International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 5 Issue 2|| February 2016 || PP.01-10

Numerical Study of Some Iterative Methods for Solving Nonlinear Equations

Azizul Hasan
Department of Mathematics, Faculty of Science, Jazan University Jazan, KSA.

Abstract: *In this paper we introduce, numerical study of some iterative methods for solving non linear equations. Many iterative methods for solving algebraic and transcendental equations is presented by the different formulae. Using bisection method , secant method and the Newton's iterative method and their results are compared. The software, matlab 2009a was used to find the root of the function for the interval [0,1]. Numerical rate of convergence of root has been found in each calculation. It was observed that the Bisection method converges at the 47 iteration while Newton and Secant methods converge to the exact root of* ***0.36042170296032*** *with error level* 10^{-14} *at the 4th and 5th iteration respectively. It was also observed that the Newton method required less number of iteration in comparison to that of secant method. However, when we compare performance, we must compare both cost and speed of convergence [6]. It was then concluded that of the three methods considered, Secant method is the most effective scheme. By the use of numerical experiments to show that secant method are more efficient than others.*

Keywords: *roots, rate of convergence, Iterative methods, Algorithm, Transcendental equations, Numerical experiments, Efficiency Index.*

I. Introduction

Solving non linear equations is one of the most important and challenging problems in science and engineering applications. The root finding problem is one of the most relevant computational problems. It arises in a wide variety of practical applications in Physics, Chemistry, Biosciences, Engineering, etc. As a matter of fact, the determination of any unknown appearing implicitly in scientific or engineering formulas, gives rise to root finding problem [1]. Relevant situations in Physics where such problems are needed to be solved include finding the equilibrium position of an object, potential surface of a field and quantized energy level of confined structure [7]. The common root-finding methods include: Bisection, Newton-Raphson, False position, Secant methods etc. Different methods converge to the root at different rates. That is, some methods are faster in converging to the root than others. The rate of convergence could be linear, quadratic or otherwise. The higher the order, the faster the method converges [3]. The study is at comparing the rate of performance (convergence) of Bisection, Newton-Raphson and Secant as methods of root-finding. Obviously, Newton-Raphson method may converge faster than any other method but when we compare performance, it is needful to consider both cost and speed of convergence. An algorithm that converges quickly but takes a few seconds per iteration may take more time overall than an algorithm that converges more slowly, but takes only a few milliseconds per iteration [8]. For the purpose of this general analysis, we may assume that the cost of an iteration is dominated by the evaluation of the function - this is likely the case in practice. So, the number of function evaluations per iteration is likely a good measure of cost [8]

Secant method requires only one function evaluation per iteration, since the value of function $f(x_{n-1})$ can be stored from the previous iteration [6]. Newton's method, on the other hand, requires one function evaluation and the one derivative evaluation per iteration. It is often difficult to estimate the cost of evaluating the derivative in general (if it is possible) [1, 4-5]. It seem safe, to assume that in most cases, evaluating the derivative is at least as costly as evaluating the function [8]. Thus, we can estimate that the Newton iteration takes about two functions evaluation per iteration. This disparity in cost means that we can run two iterations of the secant method in the same time it will take to run one iteration of Newton method. In comparing the rate of convergence of Bisection, Newton and Secant methods,[8] used C++programming language to calculate the cube roots of numbers from 1 to 25, using the three methods. They observed that the rate of convergence is in the following order: Bisection method < Newton method < Secant method. They concluded that Newton method is 7.678622465 times better than the Bisection method while Secant method is 1.389482397 times better than the Newton method. For Solving nonlinear equations Newton's method is one of the most pre dominant problems in numerical analysis [1]. Some historical points on this method can α be found in [13,14].

Definition and Notation: Let $\alpha \in R$ and $x_N \in R, N = 0,1,2,3, \ldots\ldots\ldots$ Then the sequence x_N is said to be convergence to α if $\lim_{N\to\infty} |x_N - \alpha| = 0$. If there exists a constant $c > 0$, an integer $N_0 \geq 0$ and $p \geq 0$ such that for all $N > N_0$

We have $$|x_{N+1} - \alpha| \leq c|x_N - \alpha|^p$$

Then x_N is said to be converges to α with convergence order at least p. If $p = 2$, the convergence is to be quadratic or if $p = 3$ then it is cubic.

Notation: The notation $e_n = x_n - \alpha$, is the error in the n^{th} iteration.

The equation $e_{n+1} = ce_n^p + O(en^{p+1})$ is called the error equation. By substituting $e_n = x_n - \alpha$ for all n in any iterative method and simplifying. We obtain the error equation for that method. The value of p obtained is called the order of this method.

Intermediate Value Theorem: If f is continuous in Closed interval [a, b] and K is any number between f(a) and f(b), then there exists a number c in (a; b) such that f(c) = K. In particular, if f(a) and f(b) are opposite signs, then there exists a number c in (a; b) such that f(c) = 0.

II. Material and Methods:

Bisection-Method: As the title suggests, the method is based on repeated bisections of an interval containing the root. The basic idea is very simple. Suppose f(x) = 0 is known to have a real root x = α in an interval [a, b].

• Then bisect the interval [a, b], and let $c = \frac{a+b}{2}$ be the middle point of [a, b]. If c is the root, then we are done. Otherwise, one of the intervals [a, c] or [c, b] will contain the root.

• Find the one that contains the root and bisect that interval again.

• Continue the process of bisections until the root is trapped in an interval as small as warranted by the desired accuracy.

To implement the above idea, we must know in each iteration: which of the two intervals contain the root of f(x) = 0.

Algorithm: Inputs: f(x) - The given function.

Chose (a_0, b_0) - The two numbers such that $f(a_0)f(b_0) < 0$.

Output: An approximation of the root of f(x) = 0 in [a_0, b_0].

For k = 0, 1, 2, . . ., do until satisfied:

• Compute $c_k = \frac{a_k + b_k}{2}$

• Test, using one of the criteria stated in the next section, if c_k is the desired root. If so, stop.

• If c_k is not the desired root, test if $f(c_k)f(a_k) < 0$. If so, set $b_{k+1} = c_k$ and $a_{k+1} = a_k$.

Otherwise, set $a_{k+1} = c_k$, $b_{k+1} = b_k$.

End.

Number of Iterations Needed in the Bisection Method to Achieve Certain Accuracy:

Let us now find out what is the minimum number of iterations N needed with the bisection method to achieve a certain desired accuracy. Let the interval length after N iterations is $\frac{b_0 - a_0}{2^N}$

So, to obtain an accuracy of ϵ or ϵ be the tolerance we must have $\frac{b_0 - a_0}{2^N} \leq \epsilon$

$$\text{That is } 2^{-N}(b_0 - a_0) \leq \epsilon$$

$$2^N \geq \frac{b_0 - a_0}{\epsilon}$$

$$N\, ln2 \geq \ln(b_0 - a_0) - ln\, \epsilon \quad \text{(Taking the natural log on both sides.)}$$

$$N \geq \frac{\ln(b_0 - a_0) - ln\epsilon}{\ln 2}$$

$$N \geq \frac{\log_{10}(b_0 - a_0) - \log_{10}\epsilon}{\log_{10} 2}$$

Theorem: The number of iterations N needed in the bisection method to obtain an accuracy of ϵ is given by

$$N \geq \frac{\log_{10}(b_0 - a_0) - \log_{10}\epsilon}{\log_{10} 2}$$

Remarks: (i) Since the number of iterations N needed to achieve a certain accuracy depends upon the initial length of the interval containing the root, it is desirable to choose the initial interval [a_0, b_0] as small as possible.

NEWTON-RAPHSON METHOD: We consider the problem of numerical determine a real root α of non linear equation $$f(x) = 0, \quad f: D \subset R \to R$$

(1.1)

The Newton-Raphson method finds the slope (tangent line) of the function at the current point and uses the zero of the tangent line as the next reference point. The process is repeated until the root is found [10-12]. The method is probably the most popular technique for solving nonlinear equation because of its quadratic convergence rate. But it is sometimes damped if bad initial guesses are used [9-11].It was suggested however, that Newton's method should sometimes be started with Picard iteration to improve the initial guess [9]. Newton Raphson method is much more efficient than the Bisection method. However, it requires the calculation of the derivative of a function as the reference point which is not always easy or either the derivative does not exist at all or it cannot be expressed in terms of elementary function [6, 7]. Furthermore, the tangent line often shoots wildly and might occasionally be trapped in a loop [6]. Once you have x_k the next approximation x_{k+1} is determined by treating f(x) as a linear function at x_k.

Another way of looking at the same fact: We can rewrite f(x) = 0,

$$\rightarrow \quad b(x)f(x) = 0, \qquad \rightarrow \quad x = x - b(x)f(x) = g(x)$$

Here the function b(x) is chosen in such a way, to get fastest possible convergence. We have

$$\rightarrow g'(x) = 1 - b'(x)\, f(x) - b(x)\, f'(x)$$

Let r be the root, such that f(r) = 0, and r = g(r). We have

$$\rightarrow \quad g'(r) = 1 - b(r)\, f'(r) \qquad \text{smallest possible:} \quad |g'(r)| = 0$$

Choose now

$$\rightarrow \quad 1 - b(x)f'(x) = 0, \qquad \rightarrow \quad b(x) = 1/f'(x)$$

We get a fixed point iteration for $x = g(x) = x - f(x)/f'(x)$

In a sense, Newton's iteration is the "best" fixed point iteration!

The known numerical method for solving non linear equations is the Newton's method is given by

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(k)}, \qquad k = 0, 1, 2, 3, \ldots\ldots\ldots \qquad (1.2)$$

where x_0 is an initial approximation sufficiently near to α. The convergence order of the Newton's method is quadratic for simple roots [4]. By implication, the quadratic convergence we mean that the accuracy gets doubled at each iteration.

Algorithm of the Newton- Raphson Method:

Inputs: f(x) –the given function, xo –the initial approximation, ε-the error tolerance and N –the maximum number of iteration.

Output: An approximation to the root $x = \alpha$ or a message of a failure. Assumption: $x = \alpha$ is a simple root of $f(x)=0$

- Compute f(x) , and $f'(x)$
- Compute $x_{k+1} = x_k - \frac{f(x_k)}{f'(k)}, \quad k = 0, 1, 2, 3, \ldots\ldots\ldots$ do until convergence or failure. Test for convergence of failure If $|f(x_{k+1})| < \varepsilon, \quad |x_{k+1} - x_k|/x_k < \varepsilon$ or k>N, stop.

End.

It was remarked in [6], that if none of the above criteria has been satisfied, within a predetermined, say, N, iteration, then the method has failed after the prescribed number of iteration. In this case, one could try the method again with a different x_0 . Meanwhile, a judicious choice of x_0 can sometimes be obtained by drawing the graph of *f(x)*, if possible. However, there does not seems to exist a clear- cut guideline on how to choose a right starting point, x_0 that guarantees the convergence of the Newton-Raphson method to a desire root. Newton method just needs one, namely, x_0.

Order of convergence: Let $x_0, x_1, x_2, \ldots\ldots x_n$ be a sequence of approximations to a root α produced by a numerical method, where $\lim_{k\to\infty} |x_k - \alpha| = 0$. Let $\epsilon_k = x_k - \alpha$ if $\lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|^p} = c$

for some p and some non-zero constant C, then the method has order of convergence p, and C is the asymptotic error constant. (The bigger p is the faster the convergence).

If we now consider Newton's method, where f(x) is assumed to be twice differentiable, and α is a simple root, then substituting $\epsilon_k + \alpha = x_k$ in (1) and Expanding $f(\alpha + \epsilon_k)$ and $f'(\alpha + \epsilon_{k-1})$ in taylor's series about the point α and noting that $f(\alpha) = 0$ we obtain

$$\epsilon_{k+1} = \epsilon_k - \frac{\left[\epsilon_k f'(\alpha) + \frac{1}{2}\epsilon_k{}^2 f''(\alpha) + \cdots\right]}{f'(\alpha) + \epsilon_k\, f''(\alpha) + \cdots}$$

$$\epsilon_{k+1} = \epsilon_k - \left[\epsilon_k + \frac{1}{2}\epsilon_k{}^2 \frac{f''(\alpha)}{f'(\alpha)} + \cdots\right]\left[1 + \epsilon_k \frac{f''(\alpha)}{f'(\alpha)} + \cdots\right]^{-1}$$

$$\epsilon_{k+1} = \frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}\epsilon_k{}^2 + O(\epsilon_k{}^3)$$

On neglecting $\epsilon_k{}^3$, and higher powers of ϵ_k , we get

$$\epsilon_{k+1} = C\epsilon_k{}^2 \tag{1.3}$$

Where $C = \frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}$, and taking $\rightarrow \lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|^2} = \frac{|f''(\alpha)|}{2|f'(\alpha)|}$

If $x_k \to \alpha$ as $k \to \infty$

Thus, in the case of a simple root α, Newton's method has second order— quadratic convergence, with asymptotic error constant $\frac{|f''(\alpha)|}{2|f'(\alpha)|}$ .Thus the subsequent error ϵ_{k+1} at each step is proportional to the square of the previous error ϵ_k

Secant Method: A major disadvantage of the Newton Method is the requirement of finding the value of the derivative of f(x) at each approximation. There are some functions for which this job is either extremely difficult (if not impossible) or time consuming. A way out is to approximate the derivative by knowing the values of the function at that and the previous approximation according to [6]. Or the derivative can be approximated by finite divided difference. The secant method may be regarded as an approximation to Newton's method where, instead of $f'(x_k)$, the quotient $\frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$ is used; i.e. instead of the *tangent* to the graph of f(x) at x_k , we use the *secant* joining the points $(x_{k-1}, f(x_{k-1}))$ and $(x_k, f(x_k))$. If we equate the two expressions for the slope of the secant:

$$\frac{f(x_k)}{x_k - x_{k+1}} = \frac{f(x_k)-f(x_{k-1})}{x_k - x_{k-1}}$$

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k)-f(x_{k-1})} \tag{1.4}$$

This is secant method. The advantage of the method is that f'(x) need not be evaluated; so the number of function evaluations is half that of Newton's method. However, its convergence rate is slightly less than Newton and it requires two initial guesses x_0 and x_1.

Algorithm: Inputs: f(x) - The given function

x_0, x_1 - The two initial approximations of the root

ε - The error tolerance

N - The maximum number of iterations

Output: An approximation of the exact solution δ or a message of failure.

For k = 1, 2, · · ·, until the stopping criteria is met,

- Compute $f(x_k)$ and $f(x_{k-1})$
- Compute the next approximation: $x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k)-f(x_{k-1})}$
- Test for convergence or maximum number of iterations: If $|x_{k+1} - x_k| < \varepsilon$ or if k > N, Stop

End

Convergence analysis: Let x_k be the sequence of approximation and also assume that α is a simple root of f(x)=0, substituting $\epsilon_k = x_k - \alpha$, $\epsilon_{k-1} = x_{k-1} - \alpha$, $\epsilon_{k+1} = x_{k+1} - \alpha$ in equation (1.4) we obtain

$$\alpha + \epsilon_{k+1} = \alpha + \epsilon_k - \frac{(\epsilon_k - \epsilon_{k-1})\, f(\alpha+\epsilon_k)}{f(\alpha+\epsilon_k) - f(\alpha+\epsilon_{k-1})} \tag{1.5}$$

Expanding $f(\alpha + \epsilon_k)$ and $f(\alpha + \epsilon_{k-1})$ in taylor's series about the point α and noting that $f(\alpha) = 0$ we get

$$\epsilon_{k+1} = \epsilon_k - \frac{(\epsilon_k - \epsilon_{k-1})\left[\epsilon_k f'(\alpha) + \frac{1}{2}\epsilon_k{}^2 f''(\alpha) + \cdots\right]}{(\epsilon_k - \epsilon_{k-1})f'(\alpha) + \frac{1}{2}(\epsilon_k{}^2 - \epsilon_{k-1}{}^2)f''(\alpha) + \cdots}$$

$$\epsilon_{k+1} = \epsilon_k - \left[\epsilon_k + \frac{1}{2}{\epsilon_k}^2 \frac{f''(\alpha)}{f'(\alpha)} + \cdots\right]\left[1 + \frac{1}{2}(\epsilon_{k-1} + \epsilon_k)\frac{f''(\alpha)}{f'(\alpha)} + \cdots\right]^{-1}$$

$$\epsilon_{k+1} = \frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}\epsilon_k\epsilon_{k-1} + O({\epsilon_k}^2\epsilon_{k-1} + \epsilon_k{\epsilon_{k-1}}^2)$$

$$\epsilon_{k+1} = C\epsilon_k\epsilon_{k-1} \tag{1.6}$$

Where $C = \frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}$ and higher powers of ϵ_k are neglected.

The relation of the form (1.6) is called the error equation .Keeping in view the definition of the rate of convergence; we seek a relation of the form

$$\epsilon_{k+1} = A{\epsilon_k}^p \tag{1.7}$$

Where A and p are to be determined. From (1.7) we have

$$\epsilon_k = A{\epsilon_{k-1}}^p \quad \text{or} \quad \epsilon_{k-1} = A^{-1/p}{\epsilon_k}^{1/p}$$

substituting the values of ϵ_{k+1} and ϵ_{k-1} in (1.6) we obtain

$${\epsilon_k}^p = CA^{-(1+1/p)}{\epsilon_k}^{1+\frac{1}{p}} \tag{1.8}$$

Comparing the powers of ϵ_k on both sides, we get

$$p = 1 + \frac{1}{p} \quad \text{which gives} \quad p = \frac{1}{2}(1 \pm \sqrt{5}\ )$$

Neglecting negative sign ,we find that the rate of convergence for the secant method is $p = 1.618$ Hence the rate of convergence is super linear.

III. Analysis of convergence of bisection, Newton and Secant Methods

Bisection method converges linearly While secant and Newton methods converges super linear and quadratics respectively. Bisection method converges slow but sure. Newton converges fast and accuracy is gets double after each iteration But there is no guarantee for convergence if the initial guess are not close to the desire root, there is no guarantee secant method converges. If f is differentiable on that interval and there is a point where f'=0 , then the algorithm may not converge [10].

IV. Numerical Experiments and Comparative discussion

In this section, we employ the various methods obtained in this paper to solve some nonlinear equations and compare them. We use the stopping criteria $|x_{k+1} - x_k| < \varepsilon$ and $|f(x_{k+1})| < \varepsilon$, where $\varepsilon = 10^{-14}$, for computer programs. All programs are written in Matlab2009a

Numerical roots for bisection method: The Bisection method for a single-variable function $f(x) = 3x + \sin(x) - e^x$ on [0,1], using the software, matlab. The results are presented in Table 1 to 3.

Table 1. Iteration data for bisection method, with a=0, b=1, $\varepsilon = 10^{-14}$

a	f(a)	b	f(b)	c	$\lvert f(c)\rvert$
0.000000000000	-1.000000000000	1.000000000000	1.123189156348	0.500000000000	0.330704267904
0.000000000000	-1.000000000000	0.500000000000	0.330704267904	0.500000000000	0.330704267904
0.250000000000	-0.286621457433	0.500000000000	0.330704267904	0.250000000000	0.286621457433
0.250000000000	-0.286621457433	0.375000000000	0.036281114467	0.375000000000	0.036281114467
0.312500000000	-0.121899426593	0.375000000000	0.036281114467	0.312500000000	0.121899426593

	342				
0.34375000000000	-0.04195596590346	0.37500000000000	0.03628111446785	0.34375000000000	0.04195596590346
0.35937500000000	-0.00261963457026	0.37500000000000	0.03628111446785	0.35937500000000	0.00261963457026
0.35937500000000	-0.00261963457026	0.36718750000000	0.01688575294724	0.36718750000000	0.01688575294724
0.35937500000000	-0.00261963457026	0.36328125000000	0.00714674162921	0.36328125000000	0.00714674162921
0.35937500000000	-0.00261963457026	0.36132812500000	0.00226696530234	0.36132812500000	0.00226696530234
0.36035156250000	-0.00017548279455	0.36132812500000	0.00226696530234	0.36035156250000	0.00017548279455
0.36035156250000	-0.00017548279455	0.36083984375000	0.00104595435169	0.36083984375000	0.00104595435169
0.36035156250000	-0.00017548279455	0.36059570312500	0.00043528903577	0.36059570312500	0.00043528903577
0.36035156250000	-0.00017548279455	0.36047363281250	0.00012991643276	0.36047363281250	0.00012991643276
0.36041259765625	-0.00002277985313	0.36047363281250	0.00012991643276	0.36041259765625	0.00002277985313
0.36041259765625	-0.00002277985313	0.36044311523438	0.00005356912179	0.36044311523438	0.00005356912179
0.36041259765625	-0.00002277985313	0.36042785644531	0.00001539484232	0.36042785644531	0.00001539484232
0.36042022705078	-0.00000369245340	0.36042785644531	0.00001539484232	0.36042022705078	0.00000369245340
0.36042022705078	-0.00000369245340	0.36042404174805	0.00000585120746	0.36042404174805	0.00000585120746
0.36042022705078	-0.00000369245340	0.36042213439941	0.00000107938028	0.36042213439941	0.00000107938028
0.36042118072510	-0.00000130653575	0.36042213439941	0.00000107938028	0.36042118072510	0.00000130653575
0.36042165756226	-0.00000011357753	0.36042213439941	0.00000107938028	0.36042165756226	0.00000011357753
0.36042165756226	-0.00000011357753	0.36042189598083	0.00000048290142	0.36042189598083	0.00000048290142
0.36042165756226	-0.00000011357753	0.36042177677155	0.00000018466196	0.36042177677155	0.00000018466196

0.36042165756226	-0.00000011357753	0.36042171716690	0.00000003554221	0.36042171716690	0.00000003554221
0.36042168736458	-0.00000003901766	0.36042171716690	0.00000003554221	0.36042168736458	0.00000003901766
0.36042170226574	-0.00000000173772	0.36042171716690	0.00000003554221	0.36042170226574	0.00000000173772
0.36042170226574	-0.00000000173772	0.36042170971632	0.00000001690225	0.36042170971632	0.00000001690225
0.36042170226574	-0.00000000173772	0.36042170599103	0.00000000758226	0.36042170599103	0.00000000758226
0.36042170226574	-0.00000000173772	0.36042170412838	0.00000000292227	0.36042170412838	0.00000000292227
0.36042170226574	-0.00000000173772	0.36042170319706	0.00000000059227	0.36042170319706	0.00000000059227
0.36042170273140	-0.00000000057272	0.36042170319706	0.00000000059227	0.36042170273140	0.00000000057272
0.36042170273140	-0.00000000057272	0.36042170296423	0.00000000000977	0.36042170296423	0.00000000000977
0.36042170284782	-0.00000000028147	0.36042170296423	0.00000000000977	0.36042170284782	0.00000000028147
0.36042170290602	-0.00000000013585	0.36042170296423	0.00000000000977	0.36042170290602	0.00000000013585
0.36042170293513	-0.00000000006304	0.36042170296423	0.00000000000977	0.36042170293513	0.00000000006304
0.36042170294968	-0.00000000002663	0.36042170296423	0.00000000000977	0.36042170294968	0.00000000002663
0.36042170295696	-0.00000000000843	0.36042170296423	0.00000000000977	0.36042170295696	0.00000000000843
0.36042170295696	-0.00000000000843	0.36042170296059	0.00000000000067	0.36042170296059	0.00000000000067
0.36042170295877	-0.00000000000388	0.36042170296059	0.00000000000067	0.36042170295877	0.00000000000388
0.36042170295968	-0.00000000000160	0.36042170296059	0.00000000000067	0.36042170295968	0.00000000000160
0.36042170296014	-0.00000000000046	0.36042170296059	0.00000000000067	0.36042170296014	0.00000000000046
0.36042170296014	-0.00000000000046	0.36042170296037	0.00000000000010	0.36042170296037	0.00000000000010
0.36042170296	-	0.36042170296	0.00000000000	0.36042170296	0.00000000000

025	0.000000000000018	037	010	025	018
0.360421702960031	-0.000000000000004	0.360421702960037	0.000000000000010	**0.360421702960031**	**0.000000000000004**
0.360421702960031	-0.000000000000004	0.360421702960034	0.000000000000003	**0.360421702960034**	**0.000000000000003**
0.360421702960032	-0.000000000000000	0.360421702960034	0.000000000000003	**0.360421702960032**	**0.000000000000000**

Table 1 shows that the iteration data obtained for bisection method .it was observed that the function converges to 0.36042170296032 at the 47 iteration with error level of 0.00000000000001

Numerical roots for secant method: The secant method for a single-variable function $f(x) = 3x + \sin(x) - e^x$ on [0, 0.5], using the matlab software.

Table 2. Iteration data for secant method with a=0, b=0.5, $\varepsilon = 10^{-14}$

x_{k-1}	$f(x_{k-1})$	x_k	$f(x_k)$	x_{k+1}	$\lvert f(x_{k+1})\rvert$
0 .000000000000	-1.00000000000000	0.50000000000000	0.33070426790407	0.37574088552938	0.03811468858563
0.50000000000000	0.33070426790407	0.37574088552938	0.03811468858563	0.35955405665117	0.00217136208877
0.37574088552938	0.03811468858563	0.35955405665117	-0.00217136208877	0.36042650420995	0.00001201181392
0.35955405665117	-0.00217136208877	0.36042650420995	0.00001201181392	**0.36042170444674**	**0.00000000371874**
0.36042650420995	0.00001201181392	0.36042170444674	0.00000000371874	**0.36042170296032**	**0.00000000000001**
0.36042170444674	0.00000000371874	0.36042170296032	-0.00000000000001	0.36042170296032	0.000000000000000

Table 2 shows that the function converges to root 0.36042170296032 at the iteration 5th with error level 0.00000000000001

Numerical roots for Newton method : The Newton method for a single-variable function $f(x) = 3x + \sin(x) - e^x$ on [0,1], using the software, matlab.

Table3.. Iteration data for newton method with initial Approximation $x_0 = 0$, $\varepsilon = 10^{-14}$

S/No.	x_k	$\lvert f(x_{k+1})\rvert$
1.	0.00000000000000	1.000000000000
2.	0.33333333333333	0.06841772828994
3.	0.36017071357763	6.279850705706025e-004
4.	**0.36042168047602**	**5.625155319322062e-008**
5.	**0.36042170296032**	**6.661338147750939e-016**
6.	0.36042170296032	4.440892098500626e-016
7.	0.36042170296032	2.220446049250313e-016

Table3. shows that the function converges to 0.36042170296032 at the iteration 4th with error level 0.00000000000001

From the above tables we see that the number of iteration for bisection method is too much in comparison to other two methods .Now note that we converged to almost the same root as for Newton method. but this time in 5 iteration as opposed to 4iteration for Newton . However, recall that Newton method is more costly per iteration. Newton achieved its accuracy with total of 8 calls to the function $f(x)$ and the derivatives $f'(x)$. Secant did it with only 5 calls, all to (x) . Secant was actually more efficient in terms of the number of total function call. Since $p^2 > 2$, so we can do two iteration of secant method for the same cost as one iteration of Newton method. Now we will compare Newton method and secant method based on execution time.

Executing time: The execution time of a given task is defined as the time spent by the system executing that task, including the time spent executing run time or system services on its behalf.

Table 4: Execution time comparison for four (04) iterations

S/No.	Newton raphson method (run time)	Secant method (run time)
1.	0.002652	0.002180
2.	0.002808	0.002030
3.	0.002824	0.002500
4.	0.003166	0.002650
5.	0.002793	0.002340

Table 5: Execution time comparison for five (05) iterations

S/No.	Newton raphson method (run time)	Secant method (run time)
1.	0.003129	0.002660
2.	0.002340	0.002020
3.	0.002867	0.002032
4.	0.002931	0.001720
5.	0.002792	0.001870

Thus from the above discussions we see that the Newton raphson method is taking more time in comparison to that of secant method.[15-16] Since the secant method requires only one function evaluation per iteration and Newton method requires the evaluation of both the function and derivative at every iteration.

Efficiency Index: The efficiency index of an iterative method is defined by the equation $E = p^{1/n}$ where p is the order of the method and n is the total number of call functions at each step of iteration. In this way, the efficiency index of secant method is 1.62, which is better than the 1.41 of Newton method and bisection method. Thus we conclude that the secant method has better overall performance than others method. So we observed that the rates of convergence of the above methods are in this manner.

$$\text{Bisection method} < \textit{Newton method} \leq \text{Secant method}$$

V. CONCLUSION

In Newton method**,** there is the need for two functions evaluations at each step, $f(x_n)$ and $f'(x_n)$ at the start. If a difficult problem requires much iteration to converge, the number of function evaluations with Newton's method may be many more than with linear iteration methods. Because Newton's method always uses two per iteration whereas the others take only one. Based on our results and discussions, we now conclude that the secant method is formally the most effective of the Newton method, we have considered here in the study. But requires only a single function evaluation per iteration. Analysis of efficiency from the numerical computation shows that bisection method converges too slow but sure. Thus these methods have great practical utilities.

REFERENCES:

[1]. S. Amat, S. Busquier, J.M. Gutierrez, Geometric constructions of iterative functions to solve nonlinear equations, J. Comput. Appl. Math. 157, (2003) 197-205.

[2]. K. E. Atkinson, An introduction to numerical analysis, 2nd ed., John Wiley & Sons, New York, 1987.

[3]. F. Costabile, M.I. Gualtieri, S.S. Capizzano, An iterative method for the solutions of nonlinear equations, Calcolo 30,(1999)17-34.

[4]. J. Stoer, R. Bulrisch, Introduction to Numerical Analysis, Second edition, Springer–Verlag,1993.

[5]. Ehiwario, J.C., Aghamie, S.O. Department of Mathematics, College of Education, Agbor, Delta State.

[6]. Biswa Nath Datta (2012), Lecture Notes on Numerical Solution of root Finding Problems. www.math.niu.edu/dattab.

[7]. Iwetan, C.N, Fuwape, I.A,Olajide, M.S and Adenodi, R.A(2012), Comparative Study of the Bisection and Newton Methods in solving for Zero and Extremes of a Single-Variable Function. J.of NAMP vol.21 pp 173-176.

[8]. 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.

[9]. http://www.efunda.com/math/num_rootfinding-cfm February, 2014
[10]. Wikipedia, the free Encyclopedia
[11]. Autar, K.K and Egwu, E (2008), http://www.numericalmethods.eng.usf.edu Retrieved on 20th February, 2014.
[12]. McDonough, J.M (2001), Lecture in Computationl Numerical Analysis. Dept. of Mechanical Engineering, University of Kentucky.
[13]. Allen, M.B and Isaacson E.L (1998), Numerical Analysis for Applied Science. John Wiley and sons.Pp188-195
[14]. T.Yamamoto, Historical development in convergence analysis for Newton's and Newton-like methods, J. Comput. Appl.Math,124,(2000),1-23.
[15]. Applied Numerical Methods Using MATLAB, Won Young Yang Chung-Ang , Wenwu Cao , Tae-Sang Chung Chung , John Morris , A John Willey & Sons, Inc, Publication 2005.
[16]. Applied Numerical Methods with MATLAB, for Engineers and Scientist, Third Edition, Steven C. Chapra.