Applied Mathematics and Computation 268 (2015) 129–137

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Basins of attraction for several third order methods to find multiple roots of nonlinear equations

Changbum Chun[a], Beny Neta[b,*]

[a] Department of Mathematics, Sungkyunkwan University, Suwon 440-746, Republic of Korea
[b] Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA 93943, United States

ARTICLE INFO

MSC:
65H05
65B99

Keywords:
Iterative methods
Order of convergence
Basin of attraction
Multiple roots

ABSTRACT

There are several third order methods for solving a nonlinear algebraic equation having roots of a given multiplicity m. Here we compare a recent family of methods of order three to Euler–Cauchy's method which is found to be the best in the previous work. There are fewer fourth order methods for multiple roots but we will not include them here.

Published by Elsevier Inc.

1. Introduction

There is a vast literature on the solution of nonlinear equations, see for example Ostrowski [1], Traub [2], Neta [3] and Petković et al. [4]. Here we are interested in algorithms for finding a multiple root of a nonlinear equation $f(x) = 0$. A root α of $f(x)$ is of multiplicity $m > 1$ if $f(\alpha) = 0,\ f^{(i)}(\alpha) = 0$ for $i = 1, 2, \dots, m-1$ and $f^{(m)}(\alpha) \neq 0$. The first method is due to Schröder [5] and it is also referred to as modified Newton,

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

The method is based on Newton's method for the function $G(x) = \sqrt[m]{f(x)}$ which obviously has a simple root at α, the multiple root with multiplicity m of $f(x)$.

Another method based on the same G is Laguerre's method

$$x_{n+1} = x_n - \frac{\lambda\frac{f(x_n)}{f'(x_n)}}{1 + sgn(\lambda - m)\sqrt{\left(\frac{\lambda-m}{m}\right)\left[(\lambda-1) - \lambda\frac{f(x_n)f''(x_n)}{f'(x_n)^2}\right]}} \tag{2}$$

where $\lambda\ (\neq 0, m)$ is a real parameter. When $f(x)$ is a polynomial of degree n, this method with $\lambda = n$ is the ordinary Laguerre method for multiple roots, see Bodewig [6]. This method converges cubically. One special case is Euler–Cauchy for $\lambda = 2m$

$$x_{n+1} = x_n - \frac{2m\frac{f(x_n)}{f'(x_n)}}{1 + \sqrt{(2m-1) - 2m\frac{f(x_n)f''(x_n)}{f'(x_n)^2}}}. \tag{3}$$

* Corresponding author. Tel.: +1 831 656 2235; fax: +1 831 656 2355.
E-mail addresses: cbchun@skku.edu (C. Chun), bneta@nps.edu, byneta@gmail.com (B. Neta).

http://dx.doi.org/10.1016/j.amc.2015.06.068
0096-3003/Published by Elsevier Inc.

Other special cases include Halley's method [7], Ostrowski's method, and Hansen–Patrick's family [8]. Two other cubically convergent methods are: Euler–Chebyshev [2] and Osada's method [9]. Another variation on Chebyshev's method is given by Neta [10]. Sbibih et al. [11] has recently developed a new family of third order methods for multiple roots. The family depends on a weight function given by

$$\begin{aligned} y_n &= x_n - \mu\frac{f(x_n)}{f'(x_n)}, \\ w_n &= \frac{f(y_n)}{f(x_n)} \\ x_{n+1} &= x_n - \phi(w_n)\frac{f(x_n)}{f'(x_n)}, \end{aligned} \tag{4}$$

where the weight function ϕ is a complex function, and μ is a non-zero real or complex number. They have shown that the family is of order three, for $m \geq 2$, and of order four for simple roots, if the function ϕ satisfies the following conditions:

$$\begin{aligned} &\phi(t^m) = m \\ &\phi'(t^m) = \tfrac{1}{t^{m-1}(1-t)^2} \\ &\left|\left(\tfrac{1}{\phi'}\right)'(t^m)\right| < \infty \end{aligned} \tag{5}$$

where $t = 1 - \frac{\mu}{m}$.

They have also demonstrated that the following methods are special cases:

- Dong (two methods) [12]
- Victory and Neta [13]
- Neta [10]
- Chun and Neta [14]
- Homeier [15]
- Geum and Kim [16]
- Kim and Geum [17]

The authors picked four different weight functions and compared these four methods to existing ones by solving four nonlinear equations each having a root with a different multiplicity. The members are:

- SSTZ1

$$\begin{aligned} &\phi(x) = ax + b \\ &a = \tfrac{1}{t^{m-1}(1-t)^2} \\ &b = m - \tfrac{t}{(1-t)^2} \end{aligned} \tag{6}$$

- SSTZ2

$$\begin{aligned} &\phi(x) = \tfrac{a}{b-x} \\ &a = m^2 t^{m-1}(1-t)^2 \\ &b = mt^{m-1}(1-t)^2 + t^m \end{aligned} \tag{7}$$

- SSTZ3

$$\begin{aligned} &\phi(x) = x^2 + ax + b \\ &a = \tfrac{1}{t^{m-1}(1-t)^2} - 2t^m \\ &b = m + t^{2m} - \tfrac{t}{(1-t)^2} \end{aligned} \tag{8}$$

- SSTZ4

$$\begin{aligned} &\phi(x) = \tfrac{x^2+ax+b}{(1-x)^2} \\ &a = -2t^m - 2m(1-t^m) + \tfrac{(1-t^m)^2}{t^{m-1}(1-t)^2} \\ &b = t^{2m} + m(1-t^{2m}) - \tfrac{t(1-t^m)^2}{(1-t)^2} \end{aligned} \tag{9}$$

In the next section we will discuss basins of attraction for these four methods and compare the basins with the best known third order method for multiple roots, namely Euler–Cauchy's method (see [18]).

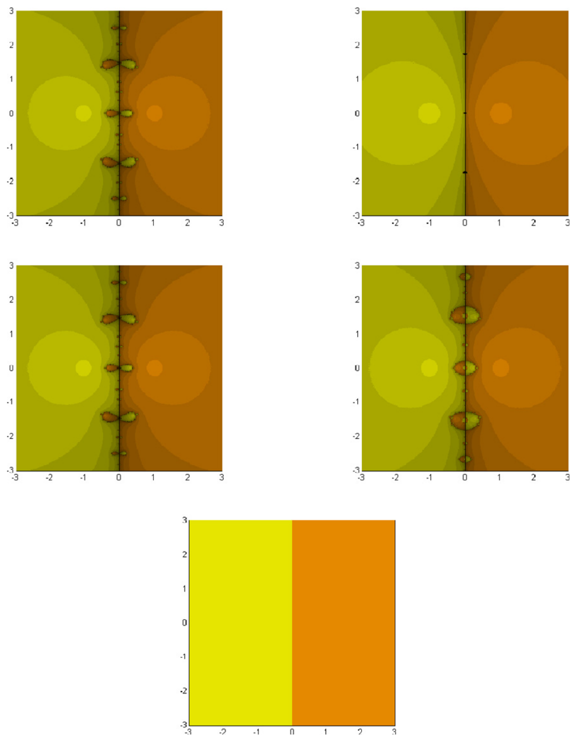

Fig. 1. The top left for SSTZ1, top right for SSTZ2, center left for SSTZ3, center right for SSTZ4 and the bottom is for Euler–Cauchy method for the roots of the polynomial $(z^2-1)^2$.

2. Numerical experiments

The Basin of Attraction is a method to visually understand how a method behaves as a function of the various starting points. This idea was started by Stewart [19] and continued in the work of Amat et al. [20], [21], [22], [23], Scott et al. [24], Chun et al. [25], Chicharro et al. [26], Cordero et al. [27], Neta et al. [28], Argyros and Magreñan [29], Magreñan [30], and Chun et al. [31]. The only papers comparing basins of attraction for methods to obtain multiple roots are due to Neta et al. [32], Neta and Chun [33], [34], and Chun and Neta [35].

We have used the above four methods with $\mu = 1/3$ and Euler–Cauchy's method for seven different polynomials having multiple roots with multiplicity $m = 2, 3, 4, 5$. The choice of the parameter μ is based on the numerical experiments reported by Sbibih et al. [11]. All the examples have roots within a square of [−3,3] by [−3,3]. We have taken 360,000 equally spaced points in the square as initial points for the methods and we have registered the total number of iterations required to converge to a root and also to which root it converged. We have also collected the CPU time (in seconds) required to run each method on all the points using Dell Multiplex 990 desktop computer. We then computed the average number of iterations required per point and the standard deviation.

Example 1. In our first example, we have taken the polynomial

$$p_1(z) = (z^2-1)^2 \tag{10}$$

whose roots $z = \pm 1$ are both real and of multiplicity $m = 2$. In Fig. 1 we have presented the basins for the five methods. The top left plot is for method SSTZ1, the top right for SSTZ2, the center left for SSTZ3 and the center right for SSTZ4. The bottom plot is for Euler–Cauchy's method. It is clear that the Euler–Cauchy's method outperformed the others. The basins are separated by a straight line in the middle for Euler–Cauchy's method without any black points (where the method used the maximum number of iterations). SSTZ2 is the second best. In Table 1 we can see that the average number of iterations per point is very close to unity for the best method and slightly below 4 for the second best. In Table 2 we have computed the standard deviation (σ), so we can see how spread are the number of iterations per point. The smaller the value of σ the closer the number of iterations per point to the average. Again, Euler–Cauchy's method is best followed by SSTZ2. In Table 3 we find the CPU time required to run each method on all the 360,000 points. The same conclusion is reached, namely Euler–Cauchy's method is best followed by SSTZ2. The worst methods are SSTZ1 and SSTZ3 which have a polynomial weight function. This is not surprising, since we have the same conclusion in [35].

Table 1
Average number of iterations per point for each example (1–7) and each of the five methods.

Example	SSTZ1	SSTZ2	SSTZ3	SSTZ4	Euler–Cauchy
1	4.4531	3.8778	4.4535	4.2884	1.0001
2	6.3350	5.0882	6.3364	6.2953	3.4661
3	6.3097	4.8149	6.3185	5.8236	3.8117
4	9.4700	6.1552	9.4724	8.0588	4.7388
5	6.3611	5.0747	6.4000	5.6255	3.8117
6	10.0776	6.6465	10.0779	8.3533	5.6106
7	6.0451	4.9196	6.0460	5.8522	4.1451
Average	7.0074	5.2253	7.0149	6.3282	3.7977

Table 2
Standard deviation for each example (1–7) and each of the five methods.

Example	SSTZ1	SSTZ2	SSTZ3	SSTZ4	Euler–Cauchy
1	2.0688	1.8453	2.0770	1.9249	0.0649
2	3.0556	2.2588	3.0693	3.0308	0.5420
3	4.5263	3.8211	4.5845	3.0010	0.6514
4	8.4660	5.8946	8.4598	5.4534	1.0020
5	4.6101	4.8056	4.8322	2.7306	0.6514
6	9.1531	6.7343	9.1552	5.8117	1.4012
7	2.6639	1.3797	2.6705	2.2443	0.6887

Table 3
CPU time (in seconds) required for each example (1–7) and each of the five methods.

Example	SSTZ1	SSTZ2	SSTZ3	SSTZ4	Euler–Cauchy
1	197.70	181.27	212.14	229.77	104.00
2	463.07	389.28	464.00	491.40	479.84
3	405.27	333.42	420.17	434.69	482.65
4	657.32	466.61	694.20	642.75	641.05
5	339.42	292.96	371.57	365.90	399.11
6	743.56	522.83	776.07	694.56	776.99
7	373.65	325.88	411.54	415.24	649.07
Average	454.28	358.89	478.53	467.76	504.67

Example 2. The second example is a polynomial whose roots are all of multiplicity three. The roots are $-2.68261500670705 \pm .358259359924043i$, 1.36523001341410, i.e.

$$p_2(z) = (z^3 + 4z^2 - 10)^3. \tag{11}$$

The basins are plotted in Fig. 2. Again the best method is Euler–Cauchy's as can be seen in the bottom sub-plot. This conclusion can be reached more quantitatively by comparing the average number of iterations per point in Table 1 and the standard deviation in Table 2. As for the CPU time in Table 3 we notice that Euler–Cauchy's method is slower. So for this example SSTZ2 could compete with Euler–Cauchy's method, at least in terms of CPU time.

Example 3. Our next example is a polynomial with roots of multiplicity four. The polynomial has the three roots of unity,

$$p_3(z) = (z^3 - 1)^4. \tag{12}$$

The basins are plotted in Fig. 3. The same conclusion can be reached here. Notice that the CPU time required for SSTZ4 is slightly higher than that of SSTZ3.

Example 4. The fourth example is a polynomial with roots of multiplicity five

$$p_4(z) = (z^4 - 1)^5 \tag{13}$$

where the roots are symmetrically located on the axis. In some sense this is similar to the first example, since in both the cases we have an even number of roots. The plots of the basins are given in Fig. 4. The best method is the same as before based on the plots in Table 1 and 2. The CPU time for SSTZ2 is lower than that for Euler–Cauchy's method (see Table 3).

Example 5. Our next example is also having the three roots of unity with double multiplicity

$$p_5(z) = (z^3 - 1)^2. \tag{14}$$

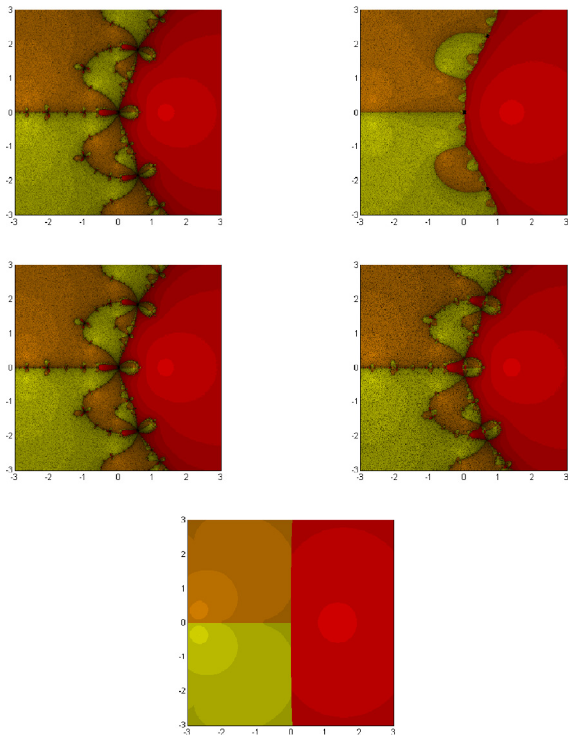

Fig. 2. The top left for SSTZ1, top right for SSTZ2, center left for SSTZ3, center right for SSTZ4 and the bottom is for Euler–Cauchy method for the roots of the polynomial $(z^3 + 4z^2 - 10)^3$.

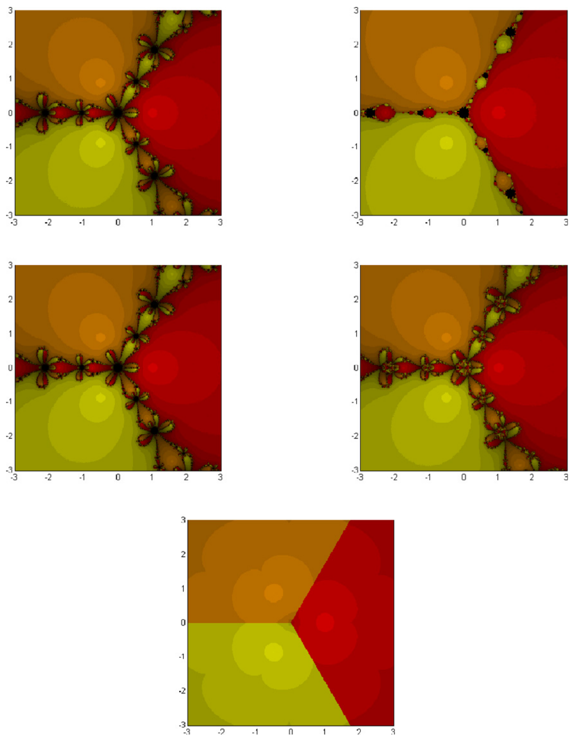

Fig. 3. The top left for SSTZ1, top right for SSTZ2, center left for SSTZ3, center right for SSTZ4 and the bottom is for Euler–Cauchy method for the roots of the polynomial $(z^3 - 1)^4$.

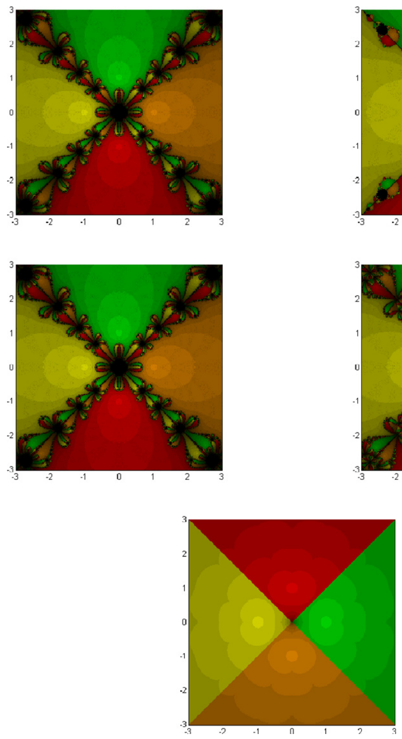

Fig. 4. The top left for SSTZ1, top right for SSTZ2, center left for SSTZ3, center right for SSTZ4 and the bottom is for Euler–Cauchy method for the roots of the polynomial $(z^4 - 1)^5$.

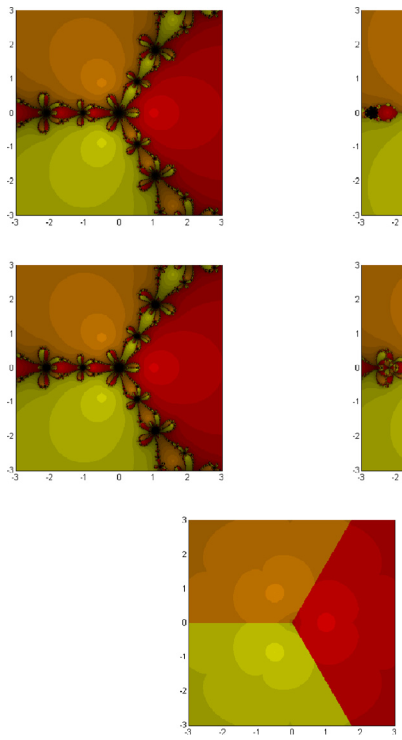

Fig. 5. The top left for SSTZ1, top right for SSTZ2, center left for SSTZ3, center right for SSTZ4 and the bottom is for Euler–Cauchy method for the roots of the polynomial $(z^3 - 1)^2$.

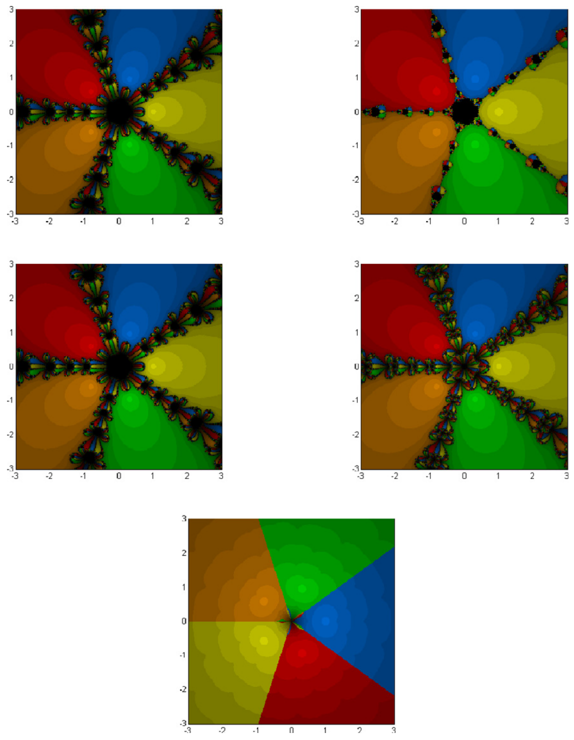

Fig. 6. The top left for SSTZ1, top right for SSTZ2, center left for SSTZ3, center right for SSTZ4 and the bottom is for Euler–Cauchy method for the roots of the polynomial $(z^5-1)^3$.

The basins are plotted in Fig. 5. The best method is again Euler–Cauchy's followed by SSTZ2. Looking at Table 1 we get the same conclusion. Based on the CPU time, we find that SSTZ2 is faster than Euler–Cauchy's method. It is interesting to note that even though the average number of iteration per point for SSTZ4 is slightly higher than that for SSTZ2, the standard deviation is smaller.

Example 6. In our sixth example we have the five roots of unity all with multiplicity three

$$p_6(z) = (z^5 - 1)^3. \tag{15}$$

The basins are plotted in Fig. 6. Again Euler–Cauchy's method is the best, but now we can see in the bottom sub-plot of Fig. 6 that there are some lobes around the origin. The CPU time for SSTZ2 is lowest even though the basins for SSTZ2 have more lobes along the boundaries. The average number of iterations per point (Table 1) and the standard deviation (Table 2) confirm the qualitative results in Fig. 6.

Example 7. In the last example we have the roots at $z = 0, \pm 1$ all with multiplicity four

$$p_7(z) = z^4(z^2 - 1)^4. \tag{16}$$

The basins are plotted in Fig. 7. The basin for $z = 0$ for Euler–Cauchy's method is the smallest. On the other hand, there are no lobes on the boundaries. It is hard to say which of the two methods (Euler–Cauchy or SSTZ2) is the best. The average number of iterations per point and the standard deviation are smallest for Euler–Cauchy's method, followed by SSTZ2.

3. Conclusions

In all seven examples, we find that the best is either Euler–Cauchy's method or SSTZ2 and the worst are those with polynomial weight functions, namely SSTZ1 and SSTZ3. In order to choose an overall best performer, we have computed the average of the values in Table 1 and 3. Based on the average CPU time for all examples, SSTZ2 is better than Euler–Cauchy's method. But based on the average number of points, the order is reversed. We must conclude that an iteration step for SSTZ2 is cheaper.

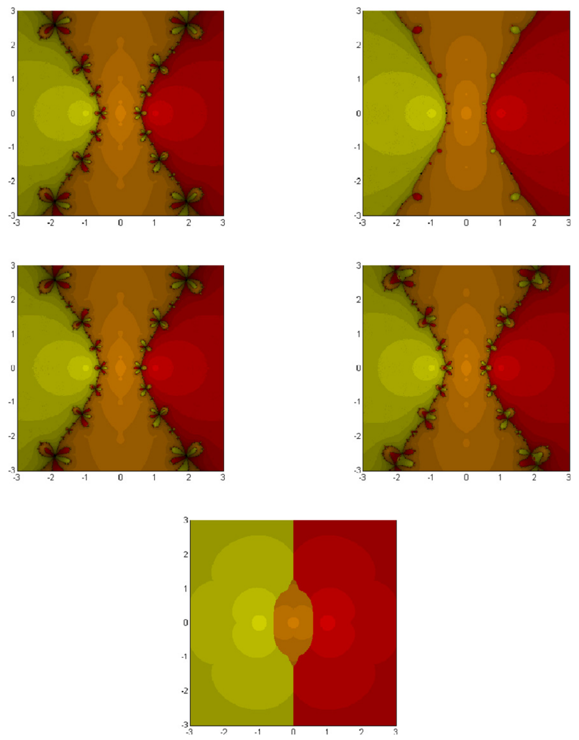

Fig. 7. The top right for SSTZ1, top left for SSTZ2, center right for SSTZ3, center left for SSTZ4 and the bottom is for Euler–Cauchy method for the roots of the polynomial $(z^3 - z)^4$.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2005012). The first author thanks the Applied Mathematics Department at the Naval Postgraduate School for hosting him during the years.

References

[1] A.M. Ostrowski, Solution of Equations in Euclidean and Banach Space, Academic Press, New York, 1973.
[2] J.F. Traub, Iterative Methods for the Solution of Equations, Chelsea publishing company, New York, 1977.
[3] B. Neta, Numerical Methods for the Solution of Equations, Net-A-Sof, California, 1983.
[4] M.S. Petković, B. Neta, L.D. Petković, J. Džunić, Multipoint Methods for Solving Nonlinear Equations, Elsevier, Waltham, MA, 2013.
[5] E. Schröder, über unendlich viele algorithmen zur auflösung der gleichungen, Math. Annal. 2 (1870) 317–365.
[6] E. Bodewig, Sur la méthode laguerre pour l'approximation des racines de certaines équations algébriques et sur la critique d'hermite, Indag. Math. 8 (1946) 570–580.
[7] E. Halley, A new, exact and easy method of finding the roots of equations generally and that without any previous reduction, Phil. Trans. Roy. Soc. London 18 (1694) 136–148.
[8] E. Hansen, M. Patrick, A family of root finding methods, Numer. Math. 27 (1977) 257–269.
[9] N. Osada, An optimal multiple root-finding method of order three, J. Comput. Appl. Math. 51 (1994) 131–133.
[10] B. Neta, New third order nonlinear solvers for multiple roots, Appl. Math. Comput. 202 (2008) 162–170.
[11] D. Sbibih, A. Serghini, A. Tijini, A. Zidna, A general family of third order method for finding multiple roots, Appl. Math. Comput. 233 (2014) 338–350.
[12] C. Dong, A basic theorem of constructing an iterative formula of the higher order for computing multiple roots of an equation, Math. Numer. Sinica 11 (1982) 445–450.
[13] H.D. Victory, B. Neta, A higher order method for multiple zeros of nonlinear functions, Int. J. Comput. Math. 12 (1983) 329–335.
[14] C. Chun, B. Neta, A third-order modification of Newton's method for multiple roots, Appl. Math. Comput. 211 (2009) 474–479.
[15] H.H.H. Homeier, On Newton-type methods for multiple roots with cubic convergence, J. Comput. Appl. Math. 231 (2009) 249–254.
[16] Y.H. Geum, Y.I. Kim, Cubic convergence of parameter-controlled newton-secant method for multiple zeros, J. Comput. Appl. Math. 233 (2009) 931–937.
[17] Y.I. Kim, Y.H. Geum, A cubic-order variant of Newton's method for finding multiple roots of nonlinear equations, Comput. Math. Appl. 62 (2011) 249–254.
[18] B. Neta, C. Chun, On a family of Laguerre methods to find multiple roots of nonlinear equations, Appl. Math. Comput. 219 (2013) 10987–11004.
[19] B.D. Stewart, Attractor Basins of Various Root-finding Methods, M.S. thesis, Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA, June 2001.
[20] S. Amat, S. Busquier, S. Plaza. Iterative Root-finding Methods, unpublished report, 2004.
[21] S. Amat, S. Busquier, S. Plaza, Review of some iterative root-finding methods from a dynamical point of view, Scientia 10 (2004) 3–35.
[22] S. Amat, S. Busquier, S. Plaza, Dynamics of a family of third-order iterative methods that do not require using second derivatives, Appl. Math. Comput. 154 (2004) 735–746.
[23] S. Amat, S. Busquier, S. Plaza, Dynamics of the king and Jarratt iterations, Aequ. Math. 69 (2005) 212–2236.
[24] M. Scott, B. Neta, C. Chun, Basin attractors for various methods, Appl. Math. Comput. 218 (2011) 2584–2599.

[25] C. Chun, M.Y. Lee, B. Neta, J. Džunić, On optimal fourth-order iterative methods free from second derivative and their dynamics, Appl. Math. Comput. 218 (2012) 6427–6438.
[26] F. Chicharro, A. Cordero, J.M. Gutiérrez, J.R. Torregrosa, Complex dynamics of derivative-free methods for nonlinear equations, Appl. Math. Comput. 219 (2013) 7023–7035.
[27] A. Cordero, J. García-Maimó, J.R. Torregrosa, M.P. Vassileva, P. Vindel, Chaos in king's iterative family, Appl. Math. Lett. 26 (2013) 842–848.
[28] B. Neta, M. Scott, C. Chun, Basin of attractions for several methods to find simple roots of nonlinear equations, Appl. Math. Comput. 218 (2012) 10548–10556.
[29] I.K. Argyros, A.A. Magreñan, On the convergence of an optimal fourth-order family of methods and its dynamics, Appl. Math. Comput. 252 (2015) 336–346.
[30] A.A. Magreñan, Different anomalies in a Jarratt family of iterative root-finding methods, Appl. Math. Comput. 233 (2014) 29–38.
[31] C. Chun, B. Neta, S. Kim, On Jarratt's family of optimal fourth-order iterative methods and their dynamics, Fractals 22 (2014) 1450013, doi:10.1142/S0218348X14500133.
[32] B. Neta, M. Scott, C. Chun, Basin attractors for various methods for multiple roots, Appl. Math. Comput. 218 (2012) 5043–5066.
[33] B. Neta, C. Chun, On a family of Laguerre methods to find multiple roots of nonlinear equations, Appl. Math. Comput. 219 (2013) 10987–11004.
[34] B. Neta, C. Chun, Basins of attraction for several optimal fourth order methods for multiple roots, Math. Comput. Simul. 103 (2014) 39–59.
[35] C. Chun, B. Neta, Basins of attraction for Zhou-Chen-Song fourth order family of methods for multiple roots, Math. Comput. Simul. 109 (2015) 74–91.