Numer Algor (2016) 72:363–376
DOI 10.1007/s11075-015-0048-9

ORIGINAL PAPER

On the new family of optimal eighth order methods developed by Lotfi et al.

Changbum Chun[1] · **Beny Neta**[2]

Received: 30 June 2015 / Accepted: 26 August 2015 / Published online: 10 September 2015

Abstract Recently Lotfi et al. (Numer. Algor. **68**, 261–288, 2015) have developed a new family of optimal order eight for the solution of nonlinear equations. They have experimented with 3 members of the family and compared them to other eighth order methods. One of the best known eight order method was not included. They also did not mention the best choice of parameters in the methods used and why. The basins of attraction were given for several examples without a quantitative comparison. It will be shown how to choose the best parameters in all these methods, and to quantitatively compare the methods.

Keywords Iterative methods · Order of convergence · Basin of attraction · Extraneous fixed points

Mathematics Subject Classification (2010) 65H05 · 65B99

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].

✉ Beny Neta
bneta@nps.edu

Changbum Chun
cbchun@skku.edu

[1] Department of Mathematics, Sungkyunkwan University, Suwon 440-746, Republic of Korea

[2] Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA 93943, USA

Lotfi et al. [5] have developed the following eighth order family of optimal methods (denoted LSSS)

$$\begin{aligned} y_n &= x_n - v_n, \\ z_n &= y_n - v_n \frac{t_n}{1-2t_n}, \\ x_{n+1} &= z_n - \frac{f(z_n)}{f'(x_n)} \frac{H(t_n)+K(s_n)}{G(u_n)}, \end{aligned} \tag{1}$$

where from here on we use the following:

$$v_n = \frac{f(x_n)}{f'(x_n)}, \tag{2}$$

and

$$t_n = \frac{f(y_n)}{f(x_n)}, \tag{3}$$

$$s_n = \frac{f(z_n)}{f(x_n)}, \tag{4}$$

$$u_n = \frac{f(z_n)}{f(y_n)}. \tag{5}$$

The weight functions $H,\ K,\ G$ satisfy

$$G(0) = 1,\ G'(0) = -1, \tag{6}$$

$$K(0) = 0,\ K'(0) = 2, \tag{7}$$

$$H(0) = 1,\ H'(0) = 2,\ H''(0) = 10,\ H'''(0) = 72. \tag{8}$$

They used in their comparison 3 members of their family (1) which we denote LSSS1, LSSS2 and LSSS3:

- LSSS1

$$\begin{aligned} G(u_n) &= 1 - u_n, \\ K(s_n) &= 2s_n, \\ H(t_n) &= 1 + 2t_n + 5t_n^2 + 12t_n^3. \end{aligned} \tag{9}$$

- LSSS2

$$\begin{aligned} G(u_n) &= 1 - \frac{u_n}{1+u_n}, \\ K(s_n) &= \frac{2s_n}{1+s_n}, \\ H(t_n) &= \frac{1+3t_n+7t_n^2+17t_n^3}{1+t_n}. \end{aligned} \tag{10}$$

- LSSS3

$$\begin{aligned} G(u_n) &= \frac{1-2u_n}{1-u_n}, \\ K(s_n) &= \frac{1+3s_n}{1+s_n} - 1, \\ H(t_n) &= 1 + t_n + 3t_n^2 + \frac{t_n + 3t_n^2 + 14t_n^3}{1+t_n}. \end{aligned} \tag{11}$$

They also include the following methods in their comparative study [5]:

- BCST, a method by Babajee et al. [6]

$$\begin{aligned} y_n &= x_n - v_n(1+v_n^5), \\ z_n &= y_n - \frac{f(y_n)}{f'(x_n)}(1-t_n)^{-2}, \\ x_{n+1} &= z_n - \frac{f(z_n)}{f'(x_n)} \frac{1+t_n^2+5t_n^4+u_n}{(1-t_n-s_n)^2}. \end{aligned} \tag{12}$$

- CFGT, a method by Cordero et al. [7]

$$\begin{aligned} y_n &= x_n - v_n, \\ z_n &= y_n - \frac{f(y_n)}{f'(x_n)} \frac{1}{1-2t_n-t_n^2-t_n^3/2}, \\ x_{n+1} &= z_n - \frac{1+3s_n}{1+s_n} \frac{f(z_n)}{f[z_n,y_n]+f[z_n,x_n,x_n](z_n-y_n)}, \end{aligned} \tag{13}$$

where the divided differences

$$f[z_n, y_n] = \frac{f(z_n)-f(y_n)}{z_n-y_n},$$

and

$$f[z_n, x_n, x_n] = \frac{f[z_n,x_n]-f'(x_n)}{z_n-x_n}.$$

Remark In the third substep, the second term on the right is similar to King's method correction with $\beta = 3$. It is possible to use other value of β without affecting the order.

- WL829, a family of methods by Wang and Liu [8]

$$\begin{aligned} y_n &= x_n - v_n, \\ z_n &= x_n - v_n G(t_n), \\ x_{n+1} &= z_n - \frac{f(z_n)}{f'(x_n)}(H(t_n) + V(t_n)W(u_n)), \end{aligned} \tag{14}$$

where the weight functions are

$$\begin{aligned} G(t_n) &= \frac{1-t_n}{1-2t_n}, \\ H(t_n) &= \frac{5-2t_n+t_n^2}{5-12t_n}, \\ V(t_n) &= 1+4t_n, \\ W(u_n) &= u_n. \end{aligned} \tag{15}$$

We also include the method by Wang and Liu [9] denoted WL8M which showed good results in our previous work (see, for example, Chun et al. [10])

$$\begin{aligned} y_n &= x_n - v_n, \\ z_n &= y_n - \frac{f(y_n)}{f'(x_n)}\frac{1}{1-2t_n}, \\ x_{n+1} &= z_n - \frac{f(z_n)}{2(f[x_n,z_n]-f[x_n,y_n])+f[y_n,z_n]+\frac{y_n-z_n}{y_n-x_n}(f[x_n,y_n]-f'(x_n))}. \end{aligned} \tag{16}$$

2 Extraneous fixed points

In solving a nonlinear equation iteratively we are looking for fixed points which are zeros of the given nonlinear function. Many multipoint iterative methods have fixed points that are not zeros of the function of interest. Thus, it is imperative to investigate the number of extraneous fixed points, their location and their properties. In the family of methods studied by Lotfi et al. [5], we choose weight functions G, H, K as rational functions (see, for example, Chun et al. [11]) in the following form

$$G(u) = \frac{a_1+b_1u}{1+c_1u}, \tag{17}$$

$$H(t) = \frac{a_2+b_2t}{1+c_2t+d_2t^2}, \tag{18}$$

$$K(s) = \frac{a_3+b_3s}{1+c_3s}. \tag{19}$$

In order to satisfy the conditions (6)–(8), we have

$$G(u) = \frac{1+(c-1)u}{1+cu}, \tag{20}$$

$$H(t) = \frac{1}{1-2t-t^2}, \tag{21}$$

and

$$K(s) = \frac{2s}{1+gs}. \tag{22}$$

The parameters c and g can be chosen to position the extraneous fixed points on the imaginary axis or, at least, close to that axis, (see, for example, Chun and Neta [12]).

In order to find the extraneous fixed points, we rewrite the methods of interest in the form

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} H_f(x_n, y_n, z_n), \tag{23}$$

where the function H_f for LSSS method is given by

$$H_f(x_n, y_n, z_n) = 1 + \frac{f(y_n)}{f(x_n) - 2f(y_n)} + \frac{f(z_n)}{f(x_n)} \frac{H(t_n) + K(s_n)}{G(u_n)}, \tag{24}$$

with $G(u),\ H(t),\ K(s)$ are given in (20), (21), (22), respectively

We have searched the parameter space (c, g) and found that the extraneous fixed points are not on the imaginary axis. We have considered two measures of closeness to the imaginary axis and experimented with those members from the parameter space.

Let $E = \{z_1, z_2, ..., z_{n_{c,g}}\}$ be the set of the extraneous fixed points corresponding to the values given to c and g. We define

$$d(c, g) = \max_{z_i \in E} |Re(z_i)|. \tag{25}$$

We look for the parameters c and g which attain the minimum of $d(c, g)$. This minimum occurs at $c = -0.4,\ g = 1.5$. We will call the corresponding method LSSS4.

Another method to choose the parameters is by considering the stability of $z \in E$ defined by

$$dq(z) = \frac{dq}{dz}(z), \tag{26}$$

where q is the iteration function of (23). We define a function called the averaged stabilty value of the set E by

$$A(c, g) = \frac{\sum_{z_i \in E} |dq(z_i)|}{n_{c,g}}. \tag{27}$$

The smaller A becomes, the less chaotic the basin of attraction tends to. The minimum of $A(c, g)$ occurs at $c = 2.6,\ g = 3.2$. We will call the corresponding method LSSS5.

3 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 [13] and continued in the work of Amat et al. [14–17], Scott et al. [18], Chun et al. [19], Chun and Neta [20], Chicharro et al. [21], Cordero et al. [22], Neta et al. [23, 24], Argyros and Magreñan [25], Magreñan [26] and Chun et al. [27]. The only papers comparing

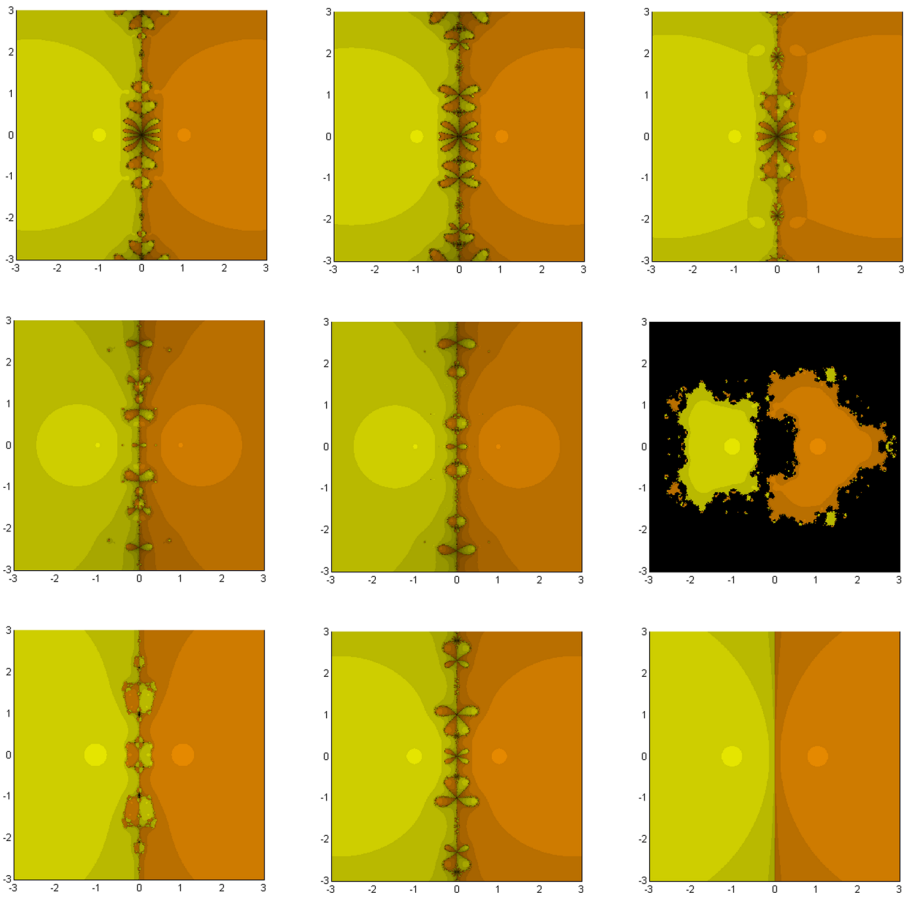

Fig. 1 The top left for LSSS1, top center for LSSS2 and top right for LSSS3, second row left for LSSS4, second row center for LSSS5 and right for BCST, third row left for CFGT, third row center for WL829 and third row right for WL8M for the roots of the polynomial $z^2 - 1$

basins of attraction for methods to obtain multiple roots are due to Neta et al. [28], Neta and Chun [29, 30], Chun and Neta [31, 32].

We have used the 9 methods LSSS1, LSSS2, LSSS3, LSSS4, LSSS5, BCST, CFGT, WL829 and WL8M for 5 different polynomials. The choice of the parameters in the families used is based on the analysis in the previous section. 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 Optiplex 990 desktop computer. We then computed the average number of iterations required per point and the standard deviation.

Table 1 Average number of iterations per point for each example (1–5) and each of the 9 methods

Method	Ex1	Ex2	Ex3	Ex4	Ex5	Average
LSSS1	2.76	5.02	8.50	8.45	7.71	6.49
LSSS2	2.98	5.13	9.05	8.88	8.42	6.89
LSSS3	2.61	4.41	7.14	7.34	7.09	5.72
LSSS4	3.26	4.41	6.22	6.72	5.74	5.27
LSSS5	3.27	4.25	5.95	6.44	5.42	5.06
BCST	28.62	16.72	12.70	12.57	11.14	16.35
CFGT	2.35	5.21	4.23	4.34	4.00	4.03
WL829	2.75	3.98	6.41	6.47	5.70	5.06
WL8M	2.27	2.71	3.52	4.04	3.44	3.19

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

$$p_1(z) = z^2 - 1 \tag{28}$$

whose roots are $z = \pm 1$. In Fig. 1 we have presented the basins for the 9 methods. The top left plot is for method LSSS1, the top center for LSSS2, and the top right for LSSS3. The second row left for LSSS4, second row center for LSSS5, second row right for BCST. The bottom row left is for CFGT, the bottom row center for WL8M, and the bottom right for WL829. It is clear that WL8M outperformed the others. This is a method we have added to the paper by Lotfi et al. [5]. The basins are separated by a straight line in the middle for WL8M without any black points (where the method used the maximum number of iterations). CFGT is second best. The plots only give a qualitative comparison. In our previous work (see [10, 32–34]) we have introduced a quantitative method for comparison. In Table 1 we can see that the average number of iterations per point is very close to 2 for the best method and above 28 for BCST. The methods by Lotfi et al. [5] and WL829 used about 3 iterations per point. This says that all methods except BCST are equivalent. 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, the worst is BCST. In Table 3 we find the CPU time required to run each

Table 2 Standard deviation for each example (1–5) and each of the 9 methods

Method	Ex1	Ex2	Ex3	Ex4	Ex5	Average
LSSS1	2.00	5.90	10.75	10.78	9.50	7.79
LSSS2	2.21	5.41	10.59	10.70	9.79	7.74
LSSS3	1.85	4.49	9.02	9.29	8.47	6.63
LSSS4	1.82	4.49	5.70	6.97	5.74	4.94
LSSS5	1.85	2.58	5.60	6.77	5.42	4.44
BCST	17.27	17.94	15.97	15.65	14.64	16.29
CFGT	1.77	8.86	3.35	2.88	2.15	3.80
WL829	1.92	3.28	7.58	8.03	6.04	5.37
WL8M	1.61	0.91	2.58	2.91	1.24	1.85

Table 3 CPU time (in seconds) required for each example (1–5) and each of the 9 methods

Method	Ex1	Ex2	Ex3	Ex4	Ex5	Average
LSSS1	286.65	706.46	1218.64	1501.99	5669.51	1876.65
LSSS2	321.72	728.10	1365.11	1574.86	6793.46	2156.65
LSSS3	319.06	747.63	1232.87	1530.82	6769.66	2120.01
LSSS4	290.19	747.63	788.15	987.10	3365.02	1235.62
LSSS5	301.21	506.61	760.35	928.77	3146.94	1128.78
BCST	1389.81	1137.16	1016.93	1143.86	4064.09	1750.37
CFGT	218.01	616.50	528.79	629.54	2112.00	820.97
WL829	205.46	379.80	637.56	750.11	2318.14	858.21
WL8M	184.82	294.54	404.83	599.11	1888.92	674.45

method on all the 360,000 points. The same conclusion is reached, namely BCST is the slowest (1389 s). We can also see that WL8M, WL829 and CFGT were the fastest (184–218 s). Table 4 gives the number of points for which a method requires 40 iterations. Here we see that again BCST is the worst (having 251,744 points) and the rest have 601–681 points out of 360,000.

Remark In [5] the basins have some black dots in the basins that should not be there. The reason for those black points is that the code did not check for convergence after every sub-step. We had that problem in a previous article [24] which we are in the process of correcting.

Example 2 Our next example is a cubic polynomial having the three roots of unity,

$$p_2(z) = z^3 - 1. \tag{29}$$

The basins of attraction are plotted in Fig. 2. Now it is clear that the best performer is WL8M and the worst is BCST. The methods LSSS4 and LSSS5 performed better

Table 4 Number of points using 40 iterations for each example (1–5) and each of the 9 methods

Method	Ex1	Ex2	Ex3	Ex4	Ex5	Average
LSSS1	601	3044	23617	27870	16810	14388.4
LSSS2	601	1625	20693	25450	16097	12893.2
LSSS3	601	995	13449	17963	11257	8853.0
LSSS4	601	69	3173	7352	640	2367
LSSS5	601	172	3453	7366	533	2425
BCST	251744	134451	91799	88327	73313	127926.8
CFGT	681	17588	1313	81	88	3950.2
WL829	601	163	7093	11150	2667	4334.8
WL8M	601	1	1201	19	0	364.4

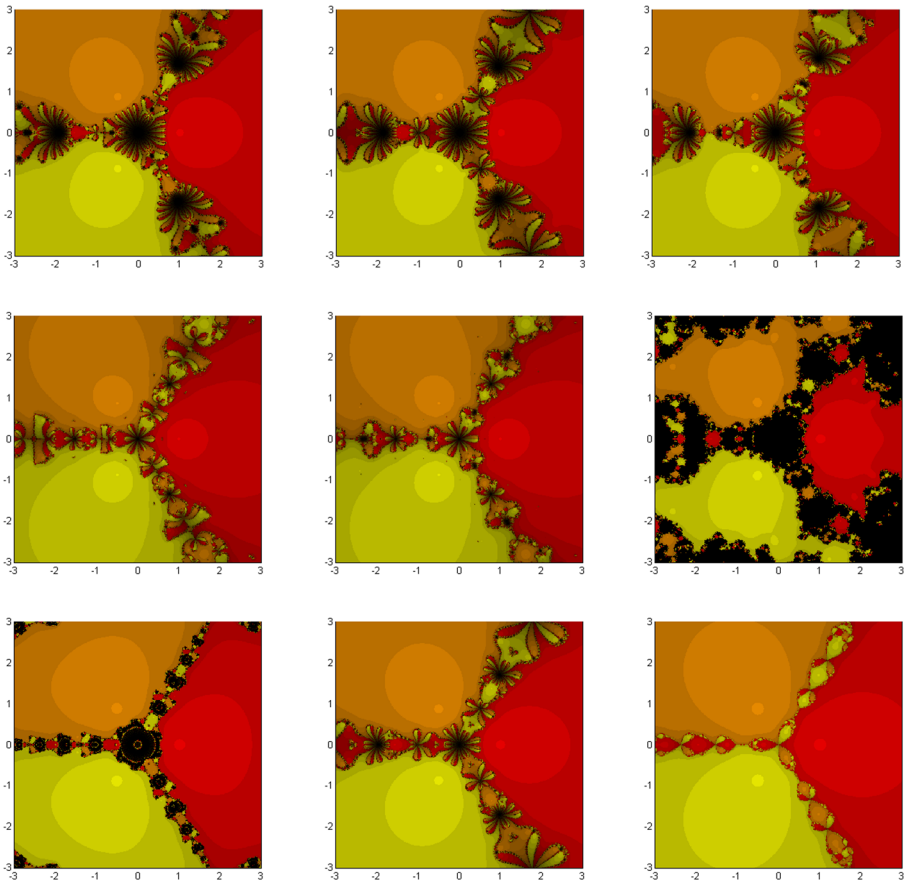

Fig. 2 The top left for LSSS1, top center for LSSS2 and top right for LSSS3, second row left for LSSS4, second row center for LSSS5 and right for BCST, third row left for CFGT, third row center for WL829 and third row right for WL8M for the roots of the polynomial $z^3 - 1$

than the other members of the family by Lotfi et al. [5]. This indicated that our idea of choosing weight function so that the extraneous fixed points are close to the imaginary axis is useful. In terms of average number of iterations per point (see Table 1) the best is WL8M followed by WL829 and LSSS5. There is no difference between LSSS4 and LSSS3. The same conclusion one arrives at when inspecting the CPU time required (Table 3). As to the number of points requiring 40 iterations (Table 4) it is clear that the best is WL8M followed by LSSS4, WL829 and LSSS5. The worst is BCST with 134,451 points requiring 40 iterations.

Example 3 Our next example is a quartic polynomial having the four roots of unity,

$$p_3(z) = z^4 - 1, \tag{30}$$

where the roots are symmetrically located on the axes. In some sense this is similar to the first example, since in both cases we have an even number of roots. The basins

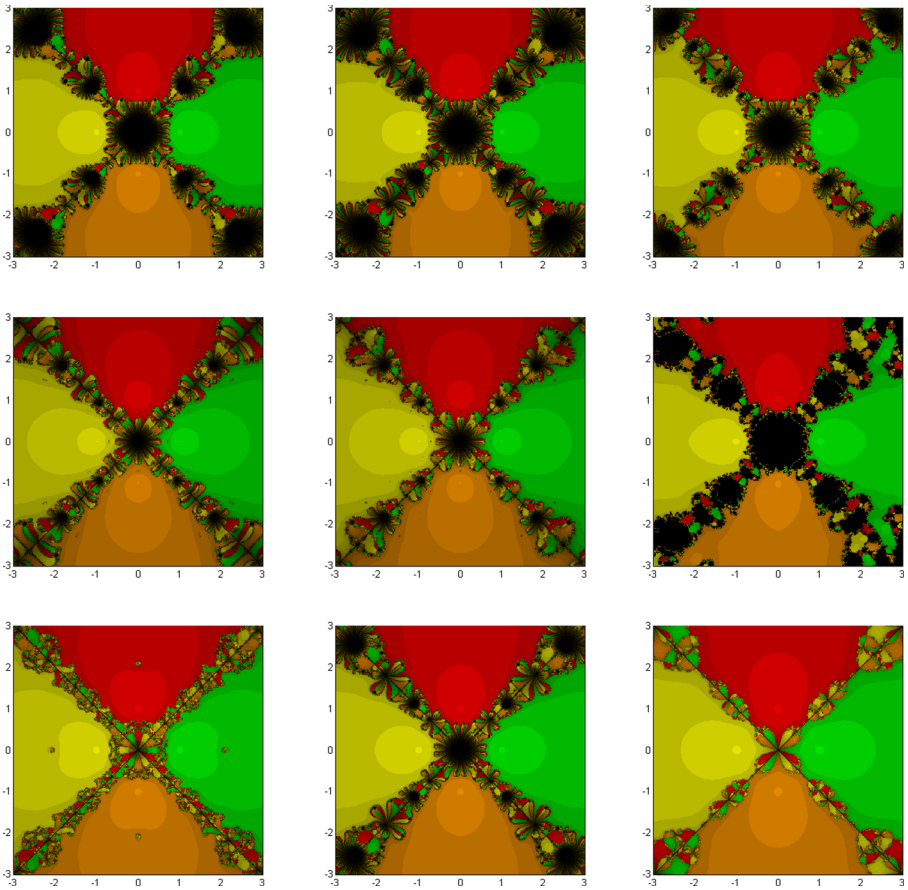

Fig. 3 The top left for LSSS1, top center for LSSS2 and top right for LSSS3, second row left for LSSS4, second row center for LSSS5 and right for BCST, third row left for CFGT, third row center for WL829 and third row right for WL8M for the roots of the polynomial $z^4 - 1$

of attraction are given in Fig. 3. The best methods are WL8M and CFGT. Based on the average number of iterations per point we can say that WL8M is best followed by CFGT, LSSS5 and LSSS4 in that order. The same conclusion from Table 4. In terms of CPU we have WL8M fastest followed by CFGT. but now WL829 is faster than LSSS4 and LSSS5.

Example 4 The fourth example is a polynomial

$$p_4(z) = z^5 - 1. \tag{31}$$

The plots of the basins are given in Fig. 4. The best method is again WL8M followed by CFGT, WL829 and LSSS5 (see Tables 1 and 3). In terms of the number

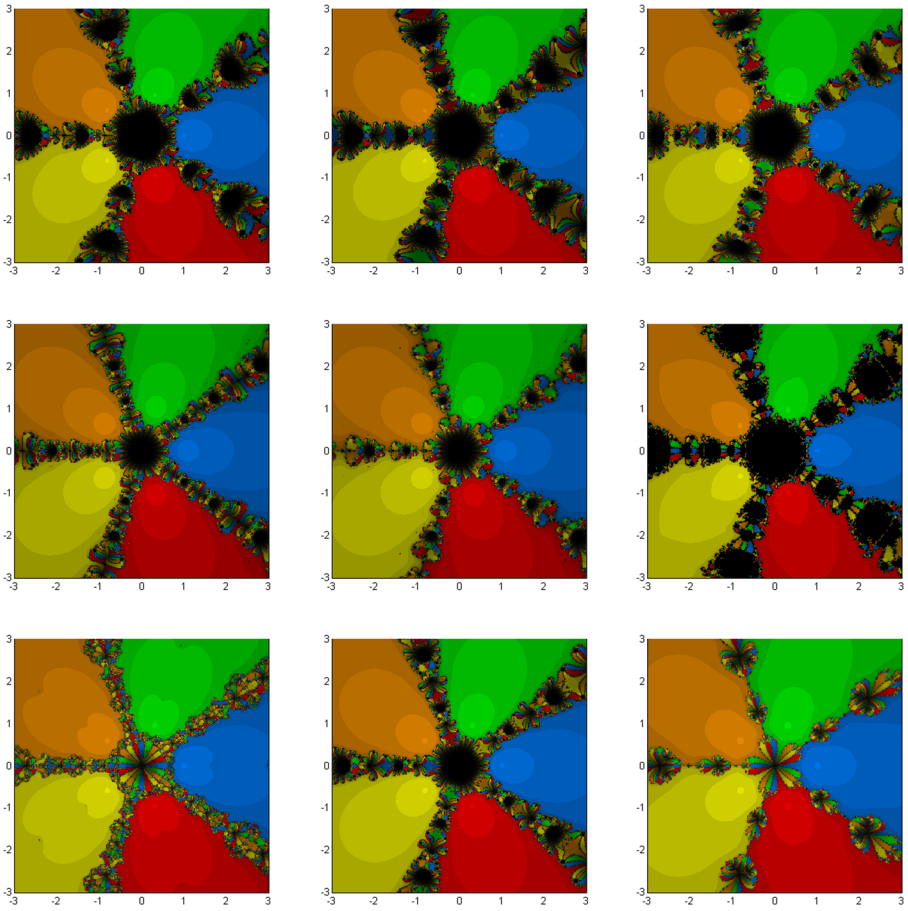

Fig. 4 The top left for LSSS1, top center for LSSS2 and top right for LSSS3, second row left for LSSS4, second row center for LSSS5 and right for BCST, third row left for CFGT, third row center for WL829 and third row right for WL8M for the roots of the polynomial $z^5 - 1$

of points requiring 40 iterations (Table 4), we see the same order except that LSSS4 is much better than WL829.

Example 5 Our last example is a polynomial with complex coefficients

$$p_5(z) = z^6 - \frac{1}{2}z^5 + \frac{11}{4}(1+i)z^4 - \frac{1}{4}(19+3i)z^3 + \frac{1}{4}(11+5i)z^2 - \frac{1}{4}(11+i)z + \frac{3}{2} - 3i. \tag{32}$$

This example was the hardest for many iterative methods as we found out in our previous work. The same conclusions can be seen from Fig. 5. Based on Table 1 we conclude that WL8M is best followed by CFGT, LSSS5, WL829 and LSSS4 in that order. Both LSSS4 and LSSS5 are better than the original methods used by Lotfi et al.

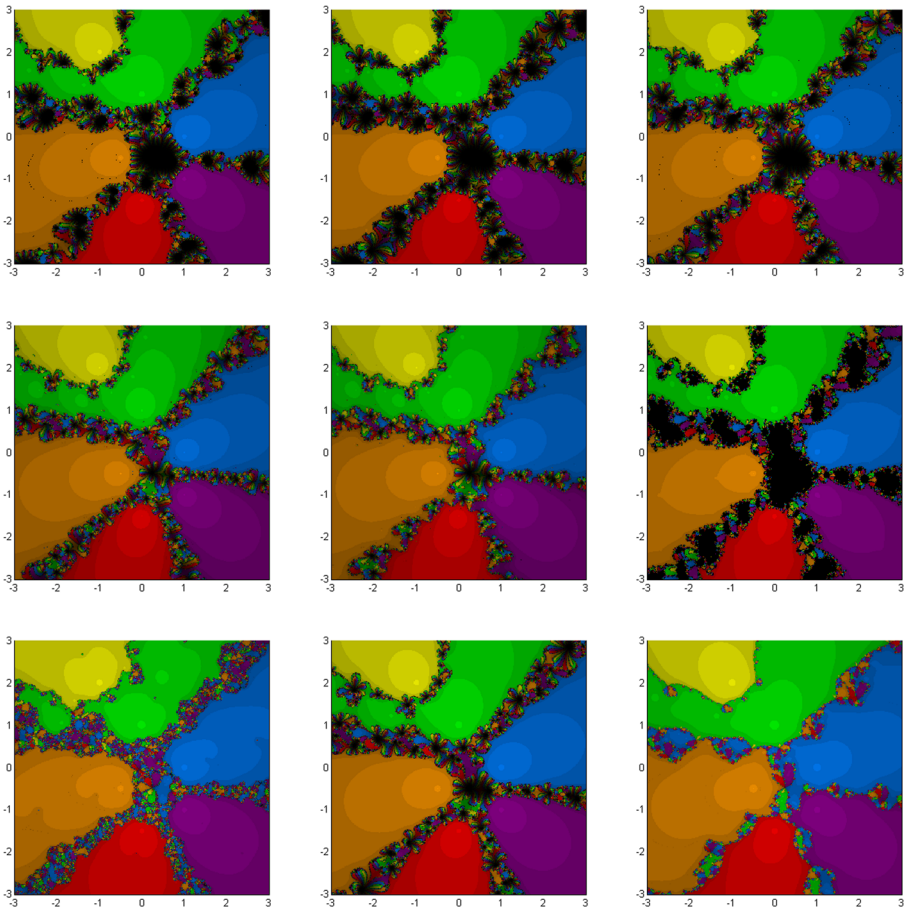

Fig. 5 The top left for LSSS1, top center for LSSS2 and top right for LSSS3, second row left for LSSS4, second row center for LSSS5 and right for BCST, third row left for CFGT, third row center for WL829 and third row right for WL8M for the roots of the polynomial with complex coefficients

[5]. The CPU time is much higher for all methods (see Table 3) but still WL8M was fastest followed by CFGT and WL829. Now LSSS1, LSSS2 and LSSS3 are slower even than BCST.

4 Conclusions

In Tables 1–4 we averaged the results across all 5 examples. Based on Table 1, we find that WL8M and CFGT are best (3–4 iterations per point). The others require 5–7 iterations except BCST which requires 16 iterations per point. The fastest method on average is WL8M (674 s) followed by CFGT (820 s) and WL829 (858 s). LSSS4 and LSSS5 are next with 1128–1235 s. Similar conclusions one can find in Table 4.

Overall we can say that WL8M is best and LSSS4 and LSSS5 are better than those suggested by Lotfi et al. [5].

Acknowledgments 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. Ostrowski, A.M.: Solution of Equations in Euclidean and Banach Space. Academic Press, New York (1973)
2. Traub, J.F.: Iterative Methods for the Solution of Equations. Chelsea Publishing Company, New York (1977)
3. Neta, B.: Numerical Methods for the Solution of Equations. Net-A-Sof, California (1983)
4. Petković, M.S., Neta, B., Petković, L.D., Džunić, J.: Multipoint Methods for Solving Nonlinear Equations. Elsevier, Waltham (2013)
5. Lotfi, T., Sharifi, S., Salimi, M., Siegmund, S.: A new class of three-point methods with optimal convergence order eight and its dynamics. Numer. Algor. **68**, 261–288 (2015)
6. Babajee, D.K.R., Cordero, A., Soleymani, F., Torregrosa, J.R.: On improved three-step schemes with high efficiency index and their dynamics. Numer. Algor. **65**, 153–169 (2014)
7. Cordero, A., Fardi, M., Ghasemi, M., Torregrosa, J.R.: Accelerated iterative methods for finding solutions of nonlinear equations and their dynamical behavior. Calcolo **51**, 17–30 (2014)
8. Wang, X., Liu, L.: New eighth-order iterative methods for solving nonlinear equations. J. Comput. Appl. Math. **234**, 1611–1620 (2010)
9. Wang, X., Liu, L.: Modified Ostrowski's method with eighth-order convergence and high efficiency index. Appl. Math. Lett. **23**, 549–554 (2010)
10. Chun, C., Neta, B.: An analysis of a King-based family of optimal eighth-order methods. Am. J. Algorithms Comput. **2**, 1–17 (2015)
11. Chun, C., Neta, B., Kozdon, J., Scott, M.: Choosing weight functions in iterative methods for simple roots. Appl. Math. Comput. **227**, 788–800 (2014)
12. Chun, C., Neta, B.: An analysis of a family of Maheshwari-based optimal eighth order methods. Appl. Math. Comput. **253**, 294–307 (2015)
13. Stewart, B.D.: Attractor Basins of Various Root-Finding Methods, M.S. thesis, Naval Postgraduate School, Department of Applied Mathematics, Monterey (2001)
14. Amat, S., Busquier, S., Plaza, S.: Iterative root-finding methods, unpublished report (2004)
15. Amat, S., Busquier, S., Plaza, S.: Review of some iterative root-finding methods from a dynamical point of view. Scientia **10**, 3–35 (2004)
16. Amat, S., Busquier, S., Plaza, S.: Dynamics of a family of third-order iterative methods that do not require using second derivatives. Appl. Math. Comput. **154**, 735–746 (2004)
17. Amat, S., Busquier, S., Plaza, S.: Dynamics of the King and Jarratt iterations. Aeq. Math. **69**, 212–223 (2005)
18. Scott, M., Neta, B., Chun, C.: Basin attractors for various methods. Appl. Math. Comput. **218**, 2584–2599 (2011)
19. Chun, C., Lee, M.Y., Neta, B., Džunić, J.: On optimal fourth-order iterative methods free from second derivative and their dynamics. Appl. Math. Comput. **218**, 6427–6438 (2012)
20. Chun, C., Neta, B.: An analysis of a new family of eighth-order optimal methods. Appl. Math. Comput. **245**, 86–107 (2014)
21. Chicharro, F., Cordero, A., Gutiérrez, J.M., Torregrosa, J.R.: Complex dynamics of derivative-free methods for nonlinear equations. Appl. Math. Comput. **219**, 7023–7035 (2013)
22. Cordero, A., García-Maimó, J., Torregrosa, J.R., Vassileva, M.P., Vindel, P.: Chaos in King's iterative family. Appl. Math. Lett. **26**, 842–848 (2013)
23. Neta, B., Scott, M., Chun, C.: Basin of attractions for several methods to find simple roots of nonlinear equations. Appl. Math. Comput. **218**, 10548–10556 (2012)

24. Neta, B., Chun, C., Scott, M.: Basins of attraction for optimal eighth order methods to find simple roots of nonlinear equations. Appl. Math. Comput. **227**, 567–592 (2014)
25. Argyros, I.K., Magreñan, A.A.: On the convergence of an optimal fourth-order family of methods and its dynamics. Appl. Math. Comput. **252**, 336–346 (2015)
26. Magreñan, A.A.: Different anomalies in a Jarratt family of iterative root-finding methods. Appl. Math. Comput. **233**, 29–38 (2014)
27. Chun, C., Neta, B., Kim, S.: On Jarratt's family of optimal fourth-order iterative methods and their dynamics. Fractals **22**, 1450013 (2014). doi:10.1142/S0218348X14500133
28. Neta, B., Scott, M., Chun, C.: Basin attractors for various methods for multiple roots. Appl. Math. Comput. **218**, 5043–5066 (2012)
29. Neta, B., Chun, C.: On a family of Laguerre methods to find multiple roots of nonlinear equations. Appl. Math. Comput. **219**, 10987–11004 (2013)
30. Neta, B., Chun, C.: Basins of attraction for several optimal fourth order methods for multiple roots. Math. Comput. Simulation **103**, 39–59 (2014)
31. Chun, C., Neta, B.: Basins of attraction for Zhou-Chen-Song fourth order family of methods for multiple roots. Math. Comput. Simul. **109**, 74–91 (2015)
32. Chun, C., Neta, B.: Comparing the basins of attraction for Kanwar-Bhatia-Kansal family to the best fourth order method. Appl. Math. Comput. **266**, 277–292 (2015)
33. Geum, Y.H., Kim, Y.I., Neta, B.: On developing a higher-order family of double-Newton methods with a bivariate weighting function. Appl. Math. Comput. **254**, 277–290 (2015)
34. Chun, C., Neta, B.: Basin of attraction for several third order methods to find multiple roots of nonlinear equations, *Appl. Math. Comput.*, accepted for publication