JOURNAL OF NUMERICAL ANALYSIS AND APPROXIMATION THEORY
J. Numer. Anal. Approx. Theory, vol. 49 (2020) no. 2, pp. 177–189
ictp.acad.ro/jnaat

BASIN ATTRACTORS FOR DERIVATIVE-FREE METHODS TO FIND SIMPLE ROOTS OF NONLINEAR EQUATIONS

BENY NETA*

Abstract. Numerous methods exist for finding zeros of nonlinear equations. Several of these schemes are derivative-free. One of the oldest is the secant method where the derivative is replaced by a divided difference. Clearly such method will need an additional starting value. Here we study the dynamics of several derivative-free methods and compare them using the idea of basin of attraction. As a results, we develop a new high-order derivative-free method and study its dynamics.

2020 Mathematics Subject Classification. 65H05.
Keywords. Basin of Attraction, Derivative-free methods, simple roots, nonlinear equations.

1. INTRODUCTION

In engineering and applied science we encounter the problem of solving a nonlinear equation $f(x) = 0$. Most numerical solution techniques are based on Newton's method, *i.e.*, starting with an initial guess x_0 for the root ξ, we create a sequence $\{x_n\}$

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

The convergence is quadratic, that is

$$|x_{n+1} - \xi| \leq C_2 |x_n - \xi|^2$$

and 2 is the largest with such property. In general, we say that a method is of order p if

$$|x_{n+1} - \xi| \leq C_p |x_n - \xi|^p$$

and p is the largest with such property.

The vast literature (see Traub [20] and the more recent book by Petković *et al.* [12]) for the numerical solution of a single nonlinear equation include relatively few derivative-free methods. The methods are multistep with Steffensen's method [17] as first step. See, Khattri and Steihaug [7], Soleymani and Shateyi [15], Soleymani and Vanani [16], Thukral [19], Zheng *et al.* [23],

*Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA 93943; e-mail: bneta@nps.edu, Tel.: 1 (831) 656-2235, Fax: 1 (831) 656-2355.

Veiseh *et al.* [21], Sharma and Goyal [14], Chicharro *et al.* [1], Cordero and Torregrosa [5], Petković *et al.* [11], Peng *et al.* [9] and Zhanlav and Otgondorj [22]. Such methods are especially useful when the derivative is very expensive and, of course, when the function is non-differentiable.

The methods are of order p using d function evaluations per step. Kung and Traub [8] conjectured that multipoint methods without memory using d evaluations could have order no larger than 2^{d-1}. The efficiency index I defined as $p^{1/d}$. Thus an optimal method of order 8 has an efficiency index of $I = 8^{1/4} = 1.6817$ and an optimal method of order 4 has an efficiency index $I = 4^{1/3} = 1.5874$ which is better than Newton's method for which $I = \sqrt{2} = 1.4142$. The efficiency index of optimal method cannot reach a value of 2. In fact realistically one uses methods of order at most 8. There are one step methods that converge super-linearly with $p = 1.618$ such as the secant method and $p = 1.839$. These one step methods require one function evaluation per step (except that they require additional starting value(s)). Therefore the efficiency index is 1.618 (secant) and 1.839. Here we will list the one-step methods along with their order of convergence. We will compare the basin of attraction of the methods and suggest a multistep method that is almost optimal but having higher efficiency index (because it uses one less function evaluation). We measure the behavior on 5 functions, 4 of which are polynomials. It was demonstrated by Petković and Herceg [10] that the conclusions are the same even if we take more examples.

The basin of attraction method was initially discussed by Stewart [18]. This is better than comparing methods on the basis of running several nonlinear functions using a certain initial value. In the last decade many papers appeared using the idea of basin of attraction to compare the efficiency of many methods. See, for example, Chun and Neta [2], [3] and references there.

The paper is organized in the following way: In the next section, we will describe several super linear and one quadratic derivative-free methods. In section 3, we compare the methods using the idea of basins of attraction. Section 4 details our new derivative-free method of order 6.219 and in section 5 we study its dynamics and compare it to a certain eighth-order derivative-free method. We close with conclusions.

2. SUPER-LINEAR DERIVATIVE-FREE SCHEMES

In this section, we describe the following derivative-free methods in the literature:

(1) Secant Method ($p = 1.618$)
(2) Three Traub's Methods ($p = 1.839$)
(3) Jarratt and Nudds' method ($p = 1.839$)
(4) Popovski's method ($p = 1.839$)
(5) Steffensen's method ($p = 2$)

Secant Method Given $f(x)$ and x_0, x_1 be 2 initial guesses for the root ξ, then

$$x_{n+1} = x_n - \frac{f_n}{\frac{f_n - f_{n-1}}{x_n - x_{n-1}}}, \qquad n = 1, 2, \ldots \tag{1}$$

where $f_k = f(x_k)$.

This algorithm is similar to Regula Falsi. The order of convergence is 1.618 and the method requires one function evaluation per iteration. thus the efficiency index $I = 1.618$. Geometrically, the method approximates the slope of $f(x_n)$ by using the secant line via the points (x_n, f_n) and (x_{n-1}, f_{n-1}).

Traub methods ([20])

(1) The first method denoted Traub1 (Method 7a on page 234) is:

$$x_{n+1} = x_n - \frac{f_n}{\frac{(f_{n-2}-f_{n-1})+(f_{n-1}-f_n)}{(x_{n-2}-x_{n-1})+(x_{n-1}-x_n)} - \frac{f_{n-2}-f_{n-1}}{x_{n-2}-x_{n-1}} + \frac{f_{n-1}-f_n}{x_{n-1}-x_n}} \tag{2}$$

(2) The second method denoted Traub2 (Method 5a on page 233) is:

$$x_{n+1} = x_n - \left[\frac{(x_{n-2}-x_{n-1})+(x_{n-1}-x_n)}{(f_{n-2}-f_{n-1})+(f_{n-1}-f_n)} - \frac{x_{n-2}-x_{n-1}}{f_{n-2}-f_{n-1}} + \frac{x_{n-1}-x_n}{f_{n-1}-f_n}\right] f_n \tag{3}$$

(3) The third method denoted Traub3 (Method 2a on page 233) is given by:

$$x_{n+1} = x_n + \left\{ \left[\frac{x_n - x_{n-1}}{f_n - f_{n-1}} - \frac{x_{n-1}-x_{n-2}}{f_{n-1}-f_{n-2}}\right] \frac{f_{n-1}}{f_n - f_{n-2}} - \frac{x_n - x_{n-1}}{f_n - f_{n-1}} \right\} f_n. \tag{4}$$

All three Traub's methods have order and efficiency index of 1.839 (see [20, p. 233]).

Jarratt and Nudds

Jarratt and Nudds [6] developed a method based on the rational interpolating function

$$y(x) = \frac{x - a}{bx + c}$$

using the last three points $(x_i, f_i), \quad i = n-2, n-1, n$ as data. Thus the method

$$x_{n+1} = x_n + \frac{[(x_n - x_{n-1}) + (x_{n-1} - x_{n-2})]\,(x_n - x_{n-1}) f_n (f_{n-1} - f_{n-2})}{f_{n-2}(x_{n-1} - x_{n-2})(f_n - f_{n-1}) - f_n (x_n - x_{n-1})(f_{n-1} - f_{n-2})}. \tag{5}$$

The rate of convergence of the method and the efficiency are 1.839.

Popovski's method

Popovski [13] used the same interpolating function as Jarratt and Nudds [6] to derive the following method of order 1.839. The method is referred to as the method of tangential hyperbolic approximation.

$$x_{n+1} = x_n - \frac{[(x_{n-2}-x_{n-1})+(x_{n-1}-x_n)](f_{n-2}-f_{n-1})(x_{n-1}-x_n)}{[(f_{n-2}-f_{n-1})+(f_{n-1}-f_n)](x_{n-2}-x_{n-1})(f_{n-1}-f_n)} f_n. \tag{6}$$

The efficiency index of the method is $I = 1.839$.

Steffensen's family of methods

Steffensen's method [17]

$$w_n = x_n + \gamma(f_n)$$
$$x_{n+1} = x_n - \frac{\gamma(f_n)^2}{f(w_n - f_n)} \tag{7}$$

The method converges quadratically and uses two function-evaluation. Therefore its efficiency index is 1.4142 as Newton's method. This is the reason that higher order derivative-free methods use the method as a first step.

3. DYNAMICS STUDY OF THE METHODS

In this section, we describe the experiments with each of the 7 methods detailed above. We chose 4 polynomials and one non-polynomial function all having roots within a 6 by 6 square centered at the origin. The square is divided horizontally and vertically by equally spaced lines. We took the intersection of all these lines as initial points in the complex plane for the iterative schemes. The code collected the number of iteration or function evaluation to converge within a tolerance of 10^{-7} and the root to which the sequence converged. If the sequence did not converge within 40 iteration, we denote it as a divergent point. Each point is colored by the color corresponding to the root. A divergent point is colored black. We also collected the CPU run time to execute the code on all initial points using Dell Optiplex 990 desktop computer.

We ran all methods on the following 5 examples, 4 of which are polynomials:

(1) $z^2 - 1$
(2) $z^3 - 1$
(3) $z^4 - 1$
(4) $z^5 - 1$
(5) $\left(e^{z+1} - 1\right)(z - 1)$

REMARK 1. The additional starting values are $x_{-1} = x_0 + 0.01$ and $x_{-2} = x_0 + 0.02$. □

EXAMPLE 2. The first example is

$$p_1(z) = z^2 - 1, \tag{8}$$

vanishing at $z = \pm 1$. The basins for all methods are given in Fig. 3.1. Clearly from the figure one can see that the best schemes are secant, Traub1 and Jarratt-Nudds methods.

From Table 1, the average number of function-evaluations per point is minimum (5.82) for Traub1 followed by Jarratt-Nudds (5.86). The highest number (11.75) was used by Traub2.

The CPU runtime in seconds, see Table 2, is the lowest for secant method (115.423) followed by Traub1 (122.283). The slowest is Traub2 with 227.337 seconds.

The secant method has no divergent point (see Table 3) and Popovski has only 1 point. Traub2 has the highest number of divergent points (52648), which is unacceptable. □

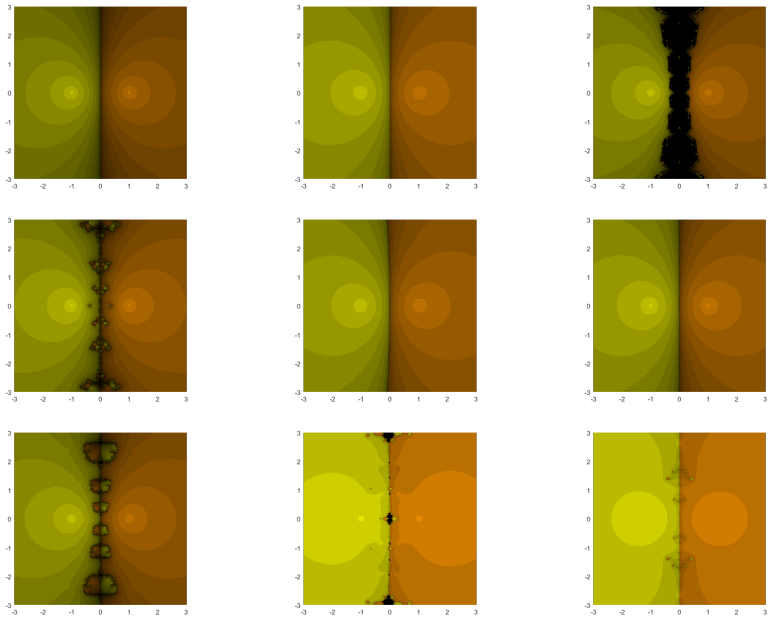

Fig. 3.1. The top row for Secant (left), Traub1 (center), Traub2 (right). Second row for Traub3 (left), Steffensen (center) and Jarratt-Nudds (right). Third row for Popovski (left), TZKO (center), Neta (right) for the roots of the polynomial $z^2 - 1$.

Method	Ex1	Ex2	Ex3	Ex4	Ex5	average
Secant	7.78	11.5	16.65	17.35	6.85	12.02
Traub1	5.82	6.97	8.88	9.62	6.52	7.56
Traub2	11.75	17.70	22.34	21.15	9.26	16.44
Traub3	7.10	10.32	15.64	16.35	6.64	11.21
Steffensen	5.78	7.93	10.84	17.47	5.28	9.46
Popovski	5.87	7.13	8.84	9.23	5.86	7.39
JN	7.32	9.79	13.24	13.98	7.28	10.32
TZKO	11.85	19.52	25.60	29.26	12.25	19.70
Neta	10.54	10.91	13.13	13.79	9.10	11.49

Table 1. Average number of function evaluations per point for each example (1–5) and each of the methods.

Example 3. The cubic polynomial

$$p_2(z) = z^3 - 1, \tag{9}$$

Method	Ex1	Ex2	Ex3	Ex4	Ex5	average
Secant	115.423	215.15	373.899	418.568	154.492	255.506
Traub1	122.283	183.568	252.397	328.295	195.38	216.385
Traub2	227.337	394.5	590.03	584.702	217.026	402.719
Traub3	170.923	287.784	508.552	557.584	245.495	354.068
Steffensen	111.546	224.375	306.197	545.721	174.007	272.370
Popovski	154.366	272.019	392.188	482.016	212.037	302.525
JN	156.286	221.251	304.169	425.87	196.356	260.786
TZKO	290.886	545.869	621.265	745.541	334.541	507.620
Neta	231.431	311.934	368.234	452.034	282.761	329.279

Table 2. CPU time (msec) for each example (1–5) and each of the methods.

Method	Ex1	Ex2	Ex3	Ex4	Ex5	average
Secant	0	14605	62225	64895	139	28375
Traub1	601	2	203	1054	809	534
Traub2	52648	110398	156658	133868	27955	96305
Traub3	61	11034	58766	68535	34	27686
Steffensen	2	6188	27514	95107	997	25962
Popovski	197	25	1771	3464	8	1093
JN	1	645	6305	11911	5180	4808
TZKO	2364	16674	27745	33419	2640	16568
Neta	2930	0	0	1	901	766

Table 3. Number of black points for each example (1–5) and each of the methods.

having the 3 roots of unity. Figure 3.2 shows that the best schemes are again Traub1 and Jarratt-Nudds methods.

The average number of function-evaluations per point (Table 1) is minimum (6.97) for Traub1 followed by Popovski (7.13). The highest number (17.70) was used by Traub2.

Based on the CPU time in seconds, we find that the fastest method is Traub1 (183.568 seconds) followed by secant method (215.15 seconds). The slowest is Traub2 with 394.5 seconds.

Traub1 (see Table 3) has the lowest number (2 divergent points), followed by Popovski (25 points). □

Example 4. The third example is a quartic polynomial

$$p_3(z) = z^4 - 1 \tag{10}$$

and the roots are just the four roots of unity. Figure 3.3 shows that the best schemes are again Jarratt-Nudds and Traub1 methods.

The minimum average number of function-evaluations per point (Table 1) is 8.84 for Popovski followed closely by Traub1 (8.88). The highest number (22.34) was for Traub2.

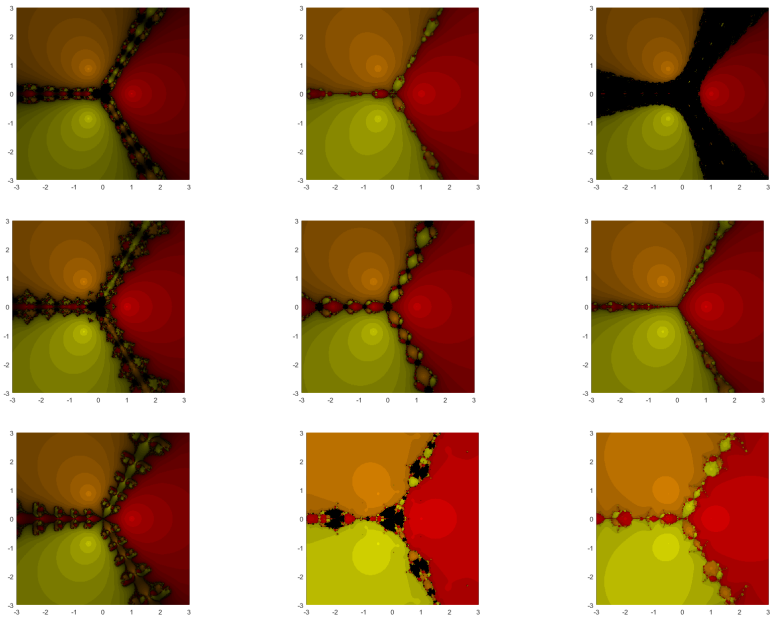

Fig. 3.2. The top row for Secant (left), Traub1 (center), Traub2 (right). Second row for Traub3 (left), Steffensen (center) and Jarratt-Nudds (right). Third row for Popovski (left), TZKO (center), Neta (right) for the roots of the polynomial $z^3 - 1$.

Based on the CPU execution time in seconds, we find again that the fastest method is Traub1 (252.397 seconds) and the slowest is again Traub2 with 590.03 seconds.

Traub1 has the lowest number of black points (see Table 3). Again Traub2 has the highest number of divergent points (156658). □

Example 5. The fourth example is a fifth degree polynomial

$$p_4(z) = z^5 - 1. \tag{11}$$

Figure 3.4 indicates that the best scheme is Jarratt-Nudds method.

Based on Table 1 the minimum average number of function-evaluations per point is 9.23 for Popovski followed by Traub1 with 9.62.

Based on the CPU time in seconds, we find again that the fastest method is Traub1 (328.295 seconds). The slowest is again Traub2.

In terms of the number of black points (see Table 3) we find again that Traub1 has the lowest number (1054 points) and the highest is Traub2 (133868 points). □

Example 6. The last example is a non-polynomial example.

$$F_1(z) = (e^{z+1} - 1)(z - 1). \tag{12}$$

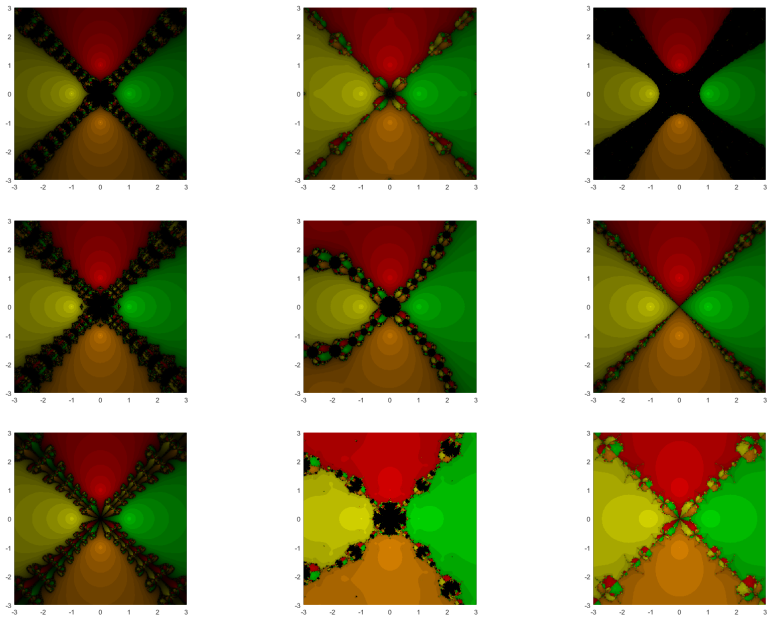

Fig. 3.3. The top row for Secant (left), Traub1 (center), Traub2 (right). Second row for Traub3 (left), Steffensen (center) and Jarratt-Nudds (right). Third row for Popovski (left), TZKO (center), Neta (right) for the roots of the polynomial $z^4 - 1$.

Figure 3.5 indicates an interesting phenomenon. The basin for $z = -1$ is much larger than the other. It seems that the best scheme is Jarratt-Nudds method, even though Traub1 give slightly more equitable division of the basins.

The minimum average number of function-evaluations per point (see Table 1) is 5.28 for Steffensen's method followed closely by Popovski with 5.86. This is the first time that Steffensen's method came at the top. The highest number (9.26) for Traub2.

The CPU runtime in seconds (Table 2) is the lowest for secant (154.492 seconds), followed by Steffensen's method with 174.007 seconds. The slowest is now Traub3.

Popovski's method has the lowest number (8) of divergent points, followed by Traub3 (34). The highest is Traub2 (27955 points). □

We now averaged the results across the five examples and find the top three methods in each category. Based on Table 1 the order is Popovski, Traub1 and Steffensen. In terms of CPU time: Traub1, Secant, Jarratt-Nudds method. The lowest number of divergent points is for Traub1 followed by Popovski and Jarratt-Nudds. Therefore we find the Traub1 is the only method in the top three in all three categories. As a results, we have decided to try and develop a method (of order 6.219) based on Traub1 as first step instead of

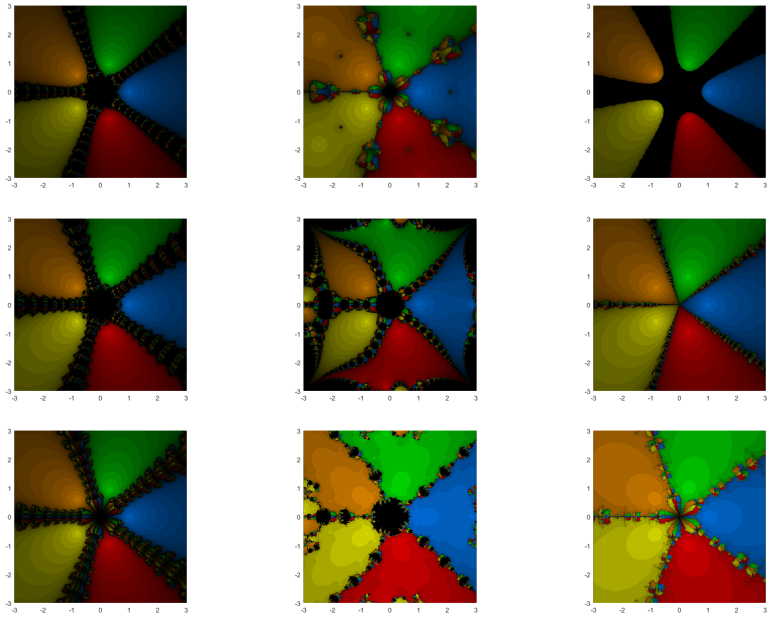

Fig. 3.4. The top row for Secant (left), Traub1 (center), Traub2 (right). Second row for Traub3 (left), Steffensen (center) and Jarratt-Nudds (right). Third row for Popovski (left), TZKO (center) for the roots of the polynomial $z^2 - 1$.

Steffensen's method. The new method is of order $p = 1.839^3 = 6.219$ and uses only 3 function evaluations per cycle. Therefore the efficiency index is $I = 6.219^{1/3} = 1.839$ which is higher than the optimal eighth order method. This is possible for methods with memory.

In the next section, we will develop the new scheme and discuss the order and efficiency index. The section after that will compare the basins for our new method to an optimal eighth order method due to Zhanlav and Otgondorj.

4. NEW METHOD

We now develop a 3-step method having Traub1 method as first step instead of Steffensen. The method is

$$
\begin{aligned}
y_n &= x_n - \frac{f_n}{\frac{(f_{n-2}-f_{n-1})+(f_{n-1}-f_n)}{(x_{n-2}-x_{n-1})+(x_{n-1}-x_n)} - \frac{f_{n-2}-f_{n-1}}{x_{n-2}-x_{n-1}} + \frac{f_{n-1}-f_n}{x_{n-1}-x_n}}, \\
z_n &= y_n - \frac{f(y_n)}{f'(y_n)}, \\
x_{n+1} &= z_n - \frac{f(z_n)}{f'(z_n)}.
\end{aligned}
\tag{13}
$$

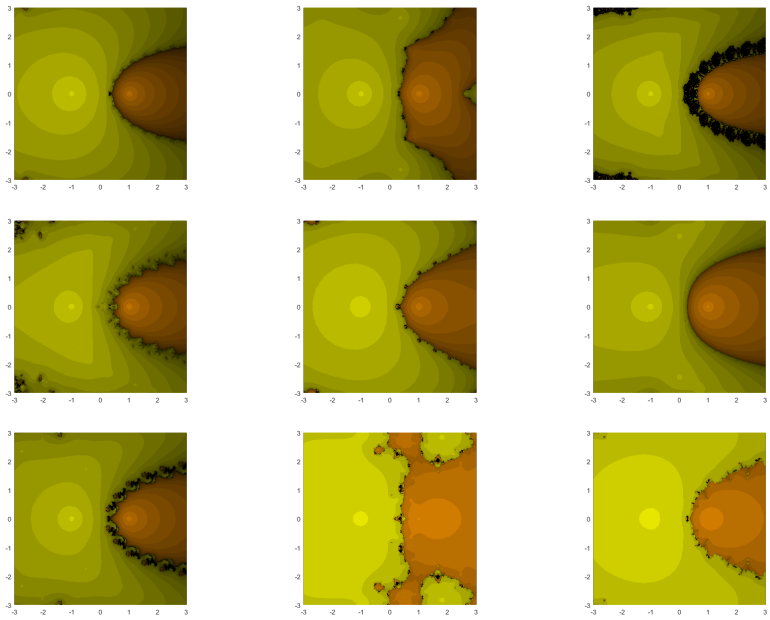

Fig. 3.5. The top row for Secant (left), Traub1 (center), Traub2 (right). Second row for Traub3 (left), Steffensen (center) and Jarratt-Nudds (right). Third row for Popovski (left), TZKO (center), Neta (right) for the roots of $(e^{z+1}-1)(z-1)$.

As usual, we approximate the derivatives in the last two steps by an interpolating polynomial, in this case, using the last 3 function values,

$$f'(y_n) = \alpha_1 f_n + \alpha_2 f_{n-1} + \alpha_3 f(y_n), \tag{14}$$

where

$$\alpha_1 = -\frac{x_{n-1}-y_n}{(x_n-y_n)(x_n-x_{n-1})},$$

$$\alpha_2 = \frac{x_n-y_n}{(x_n-x_{n-1})(x_{n-1}-y_n)}, \tag{15}$$

$$\alpha_3 = -\frac{x_n+x_{n-1}-2y_n}{(x_n-y_n)(x_{n-1}-y_n)},$$

and

$$f'(z_n) = \beta_1 f_n + \beta_2 f(y_n) + \beta_3 f(z_n), \tag{16}$$

where

$$\beta_1 = -\frac{y_n-z_n}{(x_n-z_n)(x_n-y_n)},$$

$$\beta_2 = \frac{x_n-z_n}{(x_n-y_n)(y_n-z_n)}, \tag{17}$$

$$\beta_3 = -\frac{x_n+y_n-2z_n}{(x_n-z_n)(y_n-z_n)}.$$

Index	$f(x)$	x_0	number of iterations	COC
1	$\left(e^{x+3}-1\right)(x-1)$	10.0	8	5.567
2	x^3+4x^2-10	-2.6	13	4.817
3	$(\sin x)^2-x^2+1$	2.0	4	5.162
4	$(x-1)^3-1$	3.5	5	4.826
5	x^3-10	4.0	4	5.179
6	$xe^{x^2}-(\sin x)^2+3\cos x+5$	-1.0	4	4.674
7	$e^{x^2+7x-30}-1$	4.0	9	4.814
8	$\sin x-x/2$	2.0	4	4.851
9	$x^5+x-10000$	4.0	5	4.996
10	$\sqrt{(}x)-1/x-3$	9.0	3	5.222
11	e^x+x-20	0.0	7	4.716
12	$\ln x+\sqrt{(}x)-5$	10.0	4	4.738
13	x^3-x^2-1	4.0	5	5.011
14	x^5-1	10.0	8	4.858
15	$\left(e^{x+1}-1\right)(x-1)$	5.0	6	5.242
16	$\left(e^{x+3}-1\right)\left(e^{x-1}-1\right)$	15.0	15	4.836

Table 4. Computational order of convergence for several functions using our new method.

This method uses memory, the point x_{n-1} in the second step. The order of the method is the product of the orders $p = 1.839^3 = 6.219$, see [20] Theorem 2.4. We arrived at this by using the fact that each step uses the values of f at the current and two previously computed values, and it is known that in that case the order is 1.839. Clearly this is not optimal. The efficiency index is $I = 6.219^{1/3} = 1.839$. This is better than the optimal eighth-order method (denoted TZKO) without memory of [22].

We now refer to Table 4 showing the computational order of convergence as defined by

$$coc = \frac{\ln\left|\frac{x_i - nexx}{x_{i-1} - nexx}\right|}{\ln\left|\frac{x_{i-1} - nexx}{x_{i-2} - nexx}\right|} \tag{18}$$

where $nexx$ is the final approximation for the zero.

5. DYNAMICS OF OUR NEW METHOD

We ran the same five examples from the numerical experiments section above on our new method (Neta) and compared it to TZKO [22]. The results are given in the previous figures as the last two sub figures and in Tables 1 to 3. The average over all 5 examples of the average number of function evaluation is 12.132 *versus* 18.382 for TZKO. Our new method is much faster 329.279 seconds *versus* 507.620 second for TZKO. The number of divergent point on average over all examples is 766.4 for our method *versus* 16568.4 for TZKO. We can conclude that our method is better than the optimal method of Zhanlav and Otgondorj.

CONCLUSIONS

We have considered several derivative free super linear methods and compared them to the quadratic method due to Steffensen. We found that one of Traub's method, namely Traub1, performed best on the five examples. We then developed a new method with memory based on Traub1 method as first step and we showed that our method is faster, have much less divergent points and require less iteration per initial point.

Acknowledgements. The author thank Professor M.S. Petković for his very helpful comments on the previous version of the manuscript.

REFERENCES

[1] F.I. Chicharro, A. Cordero, J.R. Torregrosa, M.P. Vassileva, *King-type derivative-free iterative families: real and memory dynamics*, Complexity, 2017, Article ID 2713145, 15 pages.

[2] C. Chun, B. Neta, *Comparative study of methods of various orders for finding simple roots of nonlinear equations*, J. Appl. Anal. Comput., **9** (2019), pp. 400–427.

[3] C. Chun, B. Neta, *Comparative study of methods of various orders for finding repeated roots of nonlinear equations*, J. Comput. Appl. Math., **340** (2018), pp. 11–42.

[4] C. Chun, B. Neta, *Comparative study of eighth-order methods for finding simple roots of nonlinear equations*, Numer. Algor., **74** (2017), pp. 1169–1201.

[5] A. Cordero, J.L. Hueso, E. Martinez, J.R. Torregrosa, *Generating optimal derivative free iteration methods for nonlinear equations by using polynomial interpolation*, Math. Comput. Model., **57** (2013) (7-8), pp. 1950–1956.

[6] P. Jarratt, D. Nudds, *The use of rational functions in the iterative solution of equations on a digital computer*, Computer J., **8** (1965), pp. 62–65.

[7] S.K. Khattri, T. Steihaug, *Algorithm for forming derivative-free optimal methods*, Numer. Algor., **65** (2014), pp. 809–824.

[8] H.T. Kung, J.F. Traub, *Optimal order of one-point and multipoint iteration*, J. Assoc. Comput. Math., **21** (1974), pp. 634–651.

[9] Y. Peng, H. Feng, Q. Li, X. Zhang, *A fourth-order derivative-free algorithm for nonlinear equations*, J. Comput. Appl. Math., **235** (2011), pp. 2551–2559.

[10] I. Petković, D. Herceg, *Computers in mathematical research: the study of three-point root-finding methods*, Numer. Algor., **84** (2020), pp. 1179–1198.

[11] M.S. Petković, S. Illic, J. Džunić, *Derivative-free two-point methods with and without memory for solving nonlinear equations*, Appl. Math. Comput., **217** (2010), pp. 1887–1895.

[12] M.S. Petković, B. Neta, L.D. Petković, J. Džunić, *Multipoint Methods for the Solution of Nonlinear Equations*, Elsevier, 2012.

[13] D.B. Popovski, *Method of tangential hyperbolic approximation for solving equations*, Proc. 3rd Int. Symp. Computers at the University, Cavtat, May 25-28, 1981, 311.1–311.6.

[14] J.R. Sharma, R.K. Goyal, *Fourth-order derivative-free methods for solving nonlinear equations*, Inter. J. Computer Math., **83** (2006), pp. 101–106.

[15] F. Soleymani, S. Shateyi, *Two optimal eighth-order derivative-free classes of iterative methods*, Abstract. Applied Anal., 2012, ID 318165, pp. 1–14.

[16] F. Soleymani, S.K. Vanani, *Optimal Steffensen-type methods with eighth order of convergence*, Comp. Math. Appl., **62** (2011), pp. 4619–4626.

[17] J.F. Steffensen, *Remarks on iteration*, Scand. Actuar. J., **1** (1933), pp. 64–72.

[18] B.D. Stewart, *Attractor Basins of Various Root-Finding Methods*, M.S. thesis, Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA, June 2001.
[19] R. Thukral, *Eighth-order iterative methods without derivatives for solving nonlinear equations*, International Scholarly Research Notices, 2011, ID 693787, pp. 1–12.
[20] J.F. Traub, *Iterative Methods for the Solution of Equations*, Prentice Hall, New York, 1964.
[21] H. Veiseh, T. Lotfi, T. Allahviranloo, *A study on the local convergence and dynamics of the two-step and derivative-free Kung-Traub's method*, Comp. Applied. Math., **37** (2018), pp. 2428–2444.
[22] T. Zhanlav, K. Otgondorj, *Comparison of some optimal derivative-free three-point iterations*, J. Numer. Anal. Approx. Theory, **49** (2020), pp. 76–90.
[23] Q. Zheng, J. Li, F. Huang, *An optimal Steffensen type family for solving nonlinear equations*, Appl. Math. Comput., **217** (2011), pp. 9592–9597.

Received by the editors: December 4, 2020; accepted: December 21, 2020; published online: February 20, 2021.