SeMA (2017) 74:613–625
https://doi.org/10.1007/s40324-016-0105-x

How good are methods with memory for the solution of nonlinear equations?

Changbum Chun[1] · **Beny Neta**[2]

Received: 15 November 2016 / Accepted: 12 December 2016 / Published online: 16 January 2017

Abstract Multipoint methods for the solution of a single nonlinear equation allow higher order of convergence without requiring higher derivatives. Such methods have an order barrier as conjectured by Kung and Traub. To overcome this barrier, one constructs multipoint methods with memory, i.e. use previously computed iterates. We compare multipoint methods with memory to the best methods without memory and show that the use of memory is computationally more expensive and the methods are not competitive.

Keywords Iterative methods with memory · Nonlinear equations · Simple roots · Order of convergence · Basin of attraction

Mathematics Subject Classification 65H10 · 47H99

1 Introduction

Many applications in science and engineering require the solution of a single nonlinear equation, for example to locate the candidates for extremum. A very well known iterative method is Newton's scheme which is of second order. There are many methods of higher order, see e.g. the books by Traub [31] and Petković et al. [27] and the comparative studies [6,7]. To develop higher order methods, one can use higher derivatives, such as in Halley [21] or use multistep methods. The multistep methods without memory have the barrier as conjectured by Kung and Traub [23] that a method using $r + 1$ function evaluations per step

✉ Beny Neta
bneta@nps.edu

Changbum Chun
cbchun@skku.edu

[1] Department of Mathematics, Sungkyunkwan University, Suwon 16419, Republic of Korea

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

can have order 2^r. In order to overcome the barrier of so called optimality, one can develop methods with memory, see e.g Chapter 6 of [31] or Chapter 6 of the more recent book [27].

Recall that Traub classified iterative methods with memory in the following way:

1. Let x_{k+1} be determined by new information at x_k and reused information at $x_{k-1}, \ldots, x_{k-p}$ by the iterative process
$$x_{k+1} = \phi(x_k;\ x_{k-1}, \ldots, x_{k-p}). \tag{1}$$
Then ϕ is called a *one-point iteration function with memory*, which defines an iterative *method with memory*.
2. Let z_j represent $p+1$ quantities $x_j, \omega_1(x_j), \ldots, \omega_p(x_j)$ $(j \geq 1)$. If x_{k+1} is calculated iteratively by
$$x_{k+1} = \phi(z_k, z_{k-1}, \ldots, z_{k-p}), \tag{2}$$
then ϕ is called a *multipoint iteration function with memory*.

Here we compare two multipoint methods with memory with the best multipoint methods without memory (see [7]).

To estimate the convergence rate of the family of multipoint iterative methods (2) with memory, we will use the concept of R-order of convergence introduced by Ortega and Rheinboldt [26].

In the next section we list the two multpoint methods with memory to be evaluated and compare to the best two methods without memory. We will experiment with these four methods and discuss the basins of attraction for each one. The idea of basin of attraction for comparative study was used by Stewart [30] and followed by the work of Amat et al. [1–3], Argyros and Magreñan [4], Chun et al. [9,10], Chun and Neta [8,11–14], Chicharro et al. [5], Cordero et al. [15], Geum et al. [17–20], Neta et al. [24,25] and Scott et al. [28].

In the next section we introduced the four methods and discuss the implementaion. In Sect. 3, we present the numerical results and the basins of attractions for the methods ran on seven examples. We close with concluding remark.

2 Methods for comparison

As we mentioned previously, we will compare two methods with memory to the best two methods without memory. The methods with their order of convergence (p), number of function- (and derivative-) evaluations per step (ν) and efficiency (I) are

1. Chun et al.'s method [9] $(p = 4,\ \nu = 3,\ I = 1.5874)$, denoted CLND
$$y_n = x_n - \frac{2}{3}\frac{f(x_n)}{f'(x_n)}, \tag{3}$$
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} H(\tilde{t}(x_n)), \tag{4}$$
where the weight function H satisfies $H(0) = 1,\ H'(0) = \frac{1}{2},\ H''(0) = 1$, and
$$\tilde{t}(x_n) = \frac{3}{2}\frac{f'(x_n) - f'(y_n)}{f'(x_n)}. \tag{5}$$
CLND is the case where the weight function $H(t)$ in (4) is given by
$$H(t) = \frac{1 + (2g - 2c - 1/2)t + gt^2}{1 + (2g - 2c - 1)t + ct^2} \tag{6}$$

with $c = 0,\ g = 0$, which is basically Jarratt's fourth-order (J4) method [22]

$$x_{n+1} = x_n - \left[1 - \frac{3}{2}\frac{f'(y_n) - f'(x_n)}{3f'(y_n) - f'(x_n)}\right]\frac{f(x_n)}{f'(x_n)}, \tag{7}$$

where y_n is given by (3).

2. Sharma–Arora's method [29] ($p = 8,\ \nu = 4,\ I = 1.6818$), denoted SA8

$$\begin{aligned} y_n &= x_n - \frac{f(x_n)}{f'(x_n)}, \\ z_n &= \phi_4(x_n, y_n), \\ x_{n+1} &= z_n - \frac{f[z_n, y_n]}{f[z_n, x_n]}\frac{f(z_n)}{2f[z_n, y_n] - f[z_n, x_n]}, \end{aligned} \tag{8}$$

where

$$\phi_4(x_n, y_n) = y_n - \frac{f(y_n)}{2f[y_n, x_n] - f'(x_n)}. \tag{9}$$

3. Ullah et al.'s method [32] (R-order 7.94449, $\nu = 3,\ I = 1.99536$), denoted UKSHA,

$$\begin{aligned} &\beta_n = -\frac{1}{N_6'(x_n)}, \quad p_n = -\frac{N_7''(w_n)}{2N_7'(w_n)}, \quad \lambda_n = \frac{1}{6}N_8'''(y_n), \quad n \geq 2, \\ &y_n = x_n - \frac{f(x_n)}{f[x_n, w_n] + p_n f(w_n)}, \quad w_n = x_n + \beta_n f(x_n), \quad n \geq 0, \\ &x_{n+1} = y_n - \frac{f(y_n)}{f[x_n, y_n] + f[w_n, x_n, y_n](y_n - x_n) + \lambda_n(y_n - x_n)(y_n - w_n)}, \end{aligned} \tag{10}$$

where

$$N_6(t) = N_6(t; x_n, y_{n-1}, w_{n-1}, x_{n-1}, y_{n-2}, w_{n-2}, x_{n-2}), \tag{11}$$

is an interpolaion polynomial of sixth degree, passing through $x_n, y_{n-1}, w_{n-1}, x_{n-1}, y_{n-2}, w_{n-2}, x_{n-2}$,

$$N_7(t) = N_7(t; w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1}, y_{n-2}, w_{n-2}, x_{n-2}), \tag{12}$$

is an interpolaion polynomial of seventh degree, passing through $w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1}, y_{n-2}, w_{n-2}, x_{n-2}$, and

$$N_8(t) = N_8(t; y_n, w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1}, y_{n-2}, w_{n-2}, x_{n-2}), \tag{13}$$

is an interpolaion polynomial of eighth degree, passing through $y_n, w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1}, y_{n-2}, w_{n-2}, x_{n-2}$.

In the case where $n = 1$, this method uses

$$\beta_n = -\frac{1}{N_3'(x_n)}, \quad p_n = -\frac{N_4''(w_n)}{2N_4'(w_n)}, \quad \lambda_n = \frac{1}{6}N_5'''(y_n), \tag{14}$$

where

$$N_3(t) = N_3(t; x_n, y_{n-1}, w_{n-1}, x_{n-1}), \tag{15}$$

is an interpolaion polynomial of third degree, passing through $x_n, y_{n-1}, w_{n-1}, x_{n-1}$,

$$N_4(t) = N_4(t; w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1}), \tag{16}$$

Table 1 Average number of function evaluations per point for each example (1–7) and each of the methods

Method	Ex1	Ex2	Ex3	Ex4	Ex5	Ex6	Ex7	Average
CLND	9.59	11.19	11.85	10.19	14.50	13.49	9.34	11.45
SA8	8.65	9.68	10.46	10.20	12.11	11.57	9.12	10.25
UKSHA	6.15	37.7	6.68	6.53	29.94	25.23	10.70	17.56
DPP	5.44	23.48	8.15	6.54	39.49	23.08	6.41	16.08

Table 2 CPU time (in seconds) required for each example (1–7) and each of the methods

Method	Ex1	Ex2	Ex3	Ex4	Ex5	Ex6	Ex7	Average
CLND	164.752	257.542	275.529	309.646	426.382	1216.745	361.438	430.290
SA8	152.381	224.969	241.178	318.273	359.240	1272.032	333.967	414.577
UKSHA	992.151	6452.779	1385.539	1522.46	5477.227	10276.925	2141.753	4035.547
DPP	283.454	787.462	470.015	515.661	1302.842	4538.803	568.967	1209.600

Table 3 Number of points requiring 40 iterations for each example (1–7) and each of the methods

Method	Ex1	Ex2	Ex3	Ex4	Ex5	Ex6	Ex7	Average
CLND	601	1	0	601	16	0	627	263.71
SA8	601	1	0	601	1	0	514	245.42
UKSHA	184	98,876	56	334	55,620	40,876	13,081	29,861
DPP	29	34,834	166	896	67,682	26,307	1328	18,748.85

is an interpolaion polynomial of fourth degree, passing through $w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1}$, and

$$N_5(t) = N_5(t; y_n, w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1}), \tag{17}$$

is an interpolaion polynomial of fifth degree, passing through $y_n, w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1}$.

In the case of $n = 0$, the initial approximations β_n, p_n, λ_n could be considered as very small positive values.

4. Džunić et al.'s method [16] (R-order $2(2+\sqrt{5}) \approx 8.47$, $\nu = 4$, $I = 2.86926$), denoted DPP,

$$\begin{aligned}
\gamma_n &= -\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}, \quad n \geq 0,\\
y_n &= x_n - \frac{f(x_n)}{\phi_n}, \quad w_n = x_n + \gamma_n f(x_n), \quad n \geq 0,\\
z_n &= y_n - h(s_n, v_n)\frac{f(y_n)}{\phi_n}, \quad n \geq 0,\\
x_{n+1} &= z_n - \frac{f(z_n)}{N_3'(z_n; z_n, y_n, x_n, w_n)}, \quad n \geq 0,
\end{aligned} \tag{18}$$

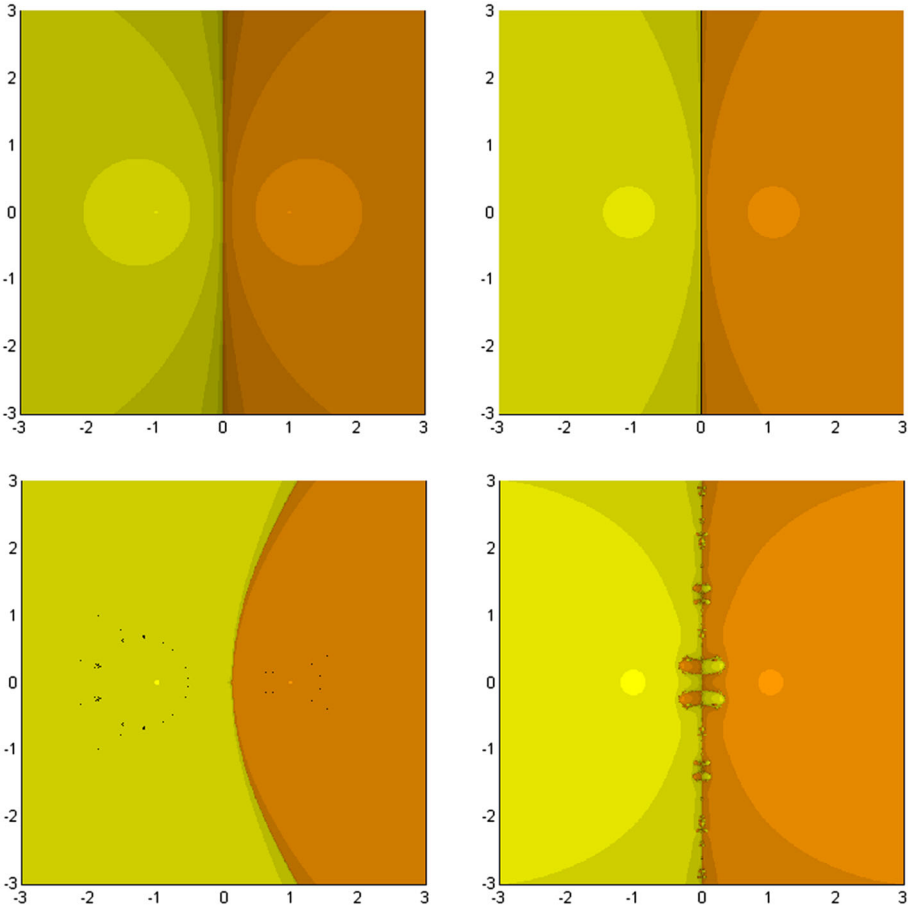

Fig. 1 The *top row* for CLND (*left*) and SA8 (*right*). *Second row* for UKSHA (*left*) and DPP (*right*) for the roots of the polynomial $z^2 - 1$

where ϕ_n is defined by

$$\phi_n = \frac{f(w_n) - f(x_n)}{\gamma_n f(x_n)}, \tag{19}$$

h is a weight function of two variables that satisfies $h(0,0) = h_s(0,0) = h_v(0,0) = 1, h_{vv}(0,0) = 2$, $s_n = \frac{f(y_n)}{f(x_n)}$, $v_n = \frac{f(y_n)}{f(w_n)}$, and $N_3'(z_n; z_n, y_n, x_n, w_n)$ is the derivative of Newton's interpolating polynomial of degree three at the points z_n, y_n, x_n, and w_n evaluated at z_n, which is given by

$$\begin{aligned} N_3'(z_n; z_n, y_n, x_n, w_n) = {} & f[z_n, y_n] + f[z_n, y_n, x_n](z_n - y_n) \\ & + f[z_n, y_n, x_n, w_n](z_n - y_n)(z_n - x_n). \end{aligned} \tag{20}$$

For the function h, we experimented with $h(s, v) = \frac{1+s}{1+v}$. Given x_{-1}, we ran the method with γ_n taken as a very small positive value to find an additional starting value x_0.

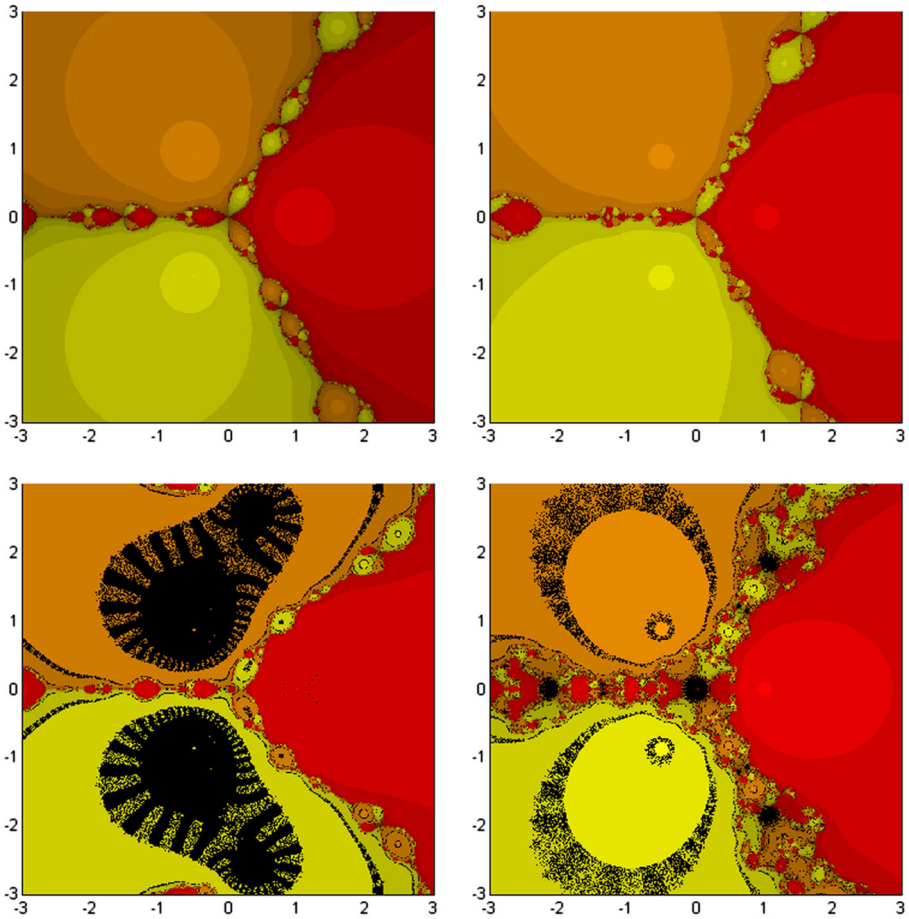

Fig. 2 The *top row* for CLND (*left*) and SA8 (*right*). Second row for UKSHA (*left*) and DPP (*right*) for the roots of the polynomial $z^3 - 1$

3 Numerical experiments

In this section, we detail the experiments we have used with each of the methods. All the examples have roots within a square of $[-3, 3]$ by $[-3, 3]$. We have taken 601^2 equally spaced points in the square as initial points for the methods and we have registered the total number of function-evaluations per point on average (NFEA) required to converge to a root (in Table 1) 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 (see Table 2) and the number of points requiring 40 iterations in Table 3. These points are painted black and we refer to them as black points or NBP.

Example 1 The first example is the quadratic polynomial

$$p_1(z) = z^2 - 1 \tag{21}$$

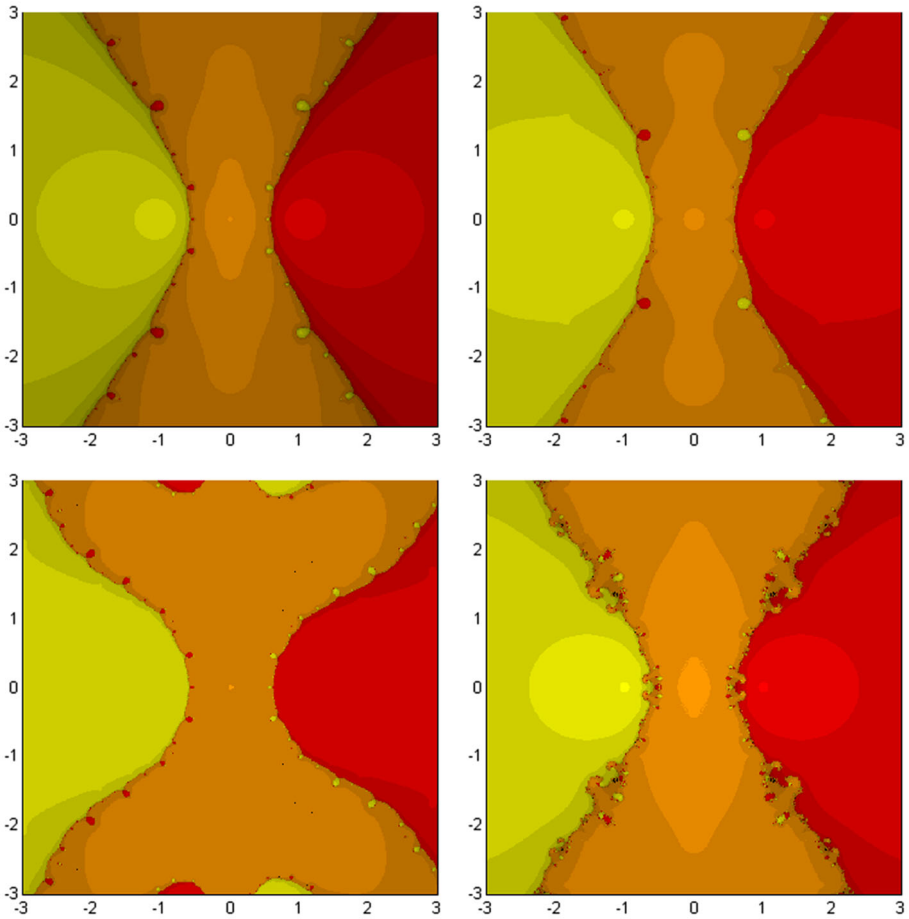

Fig. 3 The *top row* for CLND (*left*) and SA8 (*right*). *Second row* for UKSHA (*left*) and DPP (*right*) for the roots of the polynomial $z^3 - z$

whose roots are at ± 1. The basins are given in Fig. 1. The top row shows the basins of the methods without memory and the bottom for those with memory. It is clear that the best methods are those without memory, since the domain is divided equally by the vertical axis. DPP is better than UKSHA, since there is no preference to the root $z = -1$ over the other. For a more quantitative comparison, we refer to the Tables 1, 2 and 3. In Table 1 we have compared the NFEA. In Table 2 we compared the CPU time in seconds to run the method on all 601^2 points and in Table 3 we listed the number of points for which the method did not converge after 40 iterations (NBP). The CPU results show that the DPP is much faster than UKSHA and slower than the methods without memory. DPP has also the lowest NBP and the lowest NFEA. This seems encouraging for multipoint methods with memory.

Example 2 The second example is the cubic polynomial

$$p_2(z) = z^3 - 1 \tag{22}$$

having the 3 roots of unity.

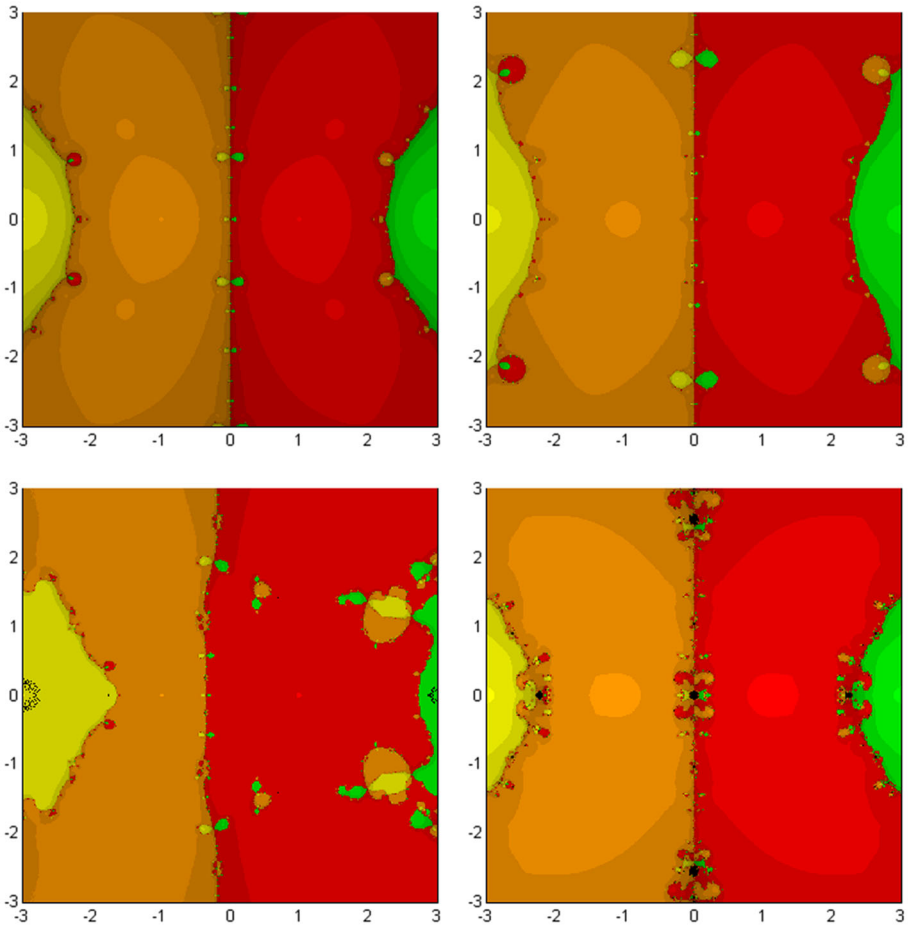

Fig. 4 The *top row* for CLND (*left*) and SA8 (*right*). *Second row* for UKSHA (*left*) and DPP (*right*) for the roots of the polynomial $z^4 - 10z^2 + 9$

The basins of attraction are given in Fig. 2. Now we see that multipoint methods with memory have many black points. It could be that when the roots are not real, the methods have difficulty. We will check that in the rest of experiments. Based on Table 1 we find that SA8 has the lowest NFEA. SA8 is the fastest (224.969 s) and has only one black point (exactly as CLND).

Example 3 The third example is another cubic polynomial, but with real roots only, i.e. the polynomial is given by:

$$p_3(z) = z^3 - z. \tag{23}$$

The basins of attraction are displayed in Fig. 3. All methods look reasonable. It is possible that the fact that all roots are real as in Example 1 that we do not have many black points for the methods with memory. UKSHA has the lowest NFEA but took more CPU (1385.539 s) than any other method. It is clear that each step of UKSHA is more computationally expensive than other methods. CLND and SA8 have no black points.

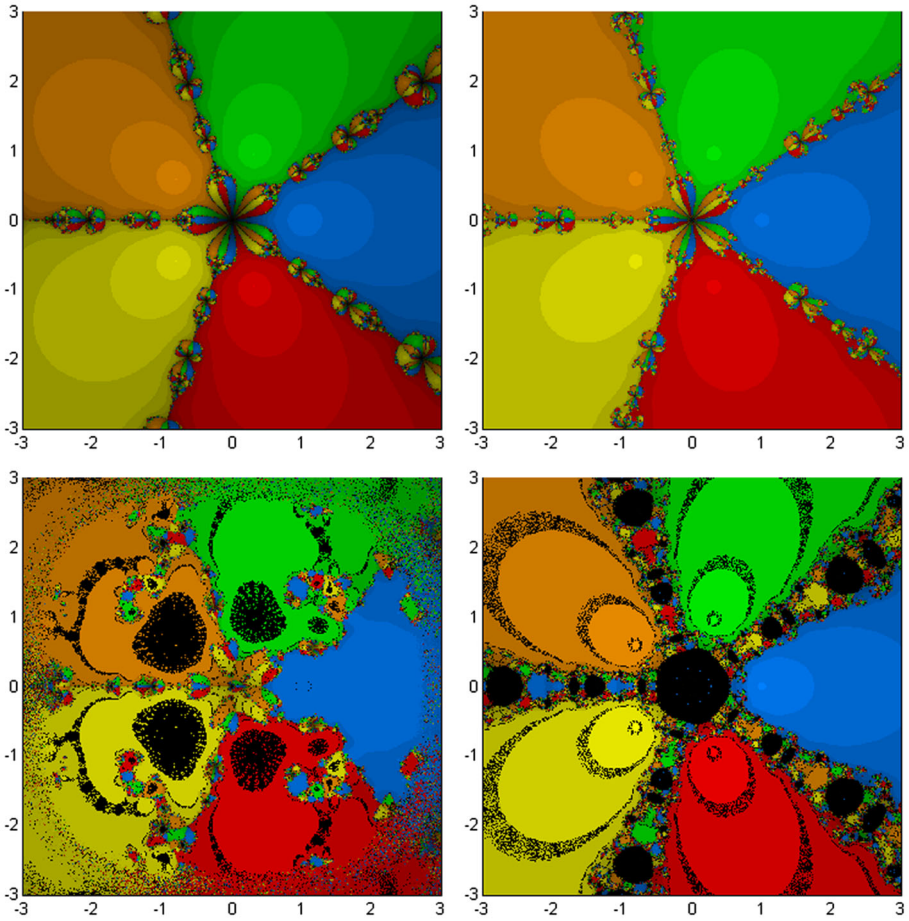

Fig. 5 The *top row* for CLND (*left*) and SA8 (*right*). *Second row* for UKSHA (*left*) and DPP (*right*) for the roots of the polynomial $z^5 - 1$

Example 4 The fourth example is a quartic polynomial with real roots at $\pm 1, \pm 3$

$$p_4(z) = z^4 - 10z^2 + 9. \tag{24}$$

The basins are displayed in Fig. 4. Again all the roots are real and the methods with memory do not have as many black points as in Example 2. The methods of memory use about the same NFEA and it is less than for CLND and SA8. In terms of CPU, CLND is fastest (309.646 s) followed by SA8 (318.273 s) and DPP (515.661 s). UKSHA has fewest black points.

Example 5 The fifth example is a fifth degree polynomial

$$p_5(z) = z^5 - 1. \tag{25}$$

The basins are displayed in Fig. 5. Now that the roots are not all real, we see more black points in the basins of methods with memory (see also Table 3). They also require higher NFEA and are very slow (over 1000 s versus around 400 s for SA8 and CLND).

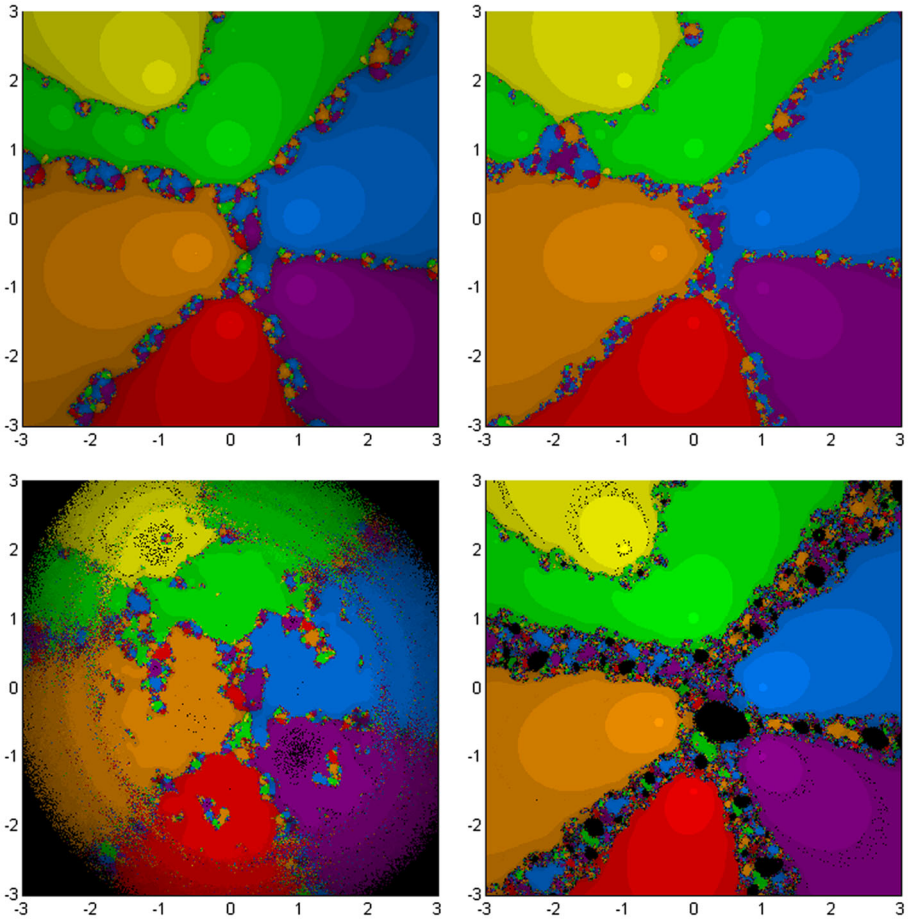

Fig. 6 The *top row* for CLND (*left*) and SA8 (*right*). *Second row* for UKSHA (*left*) and DPP (*right*) for the roots of the polynomial $z^6 - \frac{1}{2}z^5 + \frac{11(i+1)}{4}z^4 - \frac{3i+19}{4}z^3 + \frac{5i+11}{4}z^2 - \frac{i+11}{4}z + \frac{3}{2} - 3i$

Example 6 The next example is a polynomial of degree 6 with complex coefficients

$$p_6(z) = z^6 - \frac{1}{2}z^5 + \frac{11(i+1)}{4}z^4 - \frac{3i+19}{4}z^3 + \frac{5i+11}{4}z^2 - \frac{i+11}{4}z + \frac{3}{2} - 3i. \quad (26)$$

This is an example that was difficult for many methods. The basins are displayed in Fig. 6. It is clear that the basins for UKSHA are not as well defined as for the other methods. SA8 uses the least NFEA and UKSHA the most such number. The CPU time for methods without memory is about 1200 s versus DPP with 4538.803 s and UKSHA with 10276.925 s. CLND and SA8 have NO black points and UKSHA about twice the number of black points as DPP.

We now run a non-polynomial example.

Example 7

$$p_7(z) = (e^{z+1} - 1)(z - 1). \quad (27)$$

The roots are ± 1 and the basins are given in Fig. 7. Notice that in all methods the basin for $z = +1$ is much smaller. The basin for $z = +1$ is the largest for SA8. DPP uses 6.41 function

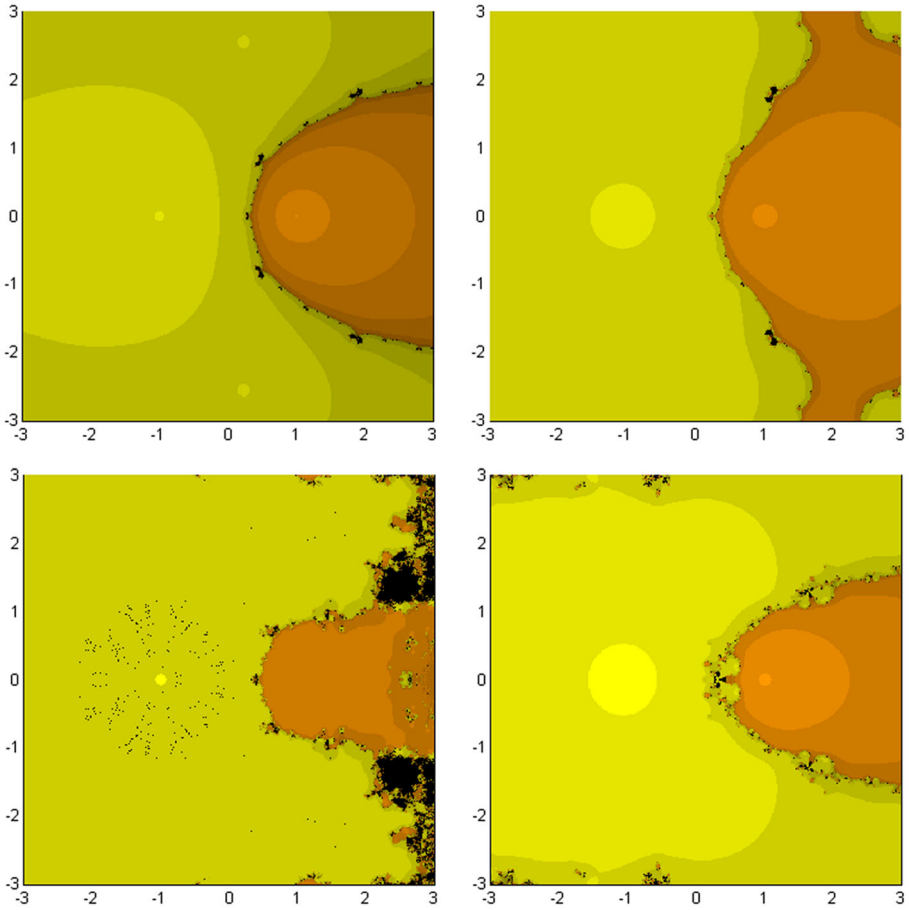

Fig. 7 The *top row* for CLND (*left*) and SA8 (*right*). *Second row* for UKSHA (*left*) and DPP (*right*) for the roots of the polynomial $(e^{z+1} - 1)(z - 1)$

evaluations per point and about 9 for SA8 and CLND. SA8 is the fastest followed closely by CLND and the slowest is UKSHA. In terms of the number of black points, it is clear that UKSHA has the most and the methods SA8 and CLND have the least.

In order to pick the best method overall, we have averaged the results in Tables 1, 2 and 3 across the seven examples. It is clear that SA8 uses the least NFEA (10.25) followed closely by CLND (11.45) and UKSHA uses the highest such number (17.56). The fastest method on average is SA8 (414.577 s) and the slowest is UKSHA (4035.547 s). Even DPP is much slower than SA8 and CLND. The average number of black points is the highest for methods with memory (over 18,000 versus 245–263 for SA8 and CLND, respectively).

Conclusions

We can see that the two methods with memory performed poorly when the function has complex roots. They are also computationally more expensive and require more function

evaluations per point on average. We thus do not recommend the use of multipoint methods with memory.

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-2016R1D1A1A09917373).

References

1. 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)
2. Amat, S., Busquier, S., Plaza, S.: Review of some iterative root-finding methods from a dynamical point of view. Scientia **10**, 3–35 (2004)
3. Amat, S., Busquier, S., Plaza, S.: Dynamics of the King and Jarratt iterations. Aeq. Math. **69**, 212–236 (2005)
4. 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)
5. 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)
6. Chun, C., Neta, B.: Comparative study of eighth order methods for finding simple roots of nonlinear equations. Numer. Algorithms (Accepted for publication)
7. Chun, C., Neta, B.: Comparative study of methods of various orders for finding simple roots of nonlinear equations (submitted for publication)
8. Chun, C., Neta, B.: The basins of attraction of Murakami's fifth order family of methods. Appl. Numer. Math. **110**, 14–25 (2016)
9. Chun, C., Lee, M.Y., Neta, B.: On optimal fourth-order iterative methods free from second derivative and their dynamics. Appl. Math. Comput. **218**, 6427–6438 (2012)
10. Chun, C., Neta, B., Kim, S.: On Jarratt's family of optimal fourth-order iterative methods and their dynamics. Fractals (2014). doi:10.1142/S0218348X14500133
11. Chun, C., Neta, B.: An analysis of a new family of eighth-order optimal methods. Appl. Math. Comput. **245**, 86–107 (2014)
12. Chun, C., Neta, B.: An analysis of a King-based family of optimal eighth-order methods. Am. J. Algorithms Comput. **2**, 1–17 (2015)
13. Chun, C., Neta, B.: On the new family of optimal eighth order methods developed by Lotfi, et al. Numer. Algorithms **72**, 363–376 (2016)
14. Chun, C., Neta, B.: Comparison of several families of optimal eighth order methods. Appl. Math. Comput. **274**, 762–773 (2016)
15. 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)
16. Džunić, J., Petković, M.S., Petković, L.D.: Three-point methods with and without memory for solving nonlinear equations. Appl. Math. Comput. **218**, 4917–4927 (2012)
17. Geum, Y.H., Kim, Y.I., Neta, B.: A family of optimal quartic-order multiple-zero finders with a weight function of the principal kth root of a derivative-to-derivative ratio and their basins of attraction. Math. Comput. Simul. (submitted for publication)
18. 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)
19. Geum, Y.H., Kim, Y.I., Neta, B.: A class of two-point sixth-order multiple-zero finders of modified double-Newton type and their dynamics. Appl. Math. Comput. **270**, 387–400 (2015)
20. Geum, Y.H., Kim, Y.I., Neta, B.: A sixth-order family of three-point modified Newton-like multiple-zero finders and the dynamics behind their extraneous fixed points. Appl. Math. Comput. **283**, 120–140 (2016)
21. Halley, E.: A new, exact and easy method of finding the roots of equations generally and that without any previous reduction. Philos. Trans. R. Soc. Lond. **18**, 136–148 (1694)
22. Jarratt, P.: Some fourth-order multipoint iterative methods for solving equations. Math. Comput. **20**, 434–437 (1966)
23. Kung, H.T., Traub, J.F.: Optimal order of one-point and multipoint iterations. J. Assoc. Comput. Mach. **21**, 643–651 (1974)
24. 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)

25. Neta, B., Chun, C., Scott, M.: Basins of attractions for optimal eighth order methods to find simple roots of nonlinear equations. Appl. Math. Comput. **227**, 567–592 (2014)
26. Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York (1970)
27. Petković, M.S., Neta, B., Petković, L.D., Džunić, J.: Multipoint Methods for Solving Nonlinear Equations. Elsevier, Waltham (2013)
28. Scott, M., Neta, B., Chun, C.: Basin attractors for various methods. Appl. Math. Comput. **218**, 2584–2599 (2011)
29. Sharma, J.R., Arora, H.: A new family of optimal eighth order methods with dynamics for nonlinear equations. Appl. Math. Comput. **273**, 924–933 (2016)
30. Stewart, B.D.: Attractor basins of various root-finding methods. M.S. thesis, Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA (2001)
31. Traub, J.F.: Iterative Methods for the Solution of Equations. Prentice-Hall Inc., Englewood Cliffs (1964)
32. Ullah, M.Z., Kosari, S., Soleymani, F., Haghani, F.K., Al-Fhaid, A.S.: A super-fast tri-parametric iterative method with memory. Appl. Math. Comput. **289**, 486–491 (2016)