Journal of the Nigerian Association of Mathematical Physics
Volume 10 (November 2006), **611 - 615**

On iterative solution of non-linear equation

***O. Ogbereyivwe**
Department of Natural and Applied sciences
Federal polytechnic, Auchi, Nigeria.
and
O. Izevbizua.
Department of Mathematics,
University of Benin, Benin City, Nigeria.

In solving non – linear equations by iterative method, Ohirhian (1994), (2005) [4],[5] developed a new algorithm based on cubic interpolation for solving non- linear equations of degree 1 to 3. The algorithm was found to be faster than the Regular falsi and the Newton Raphason method. This paper extend the algorithm to solving non linear equations of degree n by deriving a general formular, also some of the iteration procedures are reviewed for ease in computation.

1.0 Introduction

Let *f* be a real-valued function of a real variable. Any real number x for which $f(x) = 0$ is called a root of the equation or a zero of *f.* For example, the function.

$$f(x) = 6x^2 - 7x + 2$$

has $\frac{1}{2}$ and $\frac{2}{3}$ as zeros. Also $g(x) = \cos 3x - \cos 7x$

has not only the obvious zero, $x = 0$ but every integer multiple of $\frac{\pi}{5}$ and $\frac{\pi}{2}$ as well, as seen from the trigonometric identify, $CosA - CosB = 2\,Sin\frac{1}{2}(A + B)\,Sin\frac{1}{2}(B - A)$.

Because of the importance of finding solutions to scientific problems, model in the form of non linear equations like.

$$f(x) = 0 \qquad (1.1)$$

below.

We consider the algorithm in Section 2.0. There are practically no formula of finding solution to (1.1) except in a few simple cases, so that one depends almost entirely on numerical algorithms. A solution of (1.1) is a number $x = \alpha$ such that $f(\alpha) = 0$. Since there is no formula for the exact solution, we can use an approximation method, in particular an iteration method, that is, a method of solving a problem by successive approximations in such a way that each approximation comes out with a more accurate estimate by using the preceding approximation [1].

Several iterative methods have been developed including Ohirhian (1994), (2005) [4], [5]. His method was developed from the Langangian interpolation polynomial and was able to find solution to (1.1) where $f(x)$ is of degree up to 3. In this paper, an extension based on Ohirhian was considered up to degree *n* of $f(x)$ in (1.1).

* **Corresponding Author**

2.0 Formation of the algorithm.

The Langrangian interpolation polynomial of degree n is given as

$$f(x) \cong P_n(x) = \sum_{i=0}^{n} L_i(x) f_i \tag{2.1}$$

where
$$L_i(x) = \prod_{j=0,1\neq j}^{n} \frac{(x - x_j)}{(x^i - x_j)} \tag{2.2}$$

if x and y are interchanged such that $x_i = y_j$ and set $y = 0$ in (2.1) and (2.2), we have

$$X_n == \sum_{j=0}^{n} (-1)^n L_i^1(x) f_i' \tag{2.3}$$

where
$$L_i' = \prod_{j=0,i\neq j}^{n} \frac{y_i}{(y_i - y_j)} \tag{2.4}$$

If we introduce an iteration counter (k) we have
$$X^{k+1} == \sum_{k=0}^{n} (-1)^n L_{ii}^{k1}(x) f_k \tag{2.5}$$

where $k = 0,1,2,\cdots,n$ and

$$L_i' = \prod_{j=0,i\neq j}^{n} \frac{y_i}{(y_i^k - y_j^k)} \tag{2.6}$$

***Theorem* 2.1**

Suppose $f(x)$ is a continuous function on the closed interval $[\alpha_1,\alpha_2]$ *and* $f(\alpha_1).f(\alpha_2) < 0$ *then there exist a number* $\alpha \in [\alpha_1,\alpha_2]$ *with* $f(\alpha) = 0$.

Applying (2.5) in finding the solution of (1.1) such that $f(x)$ is of degree n and satisfies Theorem 2.1 on $[\alpha_1,\alpha_2]$, we set $x_0 = \alpha_1$ and $x_n = \alpha_2$ and then find points $x_1,x_2,\ldots,x_{n-1} \in [\alpha_1,\alpha_2]$ with equal interval h such that $x_1 = x_0 + h$, $x_2 = x_1 + h$, $x_3 = x_2 + h, \ldots, x_n = x_{n-1} + h$ where

$$h = \frac{x_n - x_0}{n} \tag{2.7}$$

and n the degree of the polynomial. We generate the initial iterative points as

$$\begin{array}{cc} x_0 & y_0 \\ x_1 & y_1 \\ x_2 & y_2 \\ . & . \\ . & . \\ . & . \\ x_n & y_n \end{array}$$

and X^{k+1} is calculated using (2.5) and (2.6).

If
$$x_n < X^{k+1} < x_{n+1} \tag{2.8a}$$

We substitute as
$$\cdots, X_{n+1} = X_{n+1},\ X_n = X^{k+1},\ X_{n-1} = X_n,\ X_{n-2} = X_{n-1}, \cdots, X_0 = X_1 \tag{2.8b}$$

or if
$$X^{k+1} < x_0 \tag{2.9a}$$

we substitute as:

$$X_0 = X^{k+1},\ X_1 = X_0,\ X_2 = X_1, \ldots, X_n = X_{n-1} \tag{2.9b}$$

or if
$$X^{k+1} > X_n \tag{2.10a}$$

we substitute as :

$$X_n = X^{k+1}, X_{n-1} = X_n, X_{n-2} = X_{n-1} \ldots, X_0 = X_1 \tag{2.10b}$$

and update the iteration points.

Convergence is achieved if:

i) $|X^{k+1} - X^k| \leq \text{Tol}$ (2.11)

ii) $|f(X^{k+1})| \leq \text{Tol}$ (2.12)

where Tol is a specified tolerance.

3.0 Implementation.

Several tests have been carried out on various equations of different degrees. One of such equations is the Leonardo's equations $x^3 + 2x^2 + 10x - 20 = 0$ given that the root lies between 1 and 1.5. Comparisons were made with the 'Regular falsi method' and 'Newton Raphason method' and was found that the new algorithm prove superior over both. [5].

Here, we also test the improved algorithm on quintic equation. At each iteration point, approximate error was calculated using the error estimator given by

$$\varepsilon_a = |A_v - E_v| \tag{3.1}$$

where A_v = actual value and E_v = estimated value.

Comparison were made with the Newton Raphason approximations.

***Test* 1**

To determine the root of $f(x) = x^5 - 0.2$ wthin an interval of [0.5, 1.0]. Here, $n = 5$, therefore (2.5) becomes

$$X^{k+1} = \sum_{1=0}^{5} (-1)^5 L_i^k(X) f_i^k \tag{3.2}$$

and (2.6) becomes

$$L_i' = \prod_{j=0,i\neq j}^{5} \frac{y_j^k}{\left(y_i^k - y_j^k\right)} \tag{3.3}$$

with $h = 0.1$, we generate the initial iterative points and then use these points on (3.2) and (3.3) we have the following results.

3.1 Initial iteration values.

$x_0 = 0.5$ $y_0 = -0.16875$
$x_1 = 0.6$ $y_1 = -0.12224$
$x_2 = 0.7$ $y_2 = -0.03193$
$x_3 = 0.8$ $y_3 = 0.12768$
$x_4 = 0.9$ $y_4 = 0.39049$
$x_5 = 1.0$ $y_5 = 0.8$

$X^1 = 0.70251256$

Now $x_2 < X^1 < x_3$. is of type (2.8) thus, $x_2 = X^1$, $x_1 = x_2$, $x_0 = x_1$ and we update the iteration points.

3.2 Second iteration

$x_0 = 0.6$ $y_0 = -0.12224$
$x_1 = 0.7$ $y_1 = -0.03193$
$x_2 = 0.70251256$ $y_2 = -0.028891939$
$x_3 = 0.8$ $y_3 = 0.12768$
$x_4 = 0.9$ $y_4 = 0.39049$
$x_5 = 1.0$ $y_5 = 0.8$

$X^2 = 0.704614096.$

3.3 Third iteration

$x_0 = 0.7$ $y_0 = -0.03193$
$x_1 = 0.70251256$ $y_1 = -0.028891939$
$x_2 = 0.724614096$ $y_2 = -0.000228334048$
$x_3 = 0.8$ $y_3 = 0.12768$

$x_4 = 0.9$ $y_4 = 0.39049$
$x_5 = 1.0$ $y_5 = 0.8$

$X^3 = 0.724784708.$

3.4 Fourth iteration.

$x_0 = 0.70251256$ $y_0 = -0.028891939$
$x_1 = 0.724614096$ $y_1 = -0.000228334048$
$x_2 = 0.7247847048$ $y_2 = -0.000006959897$
$x_3 = 0.8$ $y_3 = 0.12768$
$x_4 = 0.9$ $y_4 = 0.39049$

$x_5 = 1.0$ $y_5 = 0.8$

$X^4 = 0.724779663.$

The table below shows the iterations values and error of the new algorithm.

k	X^k	ε_a
1	0.70251256	2.2267103×10^{-2}
2	0.724614096	1.65567×10^{-4}
3	0.724784708	5.045×10^{-6}
4	0.724779663	0.0

Table 1: New Algorithm result.

The Newton Raphason iteration formular is given as:

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)} \qquad (3.4)$$

with $x_0 = 0.5$ we have

n	X_{n+1}	ε_a
1	1.04	$3.15220337 \times 10^{-1}$
2	0.866192169	$1.1414125044 \times 10^{-1}$
3	0.764010097	3.9230434×10^{-2}
4	0.728606861	3.827198×10^{-3}
5	0.724819659	3.9996×10^{-5}
6	0.724779667	4.0×10^{-9}
7	0.724779663	0.0

Table 2: Newton Raphason Algorithm result.

From the tables above, we observed that the algorithm used in Table 1 takes 4 iterations to converge, while the Newton Raphason used in Table 2, takes 7 iterations to converge. This implies that the new algorithm prove superior over the Newton Raphason method in terms of rate of convergence.

4.0 Choice of degree (*n*) for *f(x)*

It was observed that the choice of n in (2.7) may not necessary be that n must be the degree of $f(x)$. However, if n is the degree of $f(x)$, convergence is faster than the assumed degree n, but still better than the Newton Raphason method. This was first illustrated with the question in Test 1.

***Test* 2**

Here, we take $n = 2$ and from (2.7) we have $h = 0.25$. Thus (2.5) and (2.6) becomes:

$$X^{k+1} = \sum_{1=0}^{2} (-1)^2 L_i^k(X) f_i^k \qquad (3.5)$$

with

$$L_I^K = \prod_{j=0, i\neq j}^{2} \frac{y_j^k}{(y_i^k - y_j^k)} \qquad (3.6)$$

we have initial iteration values as :

4.1 Initial iteration values

$x_0 = 0.5$ $y_0 = 0.16875$

$x_1 = 0.75$ $y_1 = 0.037304687$

$x_2 = 1.0$ $y_2 = 0.8$

we achieved the following iterative results using (3.5) and (3.6).

k	X^k	ε_a
1	0.843186899	$2.18407236 \times 10^{-1}$
2	0.729162889	4.383226×10^{-3}
3	0.72556478	7.84915×10^{-4}
4	0.724785966	6.303×10^{-6}

Table 3: Iteration with degree less than the degree of *f*(*x*).

The freedom of choice of *n* for a function could be used to correct some of the pitfalls of the Newton Raphason method. An example of a slowly converging function with Newton Raphason is in determining the root of $f(x) = x^{10} - 1$ with $x_o = 0.5$. Steven and Raymond [3] shown that iterations up to

∞ will be attain before converging. If we choose to take $n = 3$ in the interval of [0.5, 1.5] using (2.5) and (2.6) it was found that convergence will be achieved in within 7 iterations.

5.0 Other functions

Since there is freedom in the choice of *n* as illustrated above, this implies that the algorithm can be applied on functions other than the non- linear polynomials. But the choice of *n* depends on ones knowledge of the functions and its interval of convergence. In this case, as n tends to the actual degree of f(x), convergence becomes faster but strenuous with manual computation, and as n becomes smaller, convergence becomes slower but easier with manual computation and in either cases convergence is sure.

***Test* 3.**

To determine the root of. $x - \sin x = 0$ within the interval [0.5,1]. We observed that using (3.5) and (3.6) with tolerance of 10^{-10}, the solution is achieved with only 2 iterations while it takes about 8 iterations to converges at the zero of the equation with $x_0 = 0.5$ using the Newton Raphason's method.

6.0 Conclusion

The new algorithm developed above converges faster than the existing methods and convergence is sure irrespective of the distance between initial guess and the actual value.

Reference

[1] V.P Jaggi (2006). Dictionary of mathematics. Academic (Indian) Publishers. Vol. 1, pp 133.

[2] Erwin Kreyszig (1999). Advanced Engineering mathematics. John Wiley and sons, Inc. 8th Edition, pp841 – 851.

[3] Steven C.C. and Raymond P.C (1998).Numerical methods for Engineers. The Macgraw – Hill Companies, Inc. 3rd Edition, pp 147 – 152, 485 – 489.

[4] Ohirhian P. U. (1994). Iterative solution of Non–Linear Equations. Proceeding of SPE/IACMAG international conference, Morgantown. Vol. 8, pp 242 – 250.

[5] Ohirhian P.U. (2005). A cubic interpolation Algorithm for solving non- linear equations. Journal of the NAMP, Vol 9, pp 317 – 324.