Palestine Journal of Mathematics

Vol. 10(1)(2021) , 128–134

NEW ALGORITHMS FOR COMPUTING A ROOT OF NON-LINEAR EQUATIONS USING EXPONENTIAL SERIES

Srinivasarao Thota, Tekle Gemechu and P. Shanmugasundaram

Communicated by Suheel Khoury (S. Khuri)

MSC 2010 Classifications: Primary 65Hxx; Secondary 65H04.

Keywords and phrases: Root finding algorithms, Non-linear equations, Exponential series, Real root.

The authors are thankful to the editor and reviewer for providing valuable inputs to improve the present format of manuscript.

Abstract In this paper, we present new algorithms/methods to find a non-zero real root of the transcendental equations using exponential series. The new proposed method is based on the exponential series, which produces better approximate root than some existing methods. MATLAB and Maple implementation of the proposed method is discussed. Certain numerical examples are presented to validate the efficiency of the proposed algorithm. The method will help to implement in the commercial package for finding a real root of a given transcendental equation.

1 Introduction

The non-linear problems solving in science, engineering and computing are playing important role to compute roots of transcendental equations. A root of a function $f(x)$ is a number 'α' such that $f(\alpha) = 0$. Generally, the roots of transcendental functions cannot be expressed in closed form or cannot be computed analytically. The root-finding algorithms provide us approximations to the roots, these approximations are expressed either as small isolating intervals or as floating point numbers. Most of the algorithms in the literature use iteration, producing a sequence of numbers that hopefully converge towards the root as a limit. They need one or more initial guesses of the root as starting values, then each new iteration of the method produces a successively more accurate approximate root in comparison to previous iteration. The purpose of existing methods is to provide higher order convergence with guaranteed root. The existing methods may not guarantee that they will find all the roots; in particular, if such an algorithm does not find any root, that does not mean that no root exists. There are many well known root finding algorithms available, (for example, *Bisection*, *Secant*, *Regula-Falsi*, *Newton-Raphson*, *Muller's* methods etc.) to find an approximate root of algebraic or transcendental equations, see for example [1, 2, 4–7, 9–12, 14–22, 25–27]. If the equation $f(x) = 0$ is an algebraic equation, then there are many algebraic formulae available to find the roots. However, if $f(x)$ is a polynomial of higher degree or an expression involving transcendental equations such as trigonometric, exponential, algorithmic etc., then there are no algebraic methods exist to express the root.

In this work, the proposed new methods are based on exponential series, which provides faster roots in comparison with existing algorithms. The new proposed algorithms will be useful for computing a real root of transcendental equations. The Newton-Raphson method can be derived as a special case of the proposed method, so we select a non-zero initial approximation a for a given transcendental function $f(x)$ such that $f'(a) \neq 0$.

The rest of the paper is as follows: Section 2 describes the proposed method, their mathematical formulation, calculation steps and flow chart; implementation of the proposed algorithm in Maple is presented in Section 3 with sample computations; and Section 4 discuss some numerical examples to illustrate the algorithm and comparisons are made to show efficiency of the new algorithm.

2 An Exponential Series Based Algorithm

The new iterative formulae using exponential series are proposed as

$$x_{n+1} = x_n \exp\left(\frac{-f(x_n)}{x_n f'(x_n)}\right), \quad n = 0, 1, 2, \ldots. \tag{2.1}$$

By expanding this iterative formula, one can obtain the standard Newton-Raphson method as in first two terms, and many methods are obtained based on series truncation. Indeed,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{2.2}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} + \frac{1}{2x_n}\left(\frac{f(x_n)}{f'(x_n)}\right)^2. \tag{2.3}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} + \frac{1}{2x_n}\left(\frac{f(x_n)}{f'(x_n)}\right)^2 - \frac{1}{6x_n^2}\left(\frac{f(x_n)}{f'(x_n)}\right)^3. \tag{2.4}$$

This is shown in the following theorem.

Theorem 2.1. *Suppose $\alpha \neq 0$ is a real exact root of $f(x)$ and θ is a sufficiently small neighbourhood of α. Let $f''(x)$ exist and $f'(x) \neq 0$ in θ. Then the iterative formula given in equation (2.1) produces a sequence of iterations $\{x_n : n = 0, 1, 2, \ldots\}$ with order of convergence $p \geq 2$.*

Proof. The iterative formula given in equation (2.1) can be expressed in the following form

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

Since

$$\lim_{x_n \to \alpha} \exp\left(\frac{-f(x_n)}{x_n f'(x_n)}\right) = 1,$$

and hence $x_{n+1} = \alpha$.

Using the standard expansion of e^x as

$$\exp(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \cdots \tag{2.5}$$

and from equations (2.1) and (2.5), we have

$$\begin{aligned} x_{n+1} &= x_n \exp\left(\frac{-f(x_n)}{x_n f'(x_n)}\right) \\ &= x_n\Big(1 + \left(\frac{-f(x_n)}{x_n f'(x_n)}\right) + \frac{1}{2}\left(\frac{-f(x_n)}{x_n f'(x_n)}\right)^2 + \frac{1}{6}\left(\frac{-f(x_n)}{x_n f'(x_n)}\right)^3 \\ &\quad + \frac{1}{24}\left(\frac{-f(x_n)}{x_n f'(x_n)}\right)^4 + \cdots\Big) \\ &= x_n - \frac{f(x_n)}{f'(x_n)} + \frac{1}{2x_n}\left(\frac{f(x_n)}{f'(x_n)}\right)^2 - \frac{1}{6x_n^2}\left(\frac{f(x_n)}{f'(x_n)}\right)^3 \\ &\quad + o\left(\frac{1}{24x_n^3}\left(\frac{f(x_n)}{f'(x_n)}\right)^4\right). \end{aligned}$$

Since $f(x_n) \approx 0$, when we neglect higher order terms, then the above equation becomes Newton-Rapson method. Indeed, we have the following formulae obtained from first two terms, three

terms and four terms of the expansion respectively as given in equations (2.2)-(2.4).

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

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} + \frac{1}{2x_n}\left(\frac{f(x_n)}{f'(x_n)}\right)^2.$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} + \frac{1}{2x_n}\left(\frac{f(x_n)}{f'(x_n)}\right)^2 - \frac{1}{6x_n^2}\left(\frac{f(x_n)}{f'(x_n)}\right)^3.$$

In the above equations, we obtained Newton-Rapson method having quadratic convergence in first two terms. Therefore, the order of convergence of proposed methods (2.1), (2.3) and (2.4) are at least $p \geq 2$. □

2.1 Steps for Computing Root

I Select an approximation $x_n \neq 0$ and $f'(x_n) \neq 0$.

II Apply the iterative formula given in equation (2.1).

III Repeat Step II until we get desired approximate root, for $n = 0, 1, 2, \ldots$.

Flow chat of the proposed algorithm is presented in Figure 1

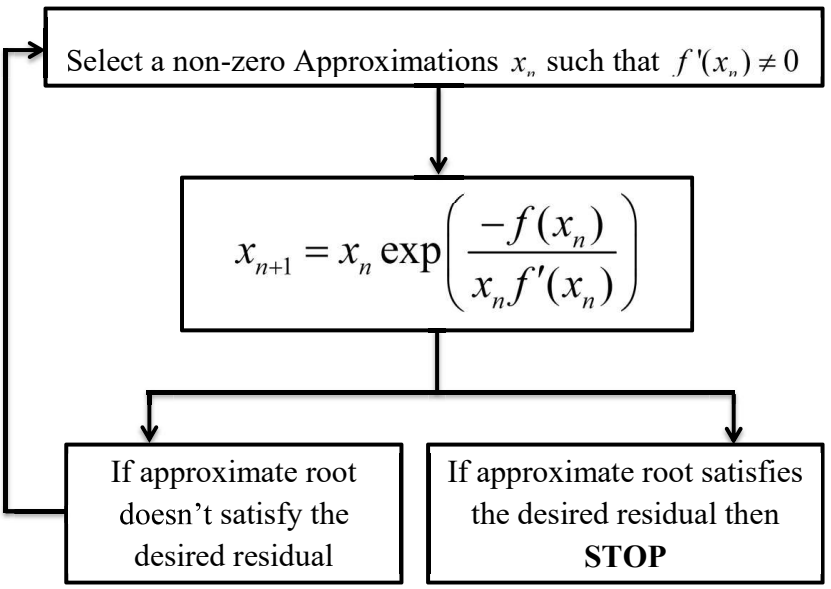

Figure 1. Flow chart for proposed algorithm

3 Implementation of Proposed Algorithm

In this section, we discuss the MATLAB and Maple implementation of the proposed algorithm. The data type `ExpNewton(f,x0,esp,n)` describes the implementation in MATLAB, where `f` is given non-linear transcendental function, `x0` is the initial approximation of the root, `esp` is the relative error and `n` is the number of iterations required.

```
function root = ExpNewton(f,x0,esp,n)
iter = 0; ea = 0; xn = x0;
fd = inline(char(diff(formula(f))),'x');
```

```
disp(' ----------------------------------------');
disp(' No     Root     f(Root)     %error ');
disp(' ----------------------------------------');
while (1)
  xnold = xn;
  xn =x0*exp(-f(x0)/(x0*fd(x0)));
  xnnew = xn;
  disp(sprintf('%4d %10.4f %10.2f %8.2f',iter+1,xn,f(xn),ea));
  iter = iter + 1;
  if xn  = 0, ea = abs((xn - xnold)/xn) * 100; end
  x0 = xn;
  if ea <= esp | iter >= n, break, end
end
disp(' ----------------------------------------');
disp(['Given function f(x) = ' char(f)]);
disp(sprintf('Approximate root = %10.10f',xn));
```

The following a data type `ExpNewton(f,x0,n)` gives the implementation in Maple, where `f` is given non-linear transcendental function, `x0` is the initial approximation of the root, and `n` is the number of iterations required.

```
ExpNewton:=proc(f,x0,n)
local iten, fx0;
for iten from 1 by 1 while iten < n+1
  do
    printf("Iteration %g :  ", iten);
    x0:=(x0*exp(-subs(x=x0,f)/(x0*subs(x=x0,diff(f,x)))));
    fx0:=subs(x = x0, f);
  end do;
return x0,fx0;
end proc:
```

Sample computations using the implementation of the proposed algorithm are presented in Section 4.

4 Numerical Examples

This section provides some numerical examples to discuss the algorithm presented in Section 2 and comparisons are taken into account to conform that the algorithm is more efficient than other existing methods.

Example 4.1. Consider the following transcendental equations [11]. We compare the number of iterations required to get approximation root with accuracy of 10^{-15}. The numerical results are provided in Table 1.

a. $f(x) = \ln(x)$, with initial approximation 0.5.

b. $f(x) = x - e^{\sin(x)} + 1$, with initial approximation 4.

c. $f(x) = 11x^{11} - 1$, with initial approximation 1.

d. $f(x) = xe^{-x} - 0.1$, with initial approximation 0.1.

The numerical results given in Table 1 shows that the proposed method is more efficient than other methods.

Example 4.2. Consider a transcendental equations of the following type. We find approximate root using formulae given in equations (2.2), (2.3) and (2.4) to show the convergence of the proposed algorithm, see Table 2. To find approximate root, we start with an initial approximation

Table 1. Comparing No. of iterations by different methods

Fun.	Exact Root	Regula Falsi method	Newton Raphson method	Steffen method	Proposed method
$a.$	1.00000	27	Divergent	Failure	2
$b.$	1.69681 & 0	32	Not Convergent	Failure	3
$c.$	0.80413	101	7	Divergent	6
$d.$	0.11183	15	Failure	Failure	2

as 1.5, and we have the exact real root is 1.134724138.

$$f(x) = x^6 - x - 1. \tag{4.1}$$

Table 2. Comparing approximate root using formulae given in equations (2.2),(2.3),(2.4)

Iteration No.	Newton-Raphson method (2.2)	Equation No. (2.3)	Equation No. (2.4)	Proposed method
1	1.300490884	1.313758847	1.313170607	1.313189657
2	1.181480417	1.193998307	1.193487993	1.193502766
3	1.139455590	1.143246378	1.143095302	1.143099361
4	1.134777625	1.134926557	1.134919460	1.134919647
5	1.134724145	1.134724255	1.134724248	1.134724248
6	1.134724138	1.134724138	1.134724138	1.134724138
7	1.134724138	1.134724138	1.134724138	1.134724138

Example 4.3. This example gives the sample computation using MatLab and Maple implementation as described in Section 3. Consider a transcendental equation of the form

$$f(x) = e^{-x} - x$$

with initial approximation of the root as 1.0.

Using MatLab implementation, we have the following computations.

```
f=inline('exp(-x) - x','x');
function root = ExpNewton(f,1.0,0.00001,10)
-----------------------------------------------------
 No      Root              f(Root)          %error
-----------------------------------------------------
 1    0.6299485325     -0.0973293196     --
 2    0.5695393922     -0.37534081e-2    10.60666586
 3    0.5671472898     -0.62676e-5       0.4217779832
 4    0.5671432906     -3e-10            0.0007051480758
 5    0.5671432904      0                3.526445668*10^-8
 6    0.5671432904      0                0
-----------------------------------------------------
Given function f(x) = exp(-x) - x
Approximate root = 0.5671432904
```

Using Maple implementation, we have the following computations.

```
> f:= exp(-x) - x:
```

```
> ExpNewton(f,1.0,7);
```

```
Iteration 1 :        0.6299485325
Iteration 2 :        0.5695393922
Iteration 3 :        0.5671472898
Iteration 4 :        0.5671432906
Iteration 5 :        0.5671432904
Iteration 6 :        0.5671432904
Iteration 7 :        0.5671432904
```

One can use the implementation of the proposed algorithm to speed up the manual calculations.

5 Conclusion

In this present work, we presented a new algorithm to compute an approximate root of a given transcendental function better than previous existing methods as illustrated. The proposed new algorithm was based on exponential function having better convergence than previous existing methods. This proposed algorithm is useful for solving the complex real life problems. One can also extend the proposed algorithm to system of non-linear equations. Implementation of the proposed algorithm in Matlab and Maple is also discussed.

References

[1] C. Gutierrez, F. Gutierrez, M.-C. Rivara: Complexity of the bisection method, Theoretical Computer Science, **382** (2007), 131–138.

[2] D. Bachrathy, G. Stépán: Bisection method in higher dimensions and the efficiency number, Mechanical Engineering, **56 (2)** (2012), 81–86.

[3] Dowell-Jarrat: A modified Regula-Falsi method for computing the real root of an equation. BIT Numerical Mathematics, **11** (1971), 168–174.

[4] E. Novak, K. Ritter, And H. Wozniakowski: Mathematics of computation, 64, (212), 1995, 1517–1539.

[5] Eiger, K. Sikorski, F. Stenger: A Bisection Method for Systems of Nonlinear Equations, ACM Transactions on Mathematical Softwares, **10 (4)** (1984), 367–377.

[6] G.R. Wood: The bisection method in higher dimensions, Mathematical Programming, 55 (1992) 319–337.

[7] J.C.Yakoubsohn: Numerical analysis of a bisection-exclusion method to find zeros of univariate analytic functions, Journal of Complexity, **21** (2005), 652–690.

[8] Johan and Ronald. The Newton-Raphson Method. International Journal of Mathematical education in Science and Technology, **26 (2)** (1995), 177–193.

[9] J. R. SHARMA and R. K. GOYAL: Fourth-order derivative-free methods for solving non-linear equations, International Journal of Computer Mathematics, **83 (1)** (2006), 101–106.

[10] J. Chen, W. Li: An exponential regula falsi method for solving nonlinear equations, Numerical Algorithms, **41** (2006), 327–338.

[11] J. Chen, W. Li: An improved exponential regula falsi methods with quadratic convergence of both diameter and point for solving nonlinear equations, Applied Numerical Mathematics, **57** (2007), 80–88.

[12] Jinhai Chen: New modified regula falsi method for nonlinear equations, Applied Mathematics and Computation, **184** (2007), 965–971.

[13] John Abbott: Quadratic interval refinement for real roots, ACM communications in computer algebra, **48 (1)**, (2014) 1–8.

[14] M.N. Vrahatis, K.I. Iordanidis: A Rapid Generalized Method of Bisection for Solving System of Nonlinear Equation, Numer. Math, **49** (1986), 123–138.

[15] Mamta, V. Kanwar, V.K. Kukreja, Sukhjit Singh: On a class of quadratically convergent iteration formulae, Applied Mathematics and Computation, **166** (2005), 633–637.

[16] Mamta, V. Kanwar, V.K. Kukreja, Sukhjit Singh: On some third-order iterative methods for solving nonlinear equations, Applied Mathematics and Computation, **171** (2005), 272–280.

[17] M. A. Noor, K. I. Noor, W. A. Khan, F. Ahmad: On iterative methods for nonlinear equations, Applied Mathematics and Computation, **183** (2006), 128–133.

[18] M. Aslam Noor, F. Ahmad: Numerical comparison of iterative methods for solving nonlinear equations, Applied Mathematics and Computation, **180** (2006), 167–172.

[19] M. A. Noor, K. I. Noor: Three-step iterative methods for nonlinear equations, Applied Mathematics and Computation, **183** (2006), 322–327.

[20] S. Hussain, V. K. Srivastav, S. Thota: Assessment of Interpolation Methods for Solving the Real Life Problem, International Journal of Mathematical Sciences and Applications, **5 (1)** (2015), 91–95.

[21] S. Thota, V. K. Srivastav: Quadratically Convergent Algorithm for Computing Real Root of Non-Linear Transcendental Equations, BMC Research Notes, (2018) **11:909**.

[22] S. Thota, V. K. Srivastav: Interpolation based Hybrid Algorithm for Computing Real Root of Non-Linear Transcendental Functions, International Journal of Mathematics and Computer Research, **2 (11)** (2014), 729–735.

[23] Saied and Liao: A new modification of False-Position method based on homotopy analysis method. Applied Mathematics and Mechanics, **29 (2)**, 223–228.

[24] Sagraloff and Mehlhorn: Computing real roots of real polynomials. 2013.

[25] T. Gemechu: Some Root Finding With Extensions to Higher Dimensions, Mathematical Theory and Modeling, **7 (4)** (2017), 1–13.

[26] V. K. Srivastav, S. Thota, M. Kumar: A New Trigonometrical Algorithm for Computing Real Root of Non-linear Transcendental Equations, International Journal of Applied and Computational Mathematics, (2019) **5:44**.

[27] X. Wu: Improved Muller method and Bisection method with global and asymptotic superlinear convergence of both point and interval for solving nonlinear equations, Applied Mathematics and Computation, **166** (2005), 299–311.

Author information

Srinivasarao Thota, Tekle Gemechu, Department of Applied Mathematics, School of Applied Natural Sciences, Adama Science and Technology University, Post Box No. 1888, Adama, Ethiopia.
E-mail: srinithota@ymail.com, srinivasarao.thota@astu.edu.et, tekgem@yahoo.com

P. Shanmugasundaram, Department of Mathematics, College of Natural & Computational sciences, Mizan Tepi University, Mizan Tepi, Ethiopia.
E-mail: psserode@gmail.com

Received: April 29, 2019.
Accepted: November 23, 2019.