URAL MATHEMATICAL JOURNAL, Vol. 9, No. 1, 2023, pp. 176–186

DOI: 10.15826/umj.2023.1.016

ON NEW HYBRID ROOT-FINDING ALGORITHMS FOR SOLVING TRANSCENDENTAL EQUATIONS USING EXPONENTIAL AND HALLEY'S METHODS[1]

Srinivasarao Thota

Department of Mathematics, Amrita School of Physical Sciences,
Amrita Vishwa Vidyapeetham, Amaravati, Andhra Pradesh–522503, India

srinithota@ymail.com

Tekle Gemechu

Department of Applied Mathematics, School of Applied Natural Sciences,
Adama Science and Technology University,
Post Box No. 1888, Adama, Ethiopia

tekgem@yahoo.com

Abayomi Ayotunde Ayoade

Department of Mathematics, University of Lagos,
Lagos State, Nigeria

ayoayoade@unilag.edu.ng

Abstract: The objective of this paper is to propose two new hybrid root finding algorithms for solving transcendental equations. The proposed algorithms are based on the well-known root finding methods namely the Halley's method, regula-falsi method and exponential method. We show using numerical examples that the proposed algorithms converge faster than other related methods. The first hybrid algorithm consists of regula-falsi method and exponential method (RF-EXP). In the second hybrid algorithm, we use regula-falsi method and Halley's method (RF-Halley). Several numerical examples are presented to illustrate the proposed algorithms, and comparison of these algorithms with other existing methods are presented to show the efficiency and accuracy. The implementation of the proposed algorithms is presented in Microsoft Excel (MS Excel) and the mathematical software tool Maple.

Keywords: Hybrid method, Halley's method, Regula-falsi method, Transcendental equations, Root-finding algorithms.

1. Introduction

The applications of nonlinear equations of the type $f(x) = 0$ arise in various branches of pure and applied sciences, such as computer science, chemical engineering, physics, etc. Getting the root of transcendental equations is of great importance. In recent time, several scientists and engineers have focused to solve nonlinear equations numerically as well as analytically. There are several iterative (hybrid) methods/algorithms available in the literature that are derived from various methods, see, for example [1–3, 10, 11, 15, 17–26]. In general, the roots of nonlinear or transcendental equations cannot be expressed in closed form or cannot be computed analytically.

[1]Corresponding Authors email: t_srinivasarao@av.amrita.edu

Root finding algorithms allow us to compute approximations to roots, these approximations are expressed as either as small isolating intervals or as floating point numbers. The concept of creating hybrid methods, combining two or more classic approaches is not new and has a long history. One of the oldest hybrid root-finding method is Dekker's method, see for example [8], introduced in 1969. The main idea of this method is the combination of the classical methods i.e. bisection method and secant method. Using the idea of the Dekker's method, Richard P. Brent proposed a new hybrid root-finding method in 1973, see for example [5], which is based on the bisection method, the secant method and inverse quadratic interpolation. Since the Brent's method uses the idea of the Dekker's method, the method is also known as the Brent-Dekker method. In 1979, Ridders proposed a root-finding algorithm [13, 16] which is simpler than Brent's method and Dekker's method. This algorithm is based on the regula-falsi method and the exponential function. Badr et al. [1] proposed two hybrid algorithms. The first hybrid algorithm is based on the false-position method and the modified secant method (FP-MSe), and the second algorithm is based on the false-position method and the trigonometric secant method (FP-TMSe). Novak et al. [14] proposed a hybrid secant-bisection approach in 1995. Sabharwal [17] proposed a new hybrid method that combines two bracketing techniques (bisection-false position). Badr et al. [2], on the other hand, created a hybrid algorithm that combines two closed algorithms (trisection-false position). They tested their strategy on fifteen nonlinear and linear equations as a benchmark. They came to the conclusion that their algorithm outperformed Sabharwal's.

In this paper, we develop two new hybrid root finding algorithms for solving transcendental equations. These algorithms are created using the well-known root finding methods, namely the Halley's method, regula-falsi method and exponential method. Using numerical examples, we show that the proposed algorithms converge faster than the other related methods. The main idea of the first hybrid algorithm is based on the regula-falsi method and the exponential method (RF-EXP) and the second hybrid algorithm is based on the regula-falsi method and the Halley's method (RF-Halley). Several numerical examples are presented to illustrate the proposed algorithms. The comparisons are made to compare the results of calculations using the proposed algorithms with other existing methods to show efficiency and accuracy. Implementation of the proposed algorithms is presented in MS Excel and Maple.

The rest of the paper is organized as follows: in Section 2, we present two new hybrid root-finding algorithms with methodology and steps involving in the proposed algorithms; Section 3 discusses the analysis of convergence; Section 4 presents several numerical examples to illustrate and validate the proposed methods/algorithms; and finally Section 5 presents the implementation of the proposed algorithms in MS Excel and of the mathematical software tool Maple with examples of computations.

2. New hybrid algorithms

In this section, we present two blended root-finding algorithms. These algorithms have the advantages of open methods (fast) and bracketing method (convergent).

2.1. New hybrid Algorithm 1 (regula falsi-exponential algorithm)

In this section, we present a new hybrid algorithm using the regula-falsi method and the exponential method (RF-EXP). The regula-false method guarantees the existence of the root, while the exponential method gives faster convergence. The iterative formula used in exponential method is as follows, more details about this method can be found in [23]

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

In regula-falsi method [4, 7, 9, 12], we take two initial guesses, say a and b, such that $f(a)f(b) < 0$. The approximate root is calculated by finding the point of intersection of the straight line joining the points $(a, f(a))$ and $(b, f(b))$ with the x-axis. Hence the approximate root can be calculated using the formula

$$x_r = a - \frac{f(a)(b-a)}{f(b) - f(a)}. \tag{2.2}$$

Now, we have to choose the appropriate interval to compute the second iteration. We have the following possible cases:

1. If $f(a)f(x_r) < 0$, then the root exists in $[a, x_r]$, and we set $b = x_r$ to find the second iteration using formula (2.2).

2. If $f(a)f(x_r) > 0$, then the root exists in $[x_r, b]$, and we set $a = x_r$ to find the second approximate root using (2.2).

3. If $f(x_r) = 0$, then the required root is x_r and we terminate the process.

Algorithm 1. In this algorithm, we have:
the input: the function $f(x)$, the interval $[a, b]$ where the exact root lies in, the absolute error *eps*, the number of iterations n; the output: the approximate root x, the function value $f(x)$. The steps of the algorithm are as follows:

1. $i = 0$
2. **while** $i! = n$ **do**
3. $\quad i = i + 1$;
4. $\quad x_{rf} = a - \dfrac{f(a)(b-a)}{f(b) - f(a)}$
5. $\quad x_i = x_{rf} \exp\left(\dfrac{-f(x_{rf})}{x_{rf} f'(x_{rf})}\right)$
6. $\quad$ **if** $|a - x_i| \leq eps$ **then**
7. $\quad\quad$ **return** x_i, $f(x_i)$ **break**;
8. $\quad$ **else if** $f(x_i) * f(a) < 0$ **then** $b = x_i$
9. $\quad$ **else** $a = x_i$
10. $\quad$ **end (if)**
11. **end (while)**

In section 4, we present several examples to illustrate this algorithm and to show its efficiency.

2.2. New hybrid Algorithm 2 (regula falsi-Halley algorithm)

In this section, we present another new hybrid algorithm using regula-falsi method and Halley's method (RF-Halley). Similar to the previous new algorithm, the regula-false method guarantees the existence of the root and the Halley's method gives the fast convergence. The Halley's method is invented by Edmond Halley. In this method, we need one initial approximation as in the Newton's method with a continuous second derivative, and this method produces a sequence of approximations to the root. We compute the sequence of iterations using the Halley's method formula [6, 7] as

$$x_{n+1} = x_n - \frac{2f(x_n)f'(x_n)}{2[f'(x_n)]^2 - f(x_n)f''(x_n)}$$

with an initial approximation x_0.

Algorithm 2. In this algorithm, we have: the input: the function $f(x)$, the interval $[a, b]$ where the exact root lies in, the absolute error *eps*, the number of iterations n; the output: the approximate root x, the function value $f(x)$. The steps of the algorithm are as follows:

1. $i = 0$
2. **while** $i! = n$ **do**
3. $\quad i = i + 1;$
4. $\quad x_{rf} = a - \dfrac{f(a)(b-a)}{f(b) - f(a)}$
5. $\quad x_i = x_{rf} - \dfrac{2f(x_{rf})f'(x_{rf})}{2[f'(x_{rf})]^2 - f(x_{rf})f''(x_{rf})}$
6. $\quad$ **if** $|a - x_i| \leq eps$ **then**
7. $\quad\quad$ **return** x_i, $f(x_i)$ **break**;
8. $\quad$ **else if** $f(x_i) * f(a) < 0$ **then** $b = x_i$
9. $\quad$ **else** $a = x_i$
10. $\quad$ **end (if)**
11. **end (while)**

2.3. Flow-diagrams

In this section, we present the flow diagrams of the proposed algorithms. In Fig. 1, we present the flow diagram of Algorithm 1 and the Fig. 2 presents the flow diagram of Algorithm 2.

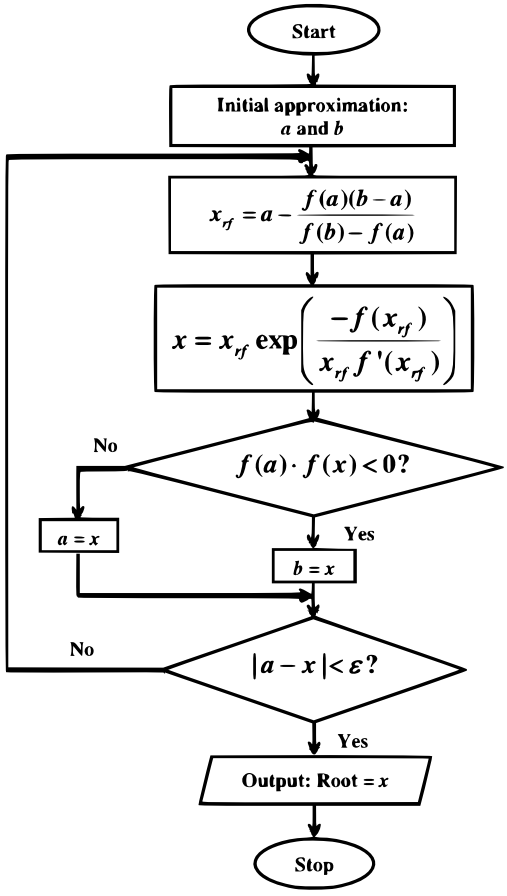

Figure 1. Flow-diagram of Algorithm 1.

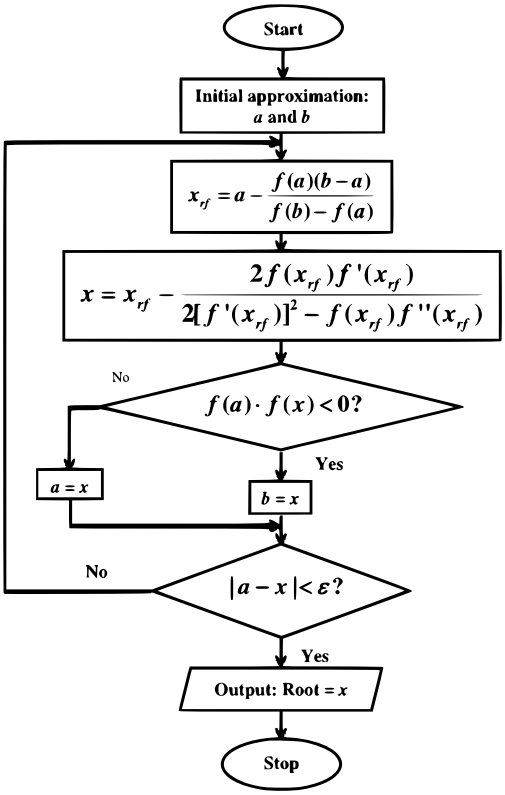

Figure 2. Flow-diagram of Algorithm 2.

In Section 4, we present several examples to illustrate this algorithm and to show the efficiency of the algorithm.

3. Convergence analysis

The main idea of the proposed algorithms is the combination of the open methods and the closed methods. Hence, the proposed algorithms converge to the approximate root faster. On the other hand, when the open methods (exponential method or Halley's method) fail, the closed method (false position) continues to get the next approximations, so the proposed algorithms RF-EXP and RF-Halley are convergent faster with a guaranteed root.

4. Numerical examples

In this section, we present several numerical examples to illustrate the proposed algorithms, and comparisons are made to confirm that the proposed algorithms give a solution faster than some existing methods. The following Example 1 and Example 2 illustrate the proposed algorithms.

Example 1. Consider the nonlinear equation

$$e^{-x} - x = 0, \tag{4.1}$$

with the initial approximations $a = 0$ and $b = 1$. Following the proposed Algorithm 1, we have

$$x_{rf} = 0.612699837,$$
$$x_1 = 0.568452077, \quad f(x_1) = -0.00205057.$$

Now, we verify the possible three conditions given in regula-falsi method, and we get the required interval where the exact root lies in, as [0, 0.568452077]. Repeat the process for higher iterations, and we have

$$\begin{gathered} x_{rf} = 0.567288811, \\ x_2 = 0.567143305, \quad f(x_2) = -2.32442 \times 10^{-8}, \\ x_{rf} = 0.567143292, \\ x_3 = 0.56714329, \quad f(x_3) \approxeq 0. \end{gathered}$$

The nonlinear equation in (4.1) is solved using well known methods to compare the results (see Table 1) with the proposed Algorithm 1 up to 8 correct decimal places.

In Table 1, BM, RFM, NRM, Halley, Steffensen and PA indicate the bisection method, regula-falsi method, Newton-Raphson method, Halley's method, Steffensen's method and the proposed algorithm (PA) respectively.

Table 1. Numerical results and comparisons.

BM	RFM	NRM	Halley	Steffensen	PA
24	8	4	3	4	3

Example 2. We apply the Algorithm 2 to the function

$$f(x) = e^x - 3x - 2$$

with initial approximations $a = 2$ and $b = 3$. Following the proposed Algorithm 2 similar to Example 1, we have

$$\begin{gathered} x_{rf} = 2.063006766, \\ x_1 = 2.12530056, \quad f(x_1) = -0.000487257. \end{gathered}$$

Now, we verify the possible three conditions given in regula-falsi method, and we get the required interval where the exact root lies in, as $[2.12530056, 3]$. Repeat the process for higher iterations. So we have

$$\begin{gathered} x_{rf} = 2.125347467, \\ x_2 = 2.1253911988111, \quad f(x_2) = -1.56319 \times 10^{-13}, \\ x_{rf} = 2.12539119881112, \\ x_3 = 2.12539119881113, \quad f(x_3) \approxeq 0. \end{gathered}$$

Example 3. In this example, we present a comparison between various existing methods and the proposed algorithms to show the efficiency and simplicity of the proposed algorithms in computation of a root. Consider ten standard nonlinear equations given in Table 2.

Using various numerical existing methods, we compute the roots of the ten equations given in Table 2 to ten decimal places. In Table 3, we present the number of iterations required to obtain

Table 2. Ten nonlinear equations for comparison with various methods.

S.No.	Equation	Initial approximations	Exact root
Eq.1	$e^x - 3x - 2 = 0$	$a = 2, \quad b = 3$	2.1253911988
Eq.2	$x - \cos x = 0$	$a = 0, \quad b = 1$	0.7390851332
Eq.3	$e^{-x} - x = 0$	$a = 0, \quad b = 1$	0.5671432904
Eq.4	$x^2 - 5$	$a = 2, \quad b = 7$	2.2360679775
Eq.5	$x^2 + e^{x/2} - 5$	$a = 1, \quad b = 2$	1.6490132683
Eq.6	$\sin x - x^2$	$a = 0.5, \quad b = 1$	0.8767262154
Eq.7	$x^2 - e^x - 3x + 2$	$a = 0, \quad b = 1$	0.2575302854
Eq.8	$x^3 - 10$	$a = 2, \quad b = 3$	2.154434690
Eq.9	$xe^{-x} - 0.1$	$a = 0, \quad b = 1$	0.1118325592
Eq.10	$\cos x - x$	$a = 0, \quad b = 1$	0.7390851332

Table 3. Numerical results and comparisons.

S.No.	BM	RFM	NRM	Halley	Steffensen	EXP	TRIG	Alg.1	Alg.2
Eq.1	29	25	6	4	5	4	4	3	3
Eq.2	31	9	5	3	5	3	3	3	2
Eq.3	32	11	4	3	4	5	4	3	2
Eq.4	32	28	4	3	6	3	5	3	2
Eq.5	30	10	5	4	5	5	6	4	2
Eq.6	30	12	5	4	4	4	5	4	2
Eq.7	32	7	4	4	4	3	4	3	1
Eq.8	29	17	5	3	21	3	4	3	1
Eq.9	34	12	5	8	6	5	5	2	2
Eq.10	31	9	4	3	5	4	4	3	1

the required root; and the terms BM, RFM, NRF, Halley, Steffensen, EXP, TRIG, Alg. 1 and Alg. 2 indicate bisection method, regula-falsi method, Newton-Raphson method, Helley's method, Steffensen's method, exponential method [23], trigonometric method [18], proposed Algorithm 1 and proposed Algorithm 2 respectively.

From Table 3, one can observe that the proposed algorithms required less number of iterations by comparing with other existing methods.

5. Implementation

In this section, we discuss the implementation of the proposed algorithms in MS Excel and Maple. We can also implement these algorithms in other mathematical software tools such as MATLAB, SCIlab, Mathematica, Singular etc.

5.1. Implementation in MS Excel

The proposed algorithms can be computed in Excel as follows. The number of iterations r, initial guesses x_l, x_u and $f(x_l)$, $f(x_u)$, x_{r1}, $f(x_{r1})$, $f'(x_{r1})$, $f''(x_{r1})$, x_r, $f(x_r)$ are entered in the MS Excel cells, for example, at A5, B5, C5, D5, E5, F5, G5, H5, I5, J5, K5 respectively. Enter the respective values in 6th row, i.e., $r = 1$, x_l, x_u and "=f(B6)", "=f(C6)", "=(B6*E6-C6*D6)/(E6-D6)", "=f(F6)", "=f'(F6)", "=f"(F6)". Now the first estimated root using Algorithm 1 is obtained by entering the formula in J6 as "=F6*EXP((-G6)/(F6*H6))"; and the first estimated root using Algorithm 2 is obtained by entering the formula in J6 as "=F6-((2*G6*H6)/((2*(H6)^2)-(G6*I6)))", where F6 in both algorithms is obtained using "=(B6*E6-C6*D6)/(E6-D6)". In the last column K6, we check the function value at the estimated root $f(x_r)$ as "=f(J6)". For second iteration, we need to check the three conditions in the method and the entries of 18th row of the excel sheet are as follows. The iteration r is entered with "=A6+1" in A18. The important steps in this algorithm (selection of appropriate sub-interval for next iterations) are entered in B7 and C7 with commands "=IF(D6*K6<0,B6,J6)" and "=IF(E6*K6<0, C6,J6)" respectively. The last columns, D6–K6 are drag down for next iteration value. Finally, drag down the entire 7th row until the required number of iterations.

Sample computations using MS Excel

Consider the function $f(x) = e^{-x} - x$ with initial approximations $x_l = 0$ and $x_u = 1$ and we have $f'(x) = -e^{-x} - 1$, $f''(x) = e^{-x}$. Now following the procedure given in Section 5.1, we have computations using Algorithm 1 as in Table 4 and computations using Algorithm 2 as in Table 5.

Table 4. Proposed Algorithm 1 in Excel.

r	x_l	x_u	$f(x_l)$	$f(x_u)$	x_{r1}	$f(x_{r1})$	$f'(x_{r1})$	xr	$f(x_{r1})$
1	0	1	1	-0.6321	0.6127	-0.07081	-1.5419	0.5671	-0.0021
2	0	0.5685	1	-0.0021	0.5673	-2.99E-04	-1.5671	0.5671	-2.3E-08
3	0	0.5671	1	-2.3E-08	0.5671	-2.59E-09	-1.5671	0.5671	0

Table 5. Proposed Algorithm 2 in Excel.

r	x_l	x_u	$f(x_l)$	$f(x_u)$	x_{r1}	$f(x_{r1})$	$f'(x_{r1})$	$f''(x_{r1})$	x_r	$f(x_{r1})$
1	0	1	1	-0.6321	0.6127	-0.07081	-1.5419	0.54189	0.5671	4.1E-06
2	0.5671	1	4.1E-06	-0.6321	0.5671	-2.99E-07	-1.5671	0.5671	0.5671	0

5.2. Maple implementation

In this section, we present the maple implementation of the proposed algorithms with sample computations as follows.

Algorithm 1 in Maple

```
RFEXP := proc (a, b, Eq, eps, n)
```

```
local a1, b1, f, c, i, c1;
i := 0;
a1 := evalf(a);
b1 := evalf(b);
f := unapply(lhs(Eq), x);
if f(a1) = 0 then
return a1
else if f(b1) = 0 then
return b1
else if 0 < f(a1)*f(b1) then
error "Should be f(a)*f(b)<0"
end if;
end if;
end if;
do
c1 := (a1*f(b1)-b1*f(a1))/(f(b1)-f(a1));
c := c1*exp(-f(c1)/(c1*(D(f))(c1)));
i := i+1;
if f(c) = 0 or |c-a1| < eps or i = n then
return c
else if f(a1)*f(c) < 0 then
b1 := c
else a1 := c
end if;
end if;
printf("Iteration %g : x = %g \n", i, c)
end do
end proc
```

Algorithm 2 in Maple

```
RFHalley := proc (a, b, Eq, eps, n)
local a1, b1, f, c, i, c1;
i := 0;
a1 := evalf(a);
b1 := evalf(b);
f := unapply(lhs(Eq), x);
if f(a1) = 0 then
return a1
else if f(b1) = 0 then
return b1
else if 0 < f(a1)*f(b1) then
error "Should be f(a)*f(b)<0"
end if;
end if;
end if;
do
c1 := (a1*f(b1)-b1*f(a1))/(f(b1)-f(a1));
c := c1-(2*f(c1)*f'(c1)/(2*f'(c1)^2-f(c1)*(f''(c1))))
```

```
i := i+1;
if f(c) = 0 or |c-a1| < eps or i = n then
return c
else if f(a1)*f(c) < 0 then
b1 := c
else a1 := c
end if;
end if;
printf("Iteration %g : x = %g \n", i, c)
end do
end proc
```

Sample computations using Maple

Consider a function $f(x) = x - \cos x$ with initial conditions $a = 0$ and $b = 1$ with $\epsilon = 10^{-10}$. Now applying the maple implementation, we have the following computations using Algorithm 1 and Algorithm 2.

```
> RFEXP(0, 1, x-cos(x) = 0, 10^(-10), 10);
```

$$\begin{aligned} Iteration\ 1 &: x = 0.742009 \\ Iteration\ 2 &: x = 0.739086 \\ Iteration\ 3 &: x = 0.739085 \\ & 0.7390851332 \end{aligned}$$

```
> RFHalley(0, 1, x-cos(x) = 0, 10^(-10), 10);
```

$$\begin{aligned} Iteration\ 1 &: x = 0.739066 \\ & 0.7390851332 \end{aligned}$$

6. Conclusion

In this paper, we propose two hybrid root finding algorithms to solve the given transcendental equations. The algorithms are based on the Halley's method, regula-falsi method and exponential method. Several numerical examples are presented to illustrate the proposed algorithms. The first hybrid algorithm consists of regula-falsi method and exponential method, and the second hybrid algorithm consists of regula-falsi method and Halley's method. MS Excel and Maple implementation of the proposed algorithms are presented with sample computations. One can implement these algorithms in other software tools such as Matlab, SCIlab, Mathematica etc. The proposed algorithms perform faster than some existing methods.

Acknowledgments

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

REFERENCES

1. Badr E., Attiya H., El Ghamry A. Novel hybrid algorithms for root determining using advantages of open methods and bracketing methods. *Alexandria Eng. J.*, 2022. Vol. 61, No. 12. P. 11579–11588. DOI: 10.1016/j.aej.2022.05.007

2. Badr E., Almotairi S., El Ghamry A. A Comparative study among new hybrid root finding algorithms and traditional methods. *Mathematics*, 2021. Vol. 9, No. 11. Art. no. 1306. DOI: 10.3390/math9111306
3. Badr E. M., ElGendy H. S. A hybrid water cycle-particle swarm optimization for solving the fuzzy underground water confined steady flow. *Indones. J. Electr. Eng. Comput. Sci.*, 2020. Vol. 19, No. 1. P. 492–504. DOI: 10.11591/ijeecs.v19.i1.pp492-504
4. Baskar S., Ganesh S. S. *Introduction to Numerical Analysis.* Powai, Mumbai, India: Depart. Math., Indian Inst. Tech. Bombay, 2016. 230 p.
5. Brent R. P. *Algorithms for Minimization without Derivatives.* Englewood Cliffs, NJ: Prentice-Hall, 1973. 195 p.
6. Burden R. L., J. Douglas Faires. *Numerical Analysis*, 3rd ed. Baston, USA: PWS Publishing, 1985. 676 p.
7. Chapra S. C., Canale R. P. *Numerical Methods for Engineers*, 7th ed. Boston, MA, USA: McGraw-Hill, 2015. 970 p.
8. Dekker T. J. Finding a zero by means of successive linear interpolation. In: *Constructive Aspects of the Fundamental Theorem of Algebra, B. Dejon, P. Henrici (eds.).* London: Wiley-Interscience, 1969. P. 37–48.
9. Fink K. D., Mathews J. H. *Numerical Methods Using Matlab*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall Inc., 2004. 696 p.
10. Gemechu T., Thota S. On new root finding algorithms for solving nonlinear transcendental equations. *Int. J. Chem., Math. Phys.*, 2020. Vol. 4, No. 2. P. 18–24. DOI: 10.22161/ijcmp.4.2.1
11. Hasan A. Numerical study of some iterative methods for solving nonlinear equations. *Int. J. Eng. Sci. Invent.*, 2016. Vol. 5, No. 2. P. 1–10.
12. Hoffman J. D. *Numerical Methods for Engineers and Scientists*, 2nd ed. NY, Basel: Marcel Dekker Inc., 2001. 823 p.
13. Kiusalaas J. *Numerical Methods in Engineering with Python,* 2nd ed. Cambridge: Cambridge University Press, 2010. 432 p.
14. Novak E., Ritter K., Woźniakowski H. Average-case optimality of a hybrid secant-bisection method. *Math. Comput.*, 1995. Vol. 64, No. 212. P. 1517–1539. DOI: 10.2307/2153369
15. Parveen T., Singh S., Thota S., Srivastav V. K. A new hybrid root-finding algorithm for transcendental equations using bisection, regula-falsi and Newton-Raphson methods. In: *National Conf. Sustainable & Recent Innovation in Science and Engineering (SUNRISE-19)*, 2019.
16. Ridders C. A new algorithm for computing a single root of a real continuous function. *IEEE Trans. Circuits Syst.*, 1979. Vol. 26, No. 11. P. 979–980. DOI: 10.1109/TCS.1979.1084580
17. Sabharwal C. L. Blended root finding algorithm outperforms bisection and regula falsi algorithms. *Mathematics*, 2019. Vol. 7, No. 11. Art. no. 1118. DOI: 10.3390/math7111118
18. Srivastav V. K., Thota S., Kumar M. A new trigonometrical algorithm for computing real root of non-linear transcendental equations. *Int. J. Appl. Comput. Math.*, 2019. Vol. 5. Art. no. 44. DOI: 10.1007/s40819-019-0600-8
19. Thota S., Srivastav V. K. An algorithm to compute real root of transcendental equations using hyperbolic tangent function. *Int. J. Open Problems Compt. Math.*, 2021. Vol. 14, No. 2. P. 1–14.
20. Thota S. A numerical algorithm to find a root of non-linear equations using householder's method. *Int. J. Adv. Appl. Sci.*, 2021. Vol. 10, No. 2. P. 141–148. DOI: 10.11591/ijaas.v10.i2.pp141-148
21. Thota S., Gemechu T., Shanmugasundaram P. New algorithms for computing a root of non-linear equations using exponential series. *Palestine J. Math.*, 2021. Vol. 10, No. 1. P. 128–134.
22. Thota S., Gemechu T. A new algorithm for computing a root of transcendental equations using series expansion. *Southeast Asian J. Sci.*, 2019. Vol. 7, No. 2. P. 106–114.
23. Thota S. A new root-finding algorithm using exponential series. *Ural Math. J.*, 2019. Vol. 5, No. 1. P. 83–90. DOI: 10.15826/umj.2019.1.008
24. Thota S., Srivastav V. K. Quadratically convergent algorithm for computing real root of non-linear transcendental equations. *BMC Res. Notes*, 2018. Vol. 11. Art. no. 909. DOI: 10.1186/s13104-018-4008-z
25. Thota S., Srivastav V. K. Interpolation based hybrid algorithm for computing real root of non-linear transcendental functions. *Int. J. Math. Comp. Res.*, 2014. Vol. 2, No. 11. P. 729–735.
26. Thota S. A new hybrid halley-false position type root finding algorithm to solve transcendental equations. In: *Istanbul International Modern Scientific Research Congress-III, 06–08 May 2022.* Istanbul, Turkey: Istanbul Gedik University, 2022. P. 1–2.