ADV MATH SCI JOURNAL
Advances in Mathematics: Scientific Journal **10** (2021), no.12, 3733–37436
ISSN: 1857-8365 (printed); 1857-8438 (electronic)
https://doi.org/10.37418/amsj.10.12.13

COMPARISION OF NUMERICAL ACCURACY OF BISECTION, NEWTON RAPHSON, FALSI-POSITION AND SECANT METHODS

Narendra Deo Dixit and Parveen Kumar Mathur[1]

ABSTRACT. In this paper we make comparision of numerical accuracy of bisection, Newton-Raphson, falsi-position and secant methods.

1. INTRODUCTION

A root-finding algorithm is a numerical method, or algorithm, for finding a value x such that $f(x) = 0$, for a given function f. Such an x is called a root of the function f. We are concerned with finding real or complex roots, approximated as floating point numbers. Finding integer roots or exact algebraic roots are separate problems, whose algorithms have little in common with those discussed here.Finding a root of $f(x) - g(x) = 0$ is the same as solving the equation $f(x) = g(x)$.Here,x is called the unknown in the equation. Conversely, any equation can take the canonical form $f(x) = 0$, so equation solving is the same thing as computing (or finding) a root of a function.

Numerical root-finding methods use iteration, producing a sequence of numbers that hopefully converge towards a limit (the so called “fixed point") which is a root. The first values of this series are initial guesses. The method computes

[1]*corresponding author*
2020 *Mathematics Subject Classification.* 65Dxx, 65Z99.
Key words and phrases. numerical methods, accuracy, comparison.
Submitted: 16.11.2021; *Accepted:* 03.12.2021; *Published:* 01.02.2022.

subsequent values based on the old ones and the function f. The behaviour of root- finding algorithms is studied in numerical analysis. Algorithms perform best when they take advantage of known characteristics of the given function. Thus an algorithm to find isolated real roots of a low-degree polynomial in one variable may bear little resemblance to an algorithm for complex roots of a "black-box" function which is not even known to be differentiable. Questions include ability to separate close roots, robustness in achieving reliable answers despite inevitable numerical errors, and rate of convergence.

The simplest root-finding algorithm is the bisection method. It works when f is a continuous function and it requires previous knowledge of two initial guesses,a and b, such that $f(a)$ and $f(b)$ have opposite signs. Although it is reliable, it converges slowly, gaining one bit of accuracy with each iteration [1-3]. Newton's method assumes the function f to have a continuous derivative. Newton's method may not converge if started too far away from a root. However, when it does converge, it is faster than the bisection method. Convergence is usually quadratic, so the number of bits of accuracy doubles with each iteration. Newton's method is also important because it readily generalizes to higher-dimensional problems. Newton-like methods with higher order of convergence are the Householder's methods. The first one after Newton's method is Halley's method with cubic order of convergence. Replacing the derivative in Newton's method with a finite difference, we get the secant method. This method does not require the computation (nor the existence) of a derivative, but the price is slower convergence (the order is approximately 1.6)

The false position method, also called the regula falsi method, is like the secant method. However, instead of retaining the last two points, it makes sure to keep one point on the either side of the root. The Falsi Position Method is faster than the bisection method and more robust than the secant method.

The secant method also arises if one approximates the unknown function f by linear interpolation. When quadratic interpolation is used instead, one arrives at MÃijller's method. It converges faster than the secant method. A particular feature of this method is that the iterates x_n may become complex. This can be avoided by interpolating the inverse of f, resulting in the inverse quadratic interpolation method. Again, convergence is asymptotically faster than the secant method, but

inverse quadratic interpolation often behaves poorly [4] when the iterates are not close to the root.

Finally, Brent's method is a combination of the bisection method, the secant method and inverse quadratic interpolation. At every iteration, Brent's method decides which method out of these three is likely to do best, and proceeds by doing a step according to that method. This gives a robust and fast method, which therefore enjoys considerable popularity.

2. BISECTION METHOD

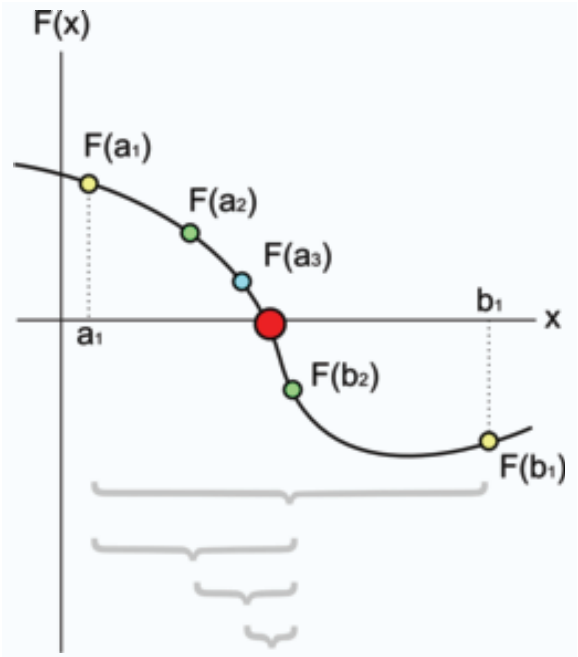

FIGURE 1. A few steps of the bisection method applied over the starting range $[a_1, b_1]$

In mathematics, the bisection method is a root-finding algorithm which repeatedly divides an interval in half and then selects the subinterval in which a root exists. It is a very simple and robust method, but it is also rather slow [5-6].

2.1. **The method.** Suppose we want to solve the equation

$$f(x) = 0$$

where f is a continuous function. The bisection method starts with two points a and b such that $f(a)$ and $f(b)$ have opposite signs. The intermediate value theorem says that f must have at least one root in the interval $[a, b]$. The method now divides the interval in two by computing $c = (a + b)/2$. There are now two possibilities: either $f(a)$ and $f(c)$ have opposite signs, or $f(c)$ and $f(b)$ have opposite signs. The bisection algorithm is then applied recursively to the sub-interval where the sign change occurs. Explicitly, if $f(a)f(c) < 0$, then the method sets b equal to c, and if $f(b)f(c) < 0$, then the method sets a equal to c. In both cases, $f(a)$ and $f(b)$ have again opposite signs, so the method can start again with the points a and b which now lie closer to each other.

2.2. **Analysis.** If f is a continuous function on the interval $[a, b]$ and $f(a)f(b) < 0$, then the bisection method converges to a root of f. In fact, the absolute error is halved at each step. Thus, the method converges linearly, which is quite slow. On the positive side, the method is guaranteed to converge if $f(a)$ and $f(b)$ have different signs.

The bisection method gives only a range where the root exists, rather than a single estimate for the root's location. Without using any other information, the best estimate for the location of the root is the midpoint of the range. In that case, the absolute error after n steps is at most

$$\frac{|b-a|}{2^{n+1}}$$

If either endpoint of the interval is used, then the maximum absolute error is

$$\frac{|b-a|}{2^n}$$

the entire length of the interval.

These formulas can be used to find the number of iterations that the bisection method needs to converge to a root within a certain tolerance. For instance, using the second formula for the error, the number of iterations n has to satisfy

$$n > \frac{\log(b-a) - \log\epsilon}{\log 2}$$

to make sure that the error is smaller than the tolerance ϵ.

If f has several roots in the interval $[a, b]$, then the bisection method finds the odd-numbered roots with equal, non-zero probability and the even-numbered

roots with zero probability. More precisely, suppose that f has x_{2k+1} simple roots $x_1 < x_2 < \ < x_{2k+1}$ in the interval $[a, b]$ (the number of roots is odd because $f(a)$ and $f(b)$ have opposite signs). If we assume that the roots are distributed independently and uniformly in this interval. Then, the probability that the bisection method converges to the root x_i with$i = 1, 2, \ldots, 2k + 1$ is zero if i is even and $1/(k + 1)$ if i is odd (Corliss 1977).

3. NEWTON'S METHOD

In numerical analysis, **Newton's method** (also known as the **Newton-Raphson method**, named after Isaac Newton and Joseph Raphson) is perhaps the best known method for finding successively better approximations to the zeroes (or roots) of a real-valued function. Newton's method can often converge remarkably quickly, especially if the iteration begins "sufficiently near" the desired root. Just how near "sufficiently near" needs to be, and just how quickly "remarkably quickly" can be, depends on the problem. This is discussed in detail below. Unfortunately, when iteration begins far from the desired root, Newton's method can easily lead an unwary user astray with little warning. Thus, good implementations of the method embed it in a routine that also detects and perhaps overcomes possible convergence failures.

Newton's method can also be used to find a minimum or maximum of such a function, by finding a zero in the function's first derivative, see Newton's method as an optimization algorithm. The algorithm is first in the class of Householder's methods, succeeded by Halley's method.

3.1. **Description of the method.**

The idea of the method is as follows: one starts with an initial guess which is reasonably close to the true root, then the function is approximated by its tangent line (which can be computed using the tools of calculus), and one computes the x-intercept of this tangent line (which is easily done with elementary algebra). This x-intercept will typically be a better approximation to the function's root than the original guess, and the method can be iterated.

Let us suppose $f : [a, b] \longrightarrow R$ is a differentiable function defined on the interval $[a, b]$ with values in the real numbers R. The formula for converging on the root can be easily derived. Suppose we have some current approximation x_n. Then we can derive the formula for a better approximation, x_{n+1} by referring to the

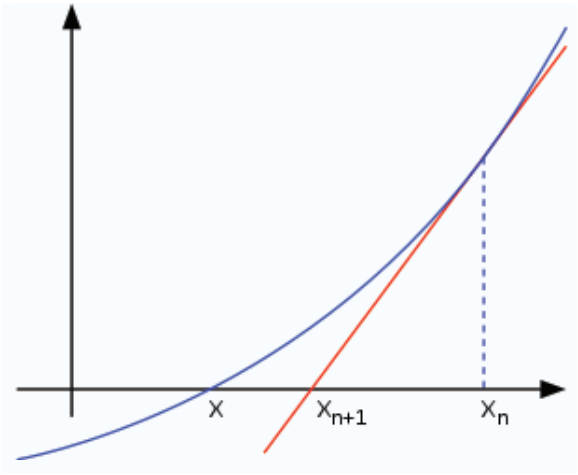

FIGURE 2. An illustration of one iteration of Newton's method (the function f is shown in blue and the tangent line is in red). Clearly x_{n+1} is a better approximation than x_n for the root x of the function f.

diagram on the right. We know from the definition of the derivative at a given point that it is the slope of a tangent at that point.

That is

$$f'(x_n) = \frac{rise}{run} = \frac{\Delta y}{\Delta x} = \frac{f(x_n) - 0}{x_n - x_{n+1}}.$$

Here,f' denotes the derivative of the function f. Then by simple algebra we can derive

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

We start the process off with some arbitrary initial value x_0. (The closer to the zero, the better. But, in the absence of any intuition about where the zero might lie, a "guess and check" method might narrow the possibilities to a reasonably small interval by appealing to the intermediate value theorem.) The method will usually converge, provided this initial guess is close enough to the unknown zero, and that $f'(x_0)0$. Furthermore, for a zero of multiplicity 1, the convergence is at least quadratic (see rate of convergence) in a neighbourhood of the zero, which intuitively means that the number of correct digits roughly at least doubles in every step. More details can be found in the analysis section below.

3.2. **History.** Newton's method was described by Isaac Newton in De analysi per aequationes numero terminorum infinitas (written in 1669, published in 1711 by William Jones) and in De metodis fluxionum et serierum infinitarum (written in 1671, translated and published as Method of Fluxions in 1736 by John Colson). However, his description differs substantially from the modern description given above: Newton applies the method only to polynomials. He does not compute the successive approximations x_n, but computes a sequence of polynomials and only at the end, he arrives at an approximation for the root x [15]. Finally, Newton views the method as purely algebraic and fails to notice the connection with calculus. Isaac Newton probably derived his method from a similar but less precise method by FranÃğois ViÃĺte. The essence of ViÃĺte's method can be found in the work of the Persian mathematician, Sharaf al-Din al-Tusi, while his successor JamshÄńd al-KÄĄshÄń used a form of Newton's method to solve $x^P - N = 0$ to find roots of N (Ypma 1995). A special case of Newton's method for calculating square roots was known much earlier and is often called the Babylonian method.

Newton's method was first published in 1685 in A Treatise of Algebra both Historical and Practical by John Wallis. In 1690, Joseph Raphson published a simplified description in Analysis aequationum universalis. Raphson again viewed Newton's method purely as an algebraic method and restricted its use to polynomials, but he describes the method in terms of the successive approximations x_n instead of the more complicated sequence of polynomials used by Newton. Finally, in 1740, Thomas Simpson described Newton's method as an iterative method for solving general nonlinear equations using fluxional calculus, essentially giving the description above. In the same publication, Simpson also gives the generalization to systems of two equations and notes that Newton's method can be used for solving optimization problems by setting the gradient to zero.

Arthur Cayley in 1879 in The Newton-Fourier imaginary problem was the first who noticed the difficulties in generalizing the Newton's method to complex roots of polynomials with degree greater than 2 and complex initial values. This opened the way to the study of the theory of iterations of rational functions.

4. FALSE POSITION METHOD

In numerical analysis, the **false position method** or **regula falsi method** is a root-finding algorithm that combines features from the bisection method and the secant method.

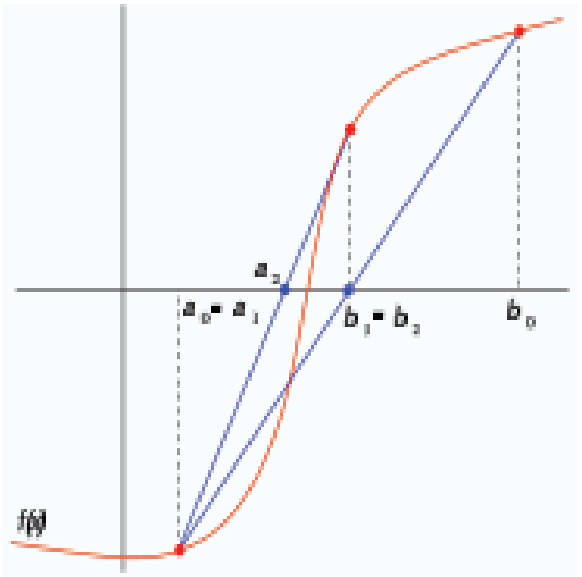

FIGURE 3. The first two iterations of the false position method. The light curve shows the function f and the blue lines are the secants.

4.1. **The method.** Like the bisection method, the false position method starts with two points a_0 and b_0 such that $f(a_0)$ and $f(b_0)$ are of opposite signs, which implies by the intermediate value theorem that the function f has a root in the interval $[a_0, b_0]$. The method proceeds by producing a sequence of shrinking intervals $[a_k, b_k]$ that all contain a root of f.

At iteration number k, the number

$$c_k = \frac{f(b_k)a_k - f(a_k)b_k}{f(b_k) - f(a_k)}$$

is computed.

As explained below, C_k is the root of the secant line through $(a_k, f(a_k), b_k, f(b_k))$ and $(b_k, f(b_k))$. If $f(a_k)$ and $f(c_k)$ have the same sign, then we set $a_{k+1} = c_k$ and $b_{k+1} = b_k$, otherwise we set $a_{k+1} = a_k$ and $b_{k+1} = c_k$. This process is repeated until the root is approximated sufficiently well [7-8].

The above formula is also used in the secant method, but the secant method always retains the last two computed points, while the false position method retains two points which certainly bracket a root. On the other hand, the only difference between the false position method and the bisection method is that the latter uses $c_k = (a_k + b_k)/2$.

5. FINDING THE ROOT OF THE SECANT

Given a_k and b_k, we construct the line through the points $(a_k, f(a_k))$ and $(b_k, f(b_k)$, as demonstrated in the picture on the right. Note that this line is a secant or chord of the graph of the function f. In point-slope form, it can be defined as [9].

$$y - f(b_k) = \frac{f(b_k) - f(a_k)}{b_k - a_k}(x - b_k).$$

We now choose c_k to be the root of this line, so c is chosen such that

$$f(b_k) + \frac{f(b_k) - f(a_k)}{b_k - a_k}(c_k - b_k) = 0.$$

Solving this equation gives the above equation for c_k.

5.1. **Analysis.** If the initial end-points a_0 and b_0 are chosen such that $f(a_0)$ and $f(b_0)$ are of opposite signs, then one of the end-points will converge to a root of f. Asymptotically, the other end-point will remain fixed for all subsequent iterations while the one end- point always being updated. As a result, unlike the bisection method, the width of the bracket does not tend to zero. As a consequence, the linear approximation to $f(x)$, which is used to pick the false position, does not improve in its quality [10].

One example of this phenomenon is the function

$$f(x) = 2x^3 - 4x^2 + 3x$$

on the initial bracket $[-1, 1]$. The left end,-1, is never replaced and thus the width of the bracket never falls below 1. Hence, the right endpoint approaches 0 at a linear rate (with a rate of convergence of $2/3$).

While it is a misunderstanding to think that the method of false position is a good method, it is equally a mistake to think that it is unsalvageable. The failure mode is easy to detect (the same end-point is retained twice in a row) and easily

remedied by next picking a modified false position, such as

$$c_k = \frac{\frac{1}{2}f(b_k)a_k - f(a_k)b_k}{\frac{1}{2}f(b_k) - f(a_k)}$$

or

$$c_k = \frac{f(b_k)a_k - \frac{1}{2}f(a_k)b_k}{f(b_k) - \frac{1}{2}f(a_k)}$$

down-weighting one of the endpoint values to force the next c_k to occur on that side of the function. The factor of 2 above looks like a hack, but it guarantees superlinear convergence (asymptotically, the algorithm will perform two regular steps after any modified step). There are other ways to pick the rescaling which give even better superlinear convergence rates [11].

Ford (1995) summarizes and analyzes the superlinear variants of the modified method of false position. Judging from the bibliography, modified regula falsi methods were well known in the 1970s and have been subsequently forgotten or misremembered in current textbooks.

6. Secant method

In numerical analysis, the **secant method** is a root-finding algorithm that uses a succession of roots of secant lines to better approximate a root of a function f.

The secant method is defined by the recurrence relation

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

As can be seen from the recurrence relation, the secant method requires two initial values, x_0 and x_1, which should ideally be chosen to lie close to the root.

7. Derivation of the method

Given x_{n-1} and x_n, we construct the line through the points $(x_{n-1}, f(x_{n-1}))$ and $(x_n, f(x_n))$, as demonstrated in the picture on the right. Note that this line is a secant or chord of the graph of the function f. In point-slope form, it can be defined as

$$y - f(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}(x - x_n).$$

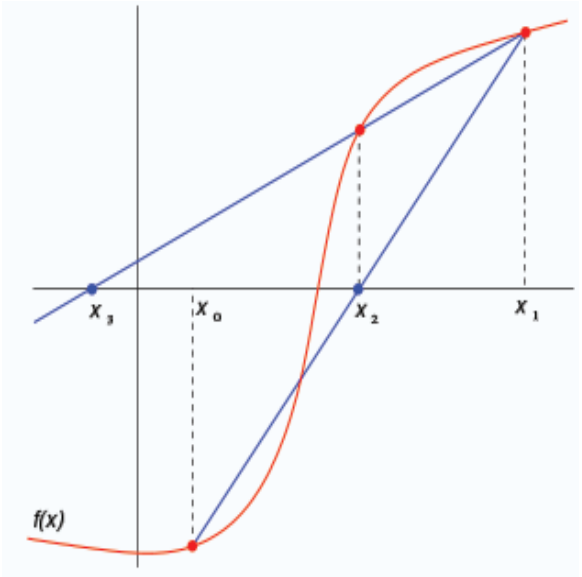

FIGURE 4. The first two iterations of the secant method. The light curve shows the function f and the blue lines are the secants.

We now choose x_{n+1} to be the root of this line, so x_{n+1} is chosen such that

$$f(x_n) + \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}(x_{n+1} - x_n) = 0.$$

Solving this equation gives the recurrence relation for the secant method.

7.1. **Convergence.** The iterates x_n of the secant method converge to a root of f, if the initial values x_0 and x_1 are sufficiently close to the root. The order of convergence is $alpha$, where

$$\alpha = \frac{1 + \sqrt{5}}{2} \approx 1.618$$

is the golden ratio. In particular, the convergence is superlinear [12].

This result only holds under some technical conditions, namely that f be twice continuously differentiable and the root in question be simple (i.e., that it not be a repeated root). If the initial values are not close to the root, then there is no guarantee that the secant method converges.

8. Comparison with other root-finding methods

The secant method does not require that the root remain bracketed like the bisection method does, and hence it does not always converge. The false position method uses the same formula as the secant method. However, it does not apply the formula on x_{n-1} and x_n, like the secant method, but on x_n and on the last iterate x_k such that $f(x_k)$ and $f(x_n)$ have a different sign. This means that the false position method always converges [13].

The recurrence formula of the secant method can be derived from the formula for Newton's method

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

by using the finite difference approximation

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

If we compare Newton's method with the secant method, we see that Newton's method converges faster (order 2 against âĽĹ 1.6). However, Newton's method requires the evaluation of both f and its derivative at every step, while the secant method only requires the evaluation of f [14]. Therefore, the secant method may well be faster in practice. For instance, if we assume that evaluating f takes as much time as evaluating its derivative and we neglect all other costs, we can do two steps of the secant method (decreasing the logarithm of the error by a factor $\alpha_2 2.6$) for the same cost as one step of Newton's method (decreasing the logarithm of the error by a factor 2), so the secant method is faster.

9. Proposed Work

We will develop C/C_{++} programs for the root finding methods viz. Bisection method, Newton-Raphson, False Position method and Secant method. With the help of these computer programs, we will find the square root, cube root, fourth root and fifth roots of the natural numbers from 1 to 50 up to eight decimal places. We will compare these values with the exact values to get the numerical accuracy of the root finding methods described above. We will also try to establish multilinear regression to get the exact values with the help of obtained values. Multilinear regression equations will enable us to find out the square root, cube root, fourth

root and fifth roots of the natural numbers using the arithmetical operators addition and multiplication. Lastly, we will arrange these root finding methods in order of their numerical accuracy and rate of convergence.

REFERENCES

[1] A.B. BABICHEV, O.B. KADYROVA, T.P. KASHEVAROVA, A.S. LESHCHENKO, A.L. SEMENOV: *A Novel Approach to Solving Systems of Algebraic Equations*, Proceedings of the International Conference on Numerical Analysis with Automatic Result Verifications, Lafayette, Louisiana, USA, (1993).

[2] J.P. DEDIEU, J.C. YAKOBSOHN: *Computing the Real Roots of a Polynomial by the Exclusion Algorithm*, Numerical Algorithms **4** (1993), 1-24.

[3] D.YU. GRIGOR'EV, N.N. VOROBJOV: *Solving Systems of Polynomial Inequalities in Subexponential Time*, J. Symbolic Computation **5** (1988), 37-64.

[4] S. MCCALLUM: *Solving Polynomial Strict Inequalities Using Cylindrical Algebraic Decomposition*, The Computer Journal, **36**(5) (1993), 432-438.

[5] R.L. BURDEN, J.D. FAIRES: *Numerical Analysis (7th ed.), Brooks/Cole, Corliss, George (1977)*, "Which root does the bisection algorithm find?", SIAM Review **19**(2) (2000),325âĂŞ327.

[6] J.A. FORD: *Improved Algorithms of Illinois-type for the Numerical Solution of Nonlinear Equations*, Technical Report CSM-257, University of Essex, 1995.

[7] R.L. BURDEN, J.D. FAIRES: *Numerical Analysis*, (7th Ed), Brooks/Cole, (2000).

[8] L.E. SIGLER: *Fibonacci's Liber Abaci*, Leonardo Pisno's Book of Calculation (2002), Springer-Verlag, New York.

[9] R.L. BURDEN, J.D. FAIRES: *Numerical Analysis*, PWS, Boston, 1993.

[10] G. ARFKEN: *Mathematical Methods for Physicists*, 3rd ed. Orlando, FL: Academic Press, (1985), 964-965.

[11] W.H. PRESS, B.P. FLANNERY, S.A. TEUKOLSKY, W.T. VETTERLING: *Bracketing and Bisection*, Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed. Cambridge, England: Cambridge University Press, (1992), 343-347.

[12] M.T. HEATH: *Scientific Computing: An Introductory Survey*, Second Edition, McGraw-Hill, (2002).

[13] J.M. BORWEIN, B.P. BORWEIN: *Pi and the AGM: A Study in Analytic Number Theory and Computational Complexity*, Wiley Interscience, (1987).

[14] T.J. YPMA: *Historical development of the Newton-Raphson method*, SIAM Review **37**(4) (1995), 531âĂŞ551.

B N COLLEGE OF ENGINEERING AND TECHNOLOGY
LUCKNOW, INDIA.

B N COLLEGE OF ENGINEERING AND TECHNOLOGY
LUCKNOW, INDIA.
Email address: drpraveenmathur8@gmail.com