, a decent research and python simulation is presented to compare different method leads finding, roots of non-linear equations. The comparison of several perspectives is based on the history behind it, the features that respective approach has, some method has pros and cons. A python code is also presented in Jupyter notebook to describe how it works. (Keywords: Non-Linear equation, Newton-Raphson method, Taylor series, Bracketing, Non-Bracketing, Regula Falsi method, Bisection method, Pros & Cons).

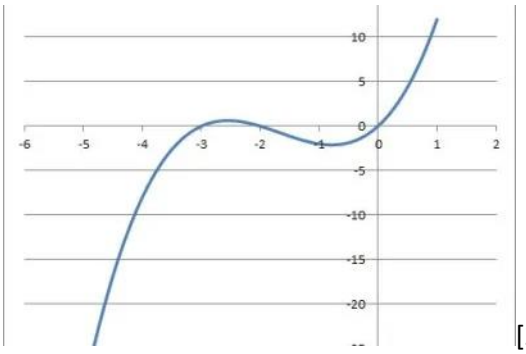
Introduction

Before starting the actual representation, it would be good to elaborate what is 'Finding roots of non-linear equation'. So, the fundamental to know is non-linear system, it is the kind of system in which function or equation has the lowest exponent of more than one (variable having exponent 1 makes it linear system). In more defined words, we can probably say that those functions where output are not dependent on their input and therefore, the graphical behaviour of non-linear system leads to the results that cannot be expected. Because the graphical representation of non-linear equation is unlike linear system (straight line). Non-linear system can never form a straight-line graph. There are plenty of non-linear equations, here are some following i: e. Quadratic equation, exponential equations, Trigonometric equation and many more. The roots of the equation are the solution of the equation when the equation is equated to zero "0 ". Or in easy way we can say whenever the non-linear system crosses the x-axis, that coordinate is called the root of equation. But moreover, with the passage of time, scientist and mathematicians had investigated different methods to find the roots of equation and phenomena behind it. The most popular are Newton method, Regula Falsi, secant method, Bisection method, Wolfram alpha, maxima etc. Conceptually, almost every method is based on iteration but has some distinction.



[1]

Figure 1: red point here denotes the root of the curve



[2]

Figure 2: Graph representing quadratic equation having solutions x=0, x=-2, x=-3

Types of Non-linear Equation

Non-linear system is so diversified, as trigonometric equation, quadratic equations, exponential equation, logarithmic equation. And of course, every type of non-linear equation has its own characteristics like trigonometric equations has trigonometry functions, exponential with exponential function and other with their respective functions. However, the method to find the roots are almost the same for all non-linear equations regardless of the approach followed.

Example of trigonometric function.

$$2\sin^2 x + \sin x - 1 = 0$$
$$(2\sin x - 1)(\sin x + 1) = 0 \quad \text{(factor)}$$
$$2\sin x - 1 = 0 \quad \text{or} \quad \sin x + 1 = 0$$
$$\sin x = \frac{1}{2} \qquad \sin x = -1$$
$$x = \frac{\pi}{6} \text{ or } \frac{5\pi}{6} \qquad x = \frac{3\pi}{2}$$
$$x = 30^\circ,\ 150^\circ \qquad x = 270^\circ$$

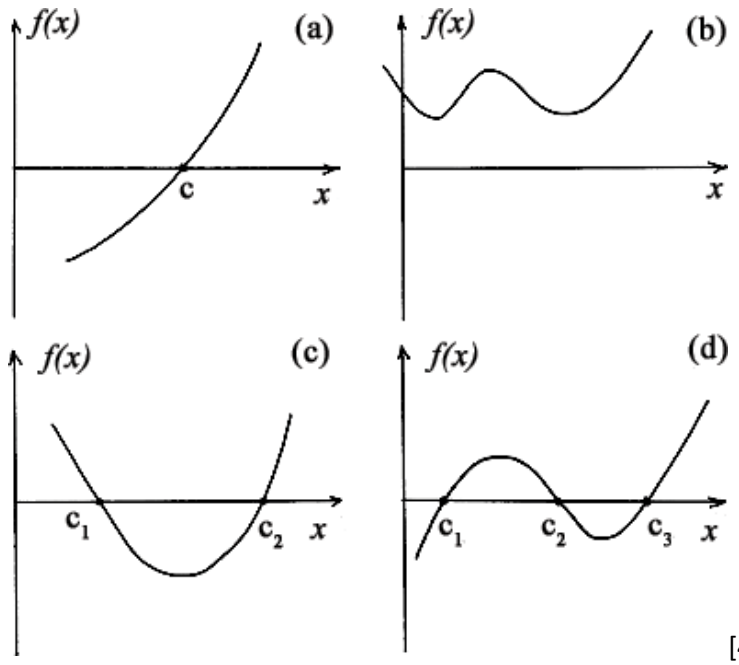
[3]

Figure 3: example of non-linear trigonometric function

Example of Exponential function

$$2^{2x} - 2^x - 6 = 0$$

$$3^{2x} - 5.3^x + 4 = 0$$



[4]

Figure 4: Graphs representing roots of non-linear curves (a) one root, (b) no real root but may be complex root exists, (c) two roots (d) three roots

Newton's Method

Newton's method of finding roots of non-linear equations is also known as Newton-Raphson method. This method was named newton's method after a mathematical work done by Isaac Newton in 1669.The main concept of this root finding method is making repetitive steps to land on an actual solution. Newton emphasized on getting a root of a function by assuming the term near a root would lie and if the correct root is not obtained so the range recorrects to the near term ignoring the previous higher terms and this is how it reaches the actual root of the non-linear equation after some repetitions. "However, this method was only applied by Newton only for polynomials and was performed for both numerical and algebraic problems which led to forming a Taylor series (the Taylor series of a function is an infinite sum of terms that are expressed in terms of the function's derivatives at a single point. Taylor polynomials are approximations of a function, which become generally better as n increases)". In 1685,

after this method was published, Joseph Raphson republished the more comprehensive narration of Newton's method in 1690 afterwards, this was named Newton-Raphson method.[5]

Characteristics of Newton's Method

This method has some distinctive features unlike other method, it has no bindings of the brackets or is said to be an open bracket typed. "Bracket type method are those, where two initial guesses are inserted in order to bind a root in a range". It follows an iteration purpose to get to the root, solution is up to concrete accuracy. As it is not bounded by bracket, so it has only one initial guess. Newton method is an approach of tangent line which means a result can never be obtained without tangent. Also, it follows a Taylor series concept so a function must expand in that form. Furthermore, this method utilizes the differentiation of a defined function to move on iteration and conclude the approximate root, f'(x) of a function f(x).[4],[6]

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + (x - x_0)^2 \frac{f''(x_0)}{2!} + \ldots$$ [4]

Figure 5: Taylor series

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$ [4]

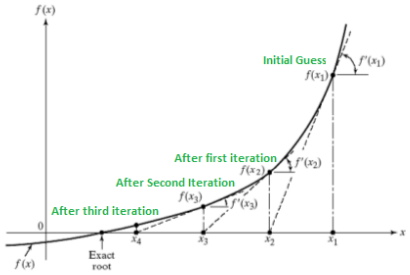
Figure 6: Newton method formula

And the process repeated as

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)}$$

The variables are as follow.

- *$x_0 = x_0$ is the initial value,*
- *x = x is the solution of the function f(x),*
- *f(x) = f(x) is the non-linear function,*
- *f'(x) = f'(x) is the derivative of function f(x)*



[7]

Figure 7: How iteration works in Newton' method

Python program Implementing Newton's Method

The solution of quadratic equation is concluded below i.e., the function is

$X^3 + 2x^2 + 10x - 20.$

```
3]: ''' approximate solution of f(x)=0 by using Newton method
parameters
--------
f : function, function for which we are searching for f(x)=0.
f_der : derivative of function f.
x : starting guess for a function f.
z : decimal places in the result.
x_z : increment in x.
i : number of repititon.
g : definig type of g.
m : defines type of z.
d : final answer , i;e. required root.


'''



import matplotlib.pyplot as plt #'matplot.lib' is library in python use for graphical representations and creative purposes
import numpy as np #'numpy' is a library essentially use for arrays in python

from sympy import *  #'sympy' is a mathematics symbol library use in python
x = Symbol("x")
f = x**3+2*x**2+10*x-20
f_der = f.diff(x) #'f_der' it is the derivative of function f
f = lambdify(x,f)
f_der = lambdify(x,f_der)
x = input("value of x : ")
z = input("How many decimal number : ")
x = float(x)
z = int(z)

i = 1
condition = True
while condition:
    g = str(x)
    x_z = x - (f(x)/f_der(x)) # Newton method formula
    print("i = ",i,">>>","x = ",x_z,"f(x) = ",f(x))

    m = str(x_z)

    if m[0:z+1]==g[0:z+1]:
        condition = False
    else:
        condition = True
        x = x_z
        i = i + 1
```

Figure 8: Implementation of Newton's method in python using Jupyter Notebook (image 1 of 4)

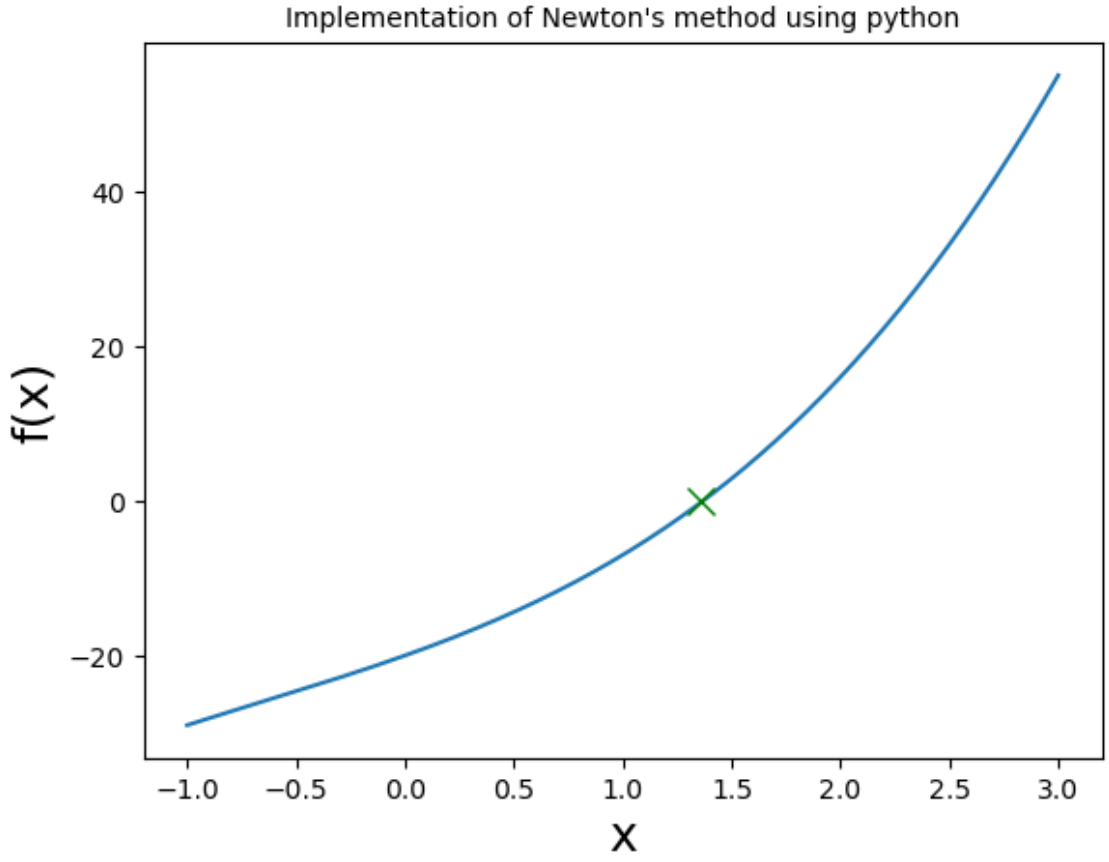
```
x = str(x)
d = x[0:z+1]       # it will give the answer to the 1. decimal places
print('Required root is ',d)
print('Solution concluded after :',i, 'iterations')

m = np.linspace(-1,3,500)
y = f(m)
plt.plot(m,y)
plt.xlabel("x", fontsize=18)
plt.ylabel("f(x)", fontsize=18)
plt.title("Implementation of Newton's method using python", fontsize=10)
d = float(d)
plt.plot(d,0, marker = 'x',markersize = '10', color = 'green')
```

Figure 9: Implementation of Newton's method in python using Jupyter Notebook (image 2 of 4)

```
value of x : 5
How many decimal number : 3
i =  1 >>> x =  3.0476190476190474 f(x) =  205.0
i =  2 >>> x =  1.9016986154795341 f(x) =  57.35838462369074
i =  3 >>> x =  1.4403816114870973 f(x) =  13.127313859752178
i =  4 >>> x =  1.3702642909248228 f(x) =  1.5415730461958717
i =  5 >>> x =  1.3688087213812625 f(x) =  0.03073279322673983
i =  6 >>> x =  1.3688081078214815 f(x) =  1.2943747229599012e-05
Required root is  1.36
Solution concluded after : 6 iterations
```

Figure 10: Results achieved through newton's method (image 3 of 4)

```
: [<matplotlib.lines.Line2D at 0x2108cb56610>]
```



[8]

Figure 11: Curve presenting the newton's method converges at 1.36 (image 4 of 4)

As Newton's method is implemented so the advantages and disadvantages could be discussed now.

Advantages of Newton's method

Newton's method has the most instant convergence rate for quadratic equation, which means that it meets the root of the equation at a faster rate compared to other method which eventually, a solution is obtained in fewer steps than other methods. Also, in newton's method the chance of error is minimized because we have seen there is no insertion of range, only one guess requirement makes it simple. The formula is easy to implement considering function and its derivative which is a built-in feature of the formula.[9]

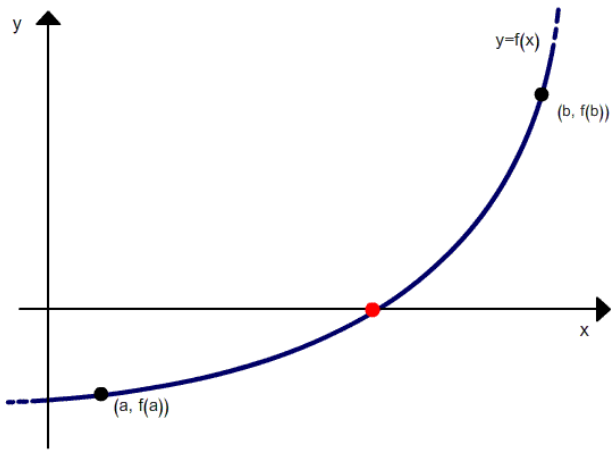
Disadvantages of Newton's method

The very common interruption in newton's method is that it does not always converge. It is said that convergence is not guaranteed. There are couple of reasons which proves this previous statement true are follows; Newton method has an obligation of derivations in denominator so if the function is given by complex formula so it would not be easy to conclude the answer. Furthermore, if the function is not always differentiable in an iterative step so the formula would not make any progress. Also, if the denominator means derivative in Newton's method becomes zero so the solution is not possible because zero denominator denotes infinity. Moreover, this method may collapse if the assumed value is far away from the actual root or in this case, it produces a very slow result.[10]

More methods for finding roots of non-linear equations are discussed below.

Regula Falsi Method

It is an old ancient method of finding roots, was founded by an Arab mathematician from Middle east. Regula Falsi method is also popularly known as "False position" method because we assume incorrect values to get the appropriate root by arranging the outcomes. There exists a method for finding roots, Bisection Method, so Regula falsi method is considered as the amended method of that because Bisection method is so slow to be implemented. However, it is a bit different from newton's method in terms of bounding by range of values. As it is a closed bracket method which means it requires an interval of values within a root lies. [11]



[9]

Figure 12: How the iteration works recorrecting the outcome of the previous iteration in Regula falsi method

Characteristics of Regula Falsi Method

Regula Falsi method calculates the new iterated result based on x-intercept of the line segment connecting to the tail points (bracketing intervals) of the function. Eventually, the solution is declared by replacing the actual function by a line segment on the bracketing interval and then it follows the standard double false position formula. [12]

This concept is also interpolated as interval valued system. The initial guesses are two. To solve non-linear equation with Regula Falsi method, the equation must be a continuous function, if a function f(x) is continuous in an interval of [a, b] and f(a)*f(b) < 0 such that the function has opposite signs at these two points a and b then the root must be in between a & b i.e. a < x < b, where x is the solution.[13]

And the formula followed in this method is derived like.

"After locating a bracketed value such that [x_0, x_1] where f(x_0) *f(x_1) <0 and a system must cross the x-axis at least once between this given interval.

The equation of the line segment the intervals A [x_0, f(x_0)] B [x_1, f(x_1)] is given by where A & B are two coordinates it produces a value for x_2 in a way that

$$x_2 = x_1 - \frac{(x_1 - x_0)}{f(x_1) - f(x_0)} * f(x_1)$$

Then the value of f(x_2) *f(x_0) and f(x_2) *f(x_1)

If f(x_2) *f(x_0) <0, then interval changes as x_0 becomes x_1

If f(x_2) *f(x_1) <0, then interval changes and x_2 becomes x_1

And again, in similar way it follows same steps until f(x_n)=0

So, the evaluated formula is:

$$x_{n+1} = x_0 - \frac{(x_n - x_1)}{f(x_n) - f(x_0)} * f(x_0)$$

"[14]

Here, value of X_{n+1} in the given function to obtain f(X) = 0

Variables are defined i.e.

X_0 = lower value of interval

X_n = upper value of interval

$f(x_0)$ = value of function at x_0

f (x_n) = value of function at x_n

Python Implementation of Regula Falsi Method

The solution of quadratic equation is concluded below i.e., the function is

$X^3 + 2x^2 + 10x - 20$.

```
''' finding root of function f(x)=0 using regula falsi method also known as false position
parameters
----

x1 : root 1
x2 : root 2
n = number of decimal place in answer
x : root required
g : type of x
m : type of x
y : funtion of f(m) is defined

'''


import matplotlib.pyplot as plt
import numpy as np

def f(x):
    return x**3+2*x**2+10*x-20 #Leonardo's Equation


x1 = input('x1 = ')
x2 = input('x2 = ')
n = input('decimal places = ')
x1 = float(x1)
x2 = float(x2)
n = int(n)

def regula(x1,x2):
    x = 0
    i = 1
    condition = True
    while condition:
        g = str(x)
        x = x1-((x2-x1)/(f(x2)-f(x1)))*f(x1)
        if f(x)<0:
            x1 = x
        else:
            x2 = x
        print('i=',i,'>>>','x=',x,'f(x)=',f(x))

        m = str(x)  #here 'm' stores the new value of x
        if m[0:n+3]==g[0:n+3]:
            condition = False # This loop will work until the value is false
        else:
            condition = True
            i = i+1
```

Figure 13: Regula Falsi method Python representation on Jupyter Notebook (image 1 of 4)

```
        x = str(x)
        d = x[0:n+3]
        print('Required root',d)
        print('Solution concluded after :',i, 'iterations')


        m = np.linspace(-1,3,100) #'-1' is the starting point, '3 is the ending point'
        y = f(m)
        plt.plot(m,y)
        d = float(d)
        plt.plot(d,0, marker = 'x', color = 'green')
        plt.title('Python Implementation of Regula Falsi method')
        plt.xlabel("(x)", fontsize=18)
        plt.ylabel("f(x)", fontsize=18)



if f(x1)*f(x2)>0:
    print('Inappropriate values')
    print('Try again')
else:
    regula(x1,x2)
```

Figure 14: Regula Falsi method Python representation on Jupyter Notebook (image 2 of 4)

```
x1 = 1
x2 = 2
decimal places = 3
i= 1 >>> x= 1.3043478260869565 f(x)= -1.3347579518369344
Required root 1.3043
Solution concluded after : 2 iterations
i= 2 >>> x= 1.3579123046578667 f(x)= -0.22913572958733397
Required root 1.3579
Solution concluded after : 3 iterations
i= 3 >>> x= 1.3669778048165133 f(x)= -0.03859187677837639
Required root 1.3669
Solution concluded after : 4 iterations
i= 4 >>> x= 1.3685009755999702 f(x)= -0.006478728147058632
Required root 1.3685
Solution concluded after : 5 iterations
i= 5 >>> x= 1.368756579007422 f(x)= -0.001087042825339779
Required root 1.3687
Solution concluded after : 6 iterations
i= 6 >>> x= 1.3687994628833735 f(x)= -0.00018237436024648446
Required root 1.3687
Solution concluded after : 6 iterations
```

Figure 15: Results of the Regula Falsi method in Python (image 3 of 4)

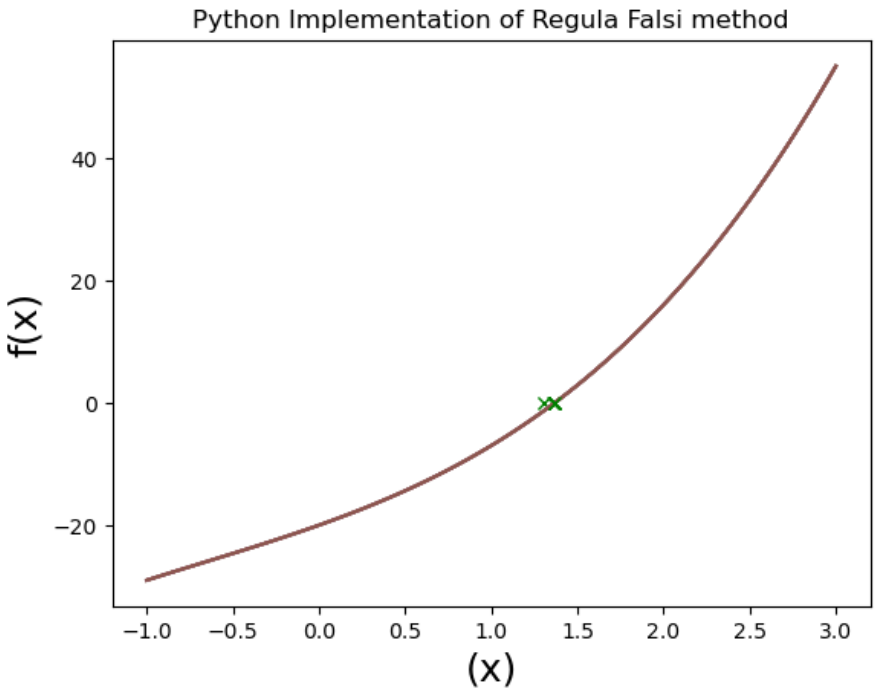


Figure 16: Graphical representation of Regula Falsi method (image 4 of 4)

[15]

As earlier the Pros and Cons of Newton's Method were mentioned so Regula Falsi also has some as follows:

Pros of Regula Falsi Method

If Regula Falsi method is compared to Newton's method so unlike that, Regula Falsi always converges. As there is no possibility of getting zero denominator so the possibility of getting infinite solution is no more so eventually it comes to some finite limits in an interval. It follows a linear rate of convergence and does have a steady rate between values. Its comparison to other approaches exceptional to Newton's method, has a fast rate i.e., Bisection method. As it was implemented in python, so it has decent approach to apply. "No concept of derivation." [16]

Cons of Regula Falsi Method

Having said that Regula Falsi method have some drawbacks, firstly its convergences at a slow rate even though, the convergence is guaranteed. It fails to conclude a result of complex roots as it cannot determine the complex roots. Also not be able to find roots of equations like f(x)= $x_{2,}$ as this method is based on bracketing terms so it's not possible to bracket single terms with square. Lastly, it cannot be

useful when the interval does not have an opposite sign, because it does not make the product of function less than zero. f(a)*f(b)<0 where (a, b) are the roots. [18], [17]

Conclusion

As in this paper, two most widely studied approaches are discussed, other than these there are few more i.e., Bisection method, Secant method etc. Bisection and Secant method have more or less similar concept compared to Regula Falsi but still they work fairly. Apparently, in this writing, it could be clearly seen that how Newton method is differed from the others. However, the advantages and dis. Advantages describes that Newton's method is more effective even though it has some drawbacks but still it's more widely recognized. The question is what we achieve with these methods, we study many diverse aspects of engineering considering these methods and these approaches for example it helps in measuring waves, parabola, different energy, and sound graphs as they all include non-linear curves and you name it, these method does that. So, it helps making our daily lives more advance and allow us to prosper in various theories and inventions.

References

[1] https://amsi.org.au/ESA_Senior_Years/SeniorTopic3/3j/3j_2content_2.html (Image) [accessed 21 Dec 2022]

[2] https://jdmeducational.com/nonlinear-equations-4-types-you-should-know-plus-how-to-solve/ (Image) [accessed 21 Dec 2022]

[3] https://mathbitsnotebook.com/Algebra2/TrigConcepts/TCEquationsMore.html (Image) [accessed 21 Dec 2022]

[4] Dr. Ian Balitsky, Physics Prof. at Old Dominion University (balitsky.com physics: 420), https://balitsky.com/teaching/phys420/Nm4_roots.pdf [23 Dec 2022]

[5] https://en.wikipedia.org/wiki/Newton%27s_method#Description [Multiple accessed between 20 Dec 2022- 2 Jan 2023]

[6] https://www.codesansar.com/numerical-methods/newton-raphson-method-features.htm (online teaching platform, codesansar.com) [Multiple accessed between 20- Dec 2022- 5 Jan 2023]

[7] https://www.geeksforgeeks.org/program-for-newton-raphson-method/ (Image) [24- Dec 2022]

[8] (Uma p BalaRaju, Assistant Professor In department of Electrical Engineering at Bhilal Institute Of technology Durg, Chhattisgarh, channel Techno corps), https://www.youtube.com/watch?v=Wzv00lCAOqc [26 - Dec 2022]

[9] https://www.codesansar.com/numerical-methods/advantages-newton-raphson-method.htm (online teaching platform, codeansar.com) [multiple accessed between 20-Dec 2022 – 5- Jan 2023]

[10] McMaster University >> cas.mcmaster.ca, http://www.cas.mcmaster.ca/~cs4te3/notes/newtons_method.pdf [accessed 24 Dec 2022]

[11] (Eklavya Chopra, author at iq.opengenus.org), https://iq.opengenus.org/regula-falsi-method/

[12] https://en.wikipedia.org/wiki/Regula_falsi

[13] Mujeeb Ur Rehman, Chemical engineer, https://www.slideshare.net/MujeebURRahman38/naca-regula-falsi-method

[14] Debashis Roy, from University of Calcutta, https://mycareerwise.com/programming/category/numerical-analysis/regula-falsi-method

*[15] (*Uma p BalaRaju, Assistant Professor In department of Electrical Engineering at Bhilal Institute Of technology Durg, Chhattisgarh, channel Techno corps), *https://www.youtube.com/watch?v=yoN7Y1eHRHM&t=912s*

[16] https://www.codesansar.com/numerical-methods/false-position-advantages.htm

[17] (Presented by: Mujeeb Rahman supervision of Sher Khan Awan slide no: 17), https://www.slideshare.net/MujeebURRahman38/naca-regula-falsi-method

[18] https://www.codesansar.com/numerical-methods/false-position-disadvantages.htm

Attention! [Accessed dates could have been flexible as URLs have been visited multiple times, hope this would not be trouble]

Thank you!