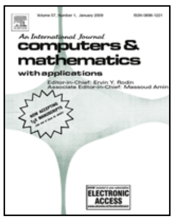
Computers and Mathematics with Applications 59 (2010) 2784–2793

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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



Accelerating generators of iterative methods for finding multiple roots of nonlinear equations[☆]

M.S. Petković[a,*], L.D. Petković[b], J. Džunić[a]

[a] *Faculty of Electronic Engineering, Department of Mathematics, University of Niš, 18000 Niš, Serbia*

[b] *Faculty of Mechanical Engineering, Department of Mathematics, University of Niš, 18000 Niš, Serbia*

ARTICLE INFO

Article history:
Received 30 July 2009
Received in revised form 19 January 2010
Accepted 19 January 2010

Keywords:
Solving nonlinear equations
Iterative methods
Multiple roots
Acceleration of convergence
Simultaneous methods

ABSTRACT

Two accelerating generators that produce iterative root-finding methods of arbitrary order of convergence are presented. Primary attention is paid to algorithms for finding multiple roots of nonlinear functions and, in particular, of algebraic polynomials. First, two classes of algorithms for solving nonlinear equations are studied: those with a known order of multiplicity and others with no information on multiplicity. We also demonstrate the acceleration of iterative methods for the simultaneous approximations of multiple roots of algebraic polynomials. A discussion about the computational efficiency of the root-solvers considered and three numerical examples are given.

1. Introduction

In this paper we consider various classes of iterative methods for finding multiple roots of nonlinear equations. We derive several new root-finding methods of a higher order for multiple roots of nonlinear functions and all multiple roots of algebraic polynomials in parallel mode. All of these methods are produced by suitable accelerating generators of iterative functions. An iterative method φ_{r+1} of order $r + 1$ is generated from the previous method φ_r of order r using a special transformation. The sequence of methods $\varphi_2, \varphi_3, \ldots, \varphi_n, \ldots$ can be derived automatically using symbolic computation (in *Mathematica, MATLAB* or *Maple*).

In Section 2 we present an accelerating formula for generating a sequence of iterative methods for determining multiple roots of equations of the form $f(x) = 0$. Using this generating formula, in Section 3 we derive several higher-order iterative methods, some of which are new ones. We distinguish two kinds of methods; those which deal with a known order of multiplicity and others with no information on multiplicity. In Section 4 we describe a new technique for generating some classes of iterative methods, based on the Weierstrass function, for the simultaneous determination of multiple roots of algebraic polynomials. Numerical examples are given in Section 5 to demonstrate the convergence behavior of the methods considered. This section also contains an analysis of the computational efficiency of the methods presented.

2. Accelerating generators of iterative methods

In this paper we shall restrict our attention to iterative methods of a higher order for finding multiple roots of nonlinear functions. To produce these methods, we use a suitable formula for the acceleration of convergence, which shall henceforth be termed the *accelerating generator*, or shorter $\mathcal{AG}$.

☆ This work was supported by the Serbian Ministry of Science under grant 144024.

* Corresponding author.
E-mail addresses: msp@junis.ni.ac.rs, msp@eunet.rs (M.S. Petković).

0898-1221/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2010.01.048

In our consideration we make use of the following well-known theorem from the theory of iterative processes.

Theorem 1 (*Traub [1, Theorem 2.2]*). *Let ϕ be an iterative function such that ϕ and its derivatives $\phi', \ldots, \phi^{(r)}$ are continuous in the neighborhood of a root α of a given function f. Then ϕ defines an iterative method of order r if and only if*

$$\varphi(\alpha) = \alpha, \qquad \varphi'(\alpha) = \cdots = \varphi^{(r-1)}(\alpha) = 0, \qquad \varphi^{(r)}(\alpha) \neq 0. \tag{1}$$

The following theorem is concerned with the acceleration of iterative methods, forming the base for generating higher-order methods for multiple roots.

Theorem 2. *Let $x_{k+1} = \varphi_r(x_k)$ $(k = 0, 1, \ldots)$ be an iterative method of order r for finding a simple or multiple root of a given function f (sufficiently many times differentiable). Then the iterative method defined by*

$$x_{k+1} = \varphi_{r+1}(x_k) := x_k - \frac{x_k - \varphi_r(x_k)}{1 - \frac{1}{r}\varphi_r'(x_k)} \quad (r \geq 2;\, k = 0, 1, \ldots), \tag{2}$$

has the order of convergence $r + 1$.

Proof. For two real or complex numbers z and w we will write $z = O_M(w)$ if $|z| = O(|w|)$ (the same order of their moduli), where O represents the Landau symbol.

Let us introduce the error $\varepsilon_k = x_k - \alpha$. Bearing in mind the relations (1), we find by Taylor's series

$$\varphi_r(x_k) = \alpha + \frac{1}{r!}\varphi_r^{(r)}(\alpha)\varepsilon_k^r + O_M(\varepsilon_k^{r+1}), \tag{3}$$

and

$$\varphi_r'(x_k) = \frac{1}{(r-1)!}\varphi_r^{(r)}(\alpha)\varepsilon_k^{r-1} + O_M(\varepsilon_k^r). \tag{4}$$

From (3) we see that

$$\varphi_r(x_k) - \alpha = O_M((x_k - \alpha)^r) = O_M(\varepsilon_k^r), \tag{5}$$

meaning that the method $x_{k+1} = \varphi_r(x_k)(k = 0, 1, \ldots)$ has order of convergence r (≥ 2), as assumed in Theorem 2.

Using (3) and (4) we obtain

$$\begin{aligned}\frac{1}{r}\varphi_r'(x_k)(x_k - \varphi_r(x_k)) &= \frac{1}{r}\varphi_r'(x_k)(x_k - \alpha - (\varphi_r(x_k) - \alpha)) \\ &= \frac{1}{r!}\varphi_r^{(r)}(\alpha)\varepsilon_k^r - \left(\frac{1}{r!}\varphi_r^{(r)}(\alpha)\right)^2 \varepsilon_k^{2r-1} + O_M(\varepsilon_k^{r+1}) \\ &= \frac{1}{r!}\varphi_r^{(r)}(\alpha)\varepsilon_k^r + O_M(\varepsilon_k^{r+1}).\end{aligned} \tag{6}$$

By virtue of (3) and (6), we get

$$\varphi_r(x_k) - \frac{1}{r}\varphi_r'(x_k)(x_k - \varphi_r(x_k)) = \alpha + O_M(\varepsilon_k^{r+1}). \tag{7}$$

In our analysis we assume that x_k is sufficiently close to the root α, meaning that $|\varepsilon_k| = |x_k - \alpha|$ is sufficiently small. According to (4) we conclude that $|\varphi_r'(x_k)| = O_M(|\varepsilon_k^{r-1}|)$ is also a very small quantity so that the following development into a geometric series

$$\frac{1}{1 - \frac{1}{r}\varphi_r'(x_k)} = 1 + \frac{1}{r}\varphi_r'(x_k) + O_M(\varepsilon_k^{2r-2}) \tag{8}$$

holds. Then, using (7) and (8) and the fact that $x_k - \phi_r(x_k) = O_M(\varepsilon_k)$, we obtain

$$\begin{aligned}\varphi_{r+1}(x_k) &:= x_k - \frac{x_k - \varphi_r(x_k)}{1 - \frac{1}{r}\varphi_r'(x_k)} = \varphi_r(x_k) - \frac{1}{r}\varphi_r'(x_k)(x_k - \varphi_r(x_k)) + O_M(\varepsilon_k^{2r-1}) \\ &= \alpha + O_M\left(\max_{r\geq 2}\{|\varepsilon_k|^{r+1}, |\varepsilon_k|^{2r-1}\}\right) = \alpha + O_M(\varepsilon_k^{r+1}),\end{aligned}$$

that is,

$$\varphi_{r+1}(x_k) - \alpha = O_M(\varepsilon_k^{r+1}) \quad (r \geq 2),$$

meaning that order of convergence of the iterative method (2) is $r + 1$. $\square$

Remark 1. The iterative formula (2) can be derived in several manners. The following one is quite simple: omitting aside $O_M(\varepsilon_k^{r+1})$ and replacing the root α by a new approximation x_{k+1} in (7), we get

$$x_{k+1} = \varphi_r(x_k) - \frac{1}{r}\varphi_r'(x_k)(x_k - \varphi_r(x_k)), \tag{9}$$

assuming that x_{k+1} is a better approximation to α than x_k. Substituting $\varphi_r(x_k) = x_{k+1}$ in the second term of (9) and solving the equation

$$x_{k+1} = \varphi_r(x_k) - \frac{1}{r}\varphi_r'(x_k)(x_k - x_{k+1})$$

in x_{k+1}, we obtain the iterative formula (2).

Remark 2. The ability of $\mathcal{AG}$ (2) to produce root-finding methods of an arbitrary order of convergence (see Theorem 2) is the main advantage of the generating formula (2). Furthermore, $\mathcal{AG}$ (2) can generate iterative formulas both for simple and multiple roots without alternation to its structure; it is sufficient to start with a suitably chosen initial iterative function $\varphi_r(x)$ $(r \geq 2)$.

Note that some relations utilized in the proof of Theorem 2 have appeared in the literature dealing with normed vector spaces (for example, see [2]). In this paper we deal with the space of complex numbers $\mathbb{C}$ and demonstrate how $\mathcal{AG}$ (2) can be successfully applied to generate accelerated root-finding methods. For example, proceeding from the Newton method

$$\varphi_2(x) = N_f(x) = x - \frac{f(x)}{f'(x)} = x - u(x) \tag{10}$$

and applying (2), we obtain Halley's third-order method

$$\varphi_3(x) = H_f(x) = x - \frac{f(x)}{f'(x) - \frac{f(x)f''(x)}{2f'(x)}},$$

assuming that methods N_f and H_f are applied to simple roots. At present, however, we shall restrict our consideration to accelerated methods for approximating multiple roots since this problem is rarely studied in the literature.

3. Applications to multiple roots

Following the derivation of (2) we observe that no assumptions on the order of the multiplicity of the root of f were imposed. Therefore, we can apply $\mathcal{AG}$ (2) to generate iterative methods of an arbitrary order of convergence for finding not only simple roots, but also multiple roots of nonlinear functions without any modification. This is a significant advantage of $\mathcal{AG}$ (2) in reference to existing generating formulas.

We will consider two classes of methods for finding multiple roots, categorized thus: (I)—methods where multiplicity is known, (II)—methods where multiplicity is unknown. In addition, we will also demonstrate the acceleration of convergence of simultaneous methods for finding all multiple roots of a polynomial with known multiplicities (Section 4). We wrote a simple program in the programming package *Mathematica* that generates iterative functions for an arbitrary r, but we omit cumbersome iterative methods of very high order to save space.

(I) *Multiplicity is known*

Let m be the order of multiplicity of the desired root α of a given function f, and let us introduce the notation

$$u = u(x) = \frac{f(x)}{f'(x)}, \qquad C_r = C_r(x) = \frac{f^{(r)}(x)}{r!f'(x)} \quad (r = 1, 2, \ldots).$$

Starting from a modified Newton method (derived by Schröder [3, p. 324] in 1870) of the second order

$$\varphi_2(x) = x - m\frac{f(x)}{f'(x)} = x - mu(x), \tag{11}$$

by $\mathcal{AG}$ (2) we obtain the third-order Halley-like method for multiple roots

$$\varphi_3(x) = x - \frac{x - \varphi_2(x)}{1 - \frac{1}{2}\varphi_2'(x)} = x - \frac{mu(x)}{\frac{1}{2}(1+m) - mC_2(x)u(x)}. \tag{12}$$

Continuing this process of the acceleration of convergence, we find by (2) (omitting argument x for brevity)

$$\varphi_4(x) = x - \frac{mu\left[\frac{1}{2}(1+m) - mC_2u\right]}{\frac{1}{3!}(1+m)(1+2m) - m(1+m)C_2u + m^2C_3u^2}$$

$$\varphi_5(x) = x - \frac{mu[(1+m)(1+2m)/3! - m(1+m)C_2u + m^2C_3u^2]}{\frac{1}{4!}(1+m)(1+2m)(1+3m) - \frac{1}{2!}(1+m)(1+2m)C_2u + (1+m)\left[C_3 + \frac{1}{2}C_2^2\right]m^2u^2 - m^3C_4u^3},$$

etc.

Remark 3. The above formulas φ_4 and φ_5 may be regarded as rediscovered formulas since they were derived originally in [4] using a different approach.

Remark 4. It was proved in [5] that, in the case of simple roots, $\mathcal{AG}$ $(2)_{m=1}$ is equivalent to Schröder's method of the second kind [3] (see [6] for a translation of this fundamental work of Schröder), sometimes called Schröder–König's method (see, e.g., [7,8])

$$S_r(x) = x - u(x)\frac{Q_{r-2}(x)}{Q_{r-1}(x)}, \quad (r \ge 2), \tag{13}$$

where Q_k is calculated by the recurrence relation

$$Q_0(x) = 1, \qquad Q_k(x) = \sum_{\lambda=1}^{k}(-1)^{\lambda+1}u(x)^{\lambda-1}C_\lambda(x)Q_{k-\lambda}(x), \qquad C_1(x) = 1, \quad (k \ge 1).$$

A modification of Schröder–König's method (13) for multiple roots is given in [1, Lemma 7.1] by $\mathcal{AG}$

$$\mathcal{F}_{r+1}(x,m) = \mathcal{F}_r(x,m) - \frac{mu(x)}{r}\mathcal{F}'_r(x,m), \qquad \mathcal{F}_2(x,m) = x - mu(x), \quad r \ge 2. \tag{14}$$

For $r = 2$, generating formula (14) produces a Chebyshev-like method of the third order

$$\widetilde{\varphi}_3(x) = x - mu(x)[(3-m)/2 + mC_2(x)u(x)], \tag{15}$$

and so on. For $m = 1$ (being the case of a simple root) $\mathcal{AG}$ (14) generates the same iterative methods as $\mathcal{AG}$ (13).

Remark 5. Regarding the iterative formula (9) it seems natural that it would also generate higher-order methods. This holds for the first generated method of the third order (of the Chebyshev type) $\widetilde{\varphi}_3$ given by (15) (taking $\varphi_2 = x - mu(x)$ in (9)). In the continuation, the generating formula (9) produces iterative methods of order $r+1$; however, due to superfluous "parasite" terms they are quite cumbersome. Assuming that both formulas (9) and (14) start with $\varphi_2 = x - mu(x)$, it is preferable to use Traub's generating relation (14) than (9). This subject was discussed in [9].

$\mathcal{AG}$ (2) and some existing third-order methods for multiple roots can be combined to produce new methods. We provide three examples. First, using the Chebyshev-like method (15) of the third order in $\mathcal{AG}$ (2), we obtain the fourth-order method

$$\varphi_4^*(x) = x - \frac{3mu(x)[3 - m + 2mC_2(x)u(x)]}{4 + 3m - m^2 + 6m(m-1)C_2(x)u(x) + 6m^2[C_3(x) - 2C_2(x)^2]u(x)^2}. \tag{16}$$

Another sequence of iterative methods for finding multiple roots can be derived starting with Osada's third-order method [10]

$$\eta_3(x) = x - \frac{1}{2}m(m+1)\frac{f(x)}{f'(x)} + \frac{1}{2}(m-1)^2\frac{f'(x)}{f''(x)}. \tag{17}$$

Using $\mathcal{AG}$ (2) we derive the fourth-order method

$$\eta_4(x) = x - \frac{3C_2(x)[(m-1)^2 - 2m(m+1)u(x)C_2(x)]}{4m(m+1)u(x)C_2(x)^3 - 6(m+1)C_2(x)^2 - 3(m-1)^2C_3(x)}. \tag{18}$$

It is assumed that $m \ne 1$ in (17), otherwise (17) reduces to the Newton method (10).

By combining Ostrowski's third-order method

$$\omega_3(x) = x - \frac{\sqrt{m}u(x)}{\sqrt{1 - 2u(x)C_2(x)}} \tag{19}$$

and $\mathcal{AG}$ (2), we generate the fourth-order method

$$\omega_4(x) = x - \frac{3\sqrt{m}u(x)[1 - 2u(x)C_2(x)]}{2[1 - 2u(x)C_2(x)]^{3/2} + \sqrt{m}[1 - 3u(x)C_2(x)] + 3\sqrt{m}u(x)^2C_3(x)}. \tag{20}$$

We have found no previous derivation of the iterative methods (16), (18) and (20).

Using recently derived third-order methods for multiple roots (see, e.g., the papers [10–14] and the references cited therein), one may continue further construction of new higher-order methods.

(II) *Multiplicity is unknown*

We shall now consider the application of $\mathcal{AG}$ (2) to the case when the order of multiplicity is not known. Let α be a multiple root of a function $f(x)$, then α is a simple root of the function $f(x)/f'(x)$. Applying the Newton method (10) to the function $u(x) = f(x)/f'(x)$, we obtain the iterative method

$$\psi_2(x) = x - \frac{f(x)f'(x)}{f'(x)^2 - f(x)f''(x)} = x - \frac{u(x)}{1 - 2C_2(x)u(x)}, \tag{21}$$

which converges quadratically. Taking $\varphi_2(x) = \psi_2(x)$ in (2) we obtain the accelerated method

$$\psi_3(x) = x - \frac{u(x)[1 - 2C_2(x)u(x)]}{1 - 3C_2(x)u(x) + 3C_3(x)u(x)^2} \tag{22}$$

with a cubic convergence. The next method of the fourth order is obtained using ψ_3 in (2),

$$\psi_4(x) = x - \frac{u(x)[1 - 3C_2(x)u(x) + 3C_3(x)u(x)^2]}{1 - 4C_2(x)u(x) + 4[2C_3(x) + C_2(x)^2]u(x)^2 - 4C_4(x)u(x)^3}. \tag{23}$$

While we can continue to generate higher-order methods using ψ_4 in (2), and so on, these iterative formulas are rather cumbersome.

Note that the initial iterative function

$$\psi_2(x) = x - \frac{f(x)f'(x)}{f'(x)^2 - f(x)f''(x)}$$

(given by (21)) allows us to rewrite $\mathcal{AG}$ (2) in the form

$$\psi_r(x) = x + (r-1)f(x)\frac{T_{r-2}(x)}{T_{r-1}(x)} \quad (r \geq 2),$$

where T_r is defined by the recurrence relation

$$T_{r-1}(x) = T'_{r-2}(x)f(x) - (r-1)T_{r-2}(x)f'(x), \qquad T_0(x) = f'(x).$$

Remark 6. Existing literature has attributed the modified Newton method (21) to various authors. However, Schröder was the first to derive this method in his paper [3] (see, also, Stewart's English translation [6]).

4. Simultaneous methods for multiple roots

Let P be a polynomial of degree n having real or complex roots $\alpha_1, \ldots, \alpha_\nu$ $(\nu \leq n)$ with the respective multiplicities $\mu_1, \ldots, \mu_\nu$, and let $x_1, \ldots, x_\nu$ be approximations to these roots. Now we will show that a modified $\mathcal{AG}$ (2) can be also applied to the acceleration of iterative methods for simultaneously determining all of the multiple roots of a polynomial. Let us first note that the accelerating generator (2) *cannot* be applied directly to simultaneous methods since Theorem 1 does not hold for simultaneous methods. In this section we provide a new and effective approach for accelerating simultaneous methods.

To demonstrate, we will consider the acceleration of a class of simultaneous methods derived by using the so-called Weierstrass function

$$W_i(x) = \frac{P(x)}{\prod\limits_{j \in \boldsymbol{I}_\nu \setminus \{i\}} (x - x_j)^{\mu_j}} \quad (i \in \boldsymbol{I}_\nu := \{1, \ldots, \nu\}), \tag{24}$$

(see [15]). Let us consider a class of iterative methods of order r of the form

$$\hat{x} = \psi_r(x; P/P', P''/P', \ldots, P^{(r-1)}/P') = \psi_r(x; u, C_2, \ldots, C_{r-1}) \tag{25}$$

for finding a single root of the polynomial P, which depends explicitly on the first $r-1$ derivatives of P. Recall that the order of convergence of such a one-point method cannot be greater than r; see Traub [1, Theorem 5.3]. Similarly to the coefficients C_r and the Newton correction $u(x) = P(x)/P'(x)$ used in Section 3, we define coefficients $C_{r,i} = W_i^{(r)}/(r!W_i')$ and Newton-like corrections $u_i = W_i/W_i'$, where

$$W_i = W_i(x_i), \qquad W_i^{(k)} = (W_i(x))^{(k)}_{x=x_i} \quad (k = 1, \ldots, r-1).$$

It is obvious that the Weierstrass function $W_i(x)$ and the polynomial $P(x)$ in (24) have the same roots. This means that the iterative formula (25) will produce the same approximations to the root α_i of $W_i(x)$, so that we can modify (25) to the form

$$\hat{x}_i = \psi_r(x_i; W_i/W_i', W_i''/W_i', \ldots, W_i^{(r-1)}/W_i') = \psi_r(x_i; u_i, C_{2,i}, \ldots, C_{r-1,i}). \tag{26}$$

We assume that approximations $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_\nu$ are temporarily fixed. Since the iterative formula (26) serves for finding a single zero, we can apply $\mathcal{AG}$ (2) to the iterative function (26) to obtain

$$\psi_{r+1}(x_i) = x_i - \frac{(x_i - \psi_r(x_i))}{1 - \frac{1}{r}\psi_r'(x_i)} \quad (i \in \boldsymbol{I}_\nu), \tag{27}$$

where we write $\psi_r(x_i)$ instead of $\psi_r(x_i; u_i, C_{2,i}, \dots, C_{r-1,i})$ for the sake of brevity.

The quantities $u_i, C_{2,i}, \dots, C_{r-1,i}$ in (26) are calculated in the following manner. Using the logarithmic derivative and the abbreviations

$$\delta_r(x) = \frac{P^{(r)}(x)}{P(x)}, \qquad \delta_{r,i} = \delta_r(x_i), \qquad S_{r,i}(x) = \sum_{j \in \boldsymbol{I}_\nu \setminus \{i\}} \frac{\mu_j}{(x - x_j)^r}, \qquad S_{r,i} = S_{r,i}(x_i),$$

$$y_{r,i} = \left[\frac{\mathrm{d}^r}{\mathrm{d}x^r}(\log W_i(x))\right]_{x=x_i} \qquad (r = 1, 2, \dots),$$

we find

$$y_{1,i} = \delta_{1,i} - S_{1,i}, \qquad y_{2,i} = \delta_{2,i} - \delta_{1,i}^2 + S_{2,i}, \qquad y_{3,i} = 2\delta_{1,i}^3 - 3\delta_{1,i}\delta_{2,i} + \delta_{3,i} - 2S_{3,i}, \text{ etc.}$$

A simple calculation yields

$$u_i = \frac{1}{y_{1,i}}, \qquad C_{2,i} = \frac{1}{2}\left(y_{1,i} + \frac{y_{2,i}}{y_{1,i}}\right), \qquad C_{3,i} = \frac{1}{6}\left(y_{1,i}^2 + 3y_{2,i} + \frac{y_{3,i}}{y_{1,i}}\right), \text{ etc.} \tag{28}$$

Now let us apply the iterative formula (26) for all $i = 1, \dots, \nu$ in parallel. Then all approximations involved in sums, appearing in $y_{r,i}$, are improved simultaneously, leading to the increase of the convergence order of the simultaneous method (26) by one related to the one-point method (25); see [15]. This means that simultaneous iterating transforms the iterative function $\psi_r(x_i)$ of order r into a new iterative function $\Psi_{r+1}(x_i)$ of order $r+1$, modifying (27) in this way to the accelerating generator for simultaneous methods

$$\Psi_{r+2}(x_i) = x_i - \frac{(x_i - \Psi_{r+1}(x_i))}{1 - \frac{1}{r}\Psi_{r+1}'(x_i)} \quad (i = 1, \dots, \nu). \tag{29}$$

The above accelerating technique will be applied to classes of simultaneous methods based on the Newton, Halley and Chebyshev methods for finding multiple zeros, given by the iterative formulas (11), (12) and (15), respectively.

Let $f \equiv P$ be a polynomial of degree n in the modified Newton method (11). Substituting $W_i(x)$ and $W_i'(x)$ instead of $P(x)$ and $P'(x)$ in (11), we obtain

$$\hat{x} = g_{2,i}(x) := x - \mu_i(\delta_1(x) - S_{1,i}(x))^{-1}. \tag{30}$$

Assuming that approximations to the roots $\alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_\nu$ are fixed and x is an approximation to a single root $\alpha = \alpha_i$, the method (30) is of the second order. However, a simultaneous approximation of *all* roots by the iterative method (30) transforms $g_{2,i}$ into the following simultaneous method

$$\hat{x}_i = G_{3,i}(x_i) = x_i - \mu_i(\delta_{1,i} - S_{1,i})^{-1} = x_i - \frac{\mu_i}{y_{1,i}} \quad (i = 1, \dots, \nu). \tag{31}$$

This is the well-known Ehrlich–Aberth method of the third-order [16,17], one of the most efficient and frequently used simultaneous methods. Here, $\hat{x}_1, \dots, \hat{x}_\nu$ represent new approximations to the roots $\alpha_1, \dots, \alpha_\nu$.

By differentiating (30) we find

$$g_{2,i}'(x) = 1 + \frac{\mu_i(\delta_2(x) - \delta_1(x)^2 + S_{2,i}(x))}{(\delta_1(x) - S_{1,i}(x))^2}.$$

Replacing $g_{2,i}$ and $g_{2,i}'$ in (27) yields

$$\begin{aligned} g_{3,i}(x) &= x - \frac{\mu_i(\delta_1(x) - S_{1,i}(x))^{-1}}{1 - \frac{1}{2}g_{2,i}'(x)} = x - \frac{2\mu_i(\delta_1(x) - S_{1,i}(x))^{-1}}{1 - \mu_i\frac{h_2(x) - \delta_1(x)^2 + S_{2,i}(x)}{(\delta_1(x) - S_{1,i}(x))^2}} \\ &= x - \frac{2\mu_i(\delta_1(x) - S_{1,i}(x))}{\left(\delta_1(x) - S_{1,i}(x)\right)^2 - \mu_i(\delta_2(x) - \delta_1(x)^2 + S_{2,i}(x))}. \end{aligned}$$

Putting $x = x_i$ in the last relation, we obtain the fourth-order method for the simultaneous determination of multiple roots of a polynomial,

$$\hat{x}_i = G_{4,i}(x_i) = x_i - \frac{2\mu_i(\delta_{1,i} - S_{1,i})}{(\delta_{1,i} - S_{1,i})^2 - \mu_i(\delta_{2,i} - \delta_{1,i}^2 + S_{2,i})} = x_i - \frac{2\mu_i y_{1,i}}{y_{1,i}^2 - \mu_i y_{2,i}} \quad (i \in \boldsymbol{I}_\nu), \tag{32}$$

assuming that all approximations $\hat{x}_1, \dots, \hat{x}_\nu$ are calculated in the parallel mode. Recall that we put $r = 2$ in (29) since the Ehrlich–Aberth method (31) is derived from the *second-order* Newton method (11).

The following new fifth-order simultaneous method arises from the method $G_{4,i}$ by using $\mathcal{AG}$ (29),

$$G_{5,i}(x_i) = x_i - \frac{3\mu_i(y_{1,i}^2 - \mu_i y_{2,i})}{y_{1,i}^3 - 3\mu_i y_{1,i} y_{2,i} + \mu_i^2 y_{3,i}} \quad (i \in \boldsymbol{I}_\nu). \tag{33}$$

We could continue in this vein with the accelerating procedure to construct a simultaneous method of higher order, but we shall conclude here to save space; the industrious reader may certainly proceed. Bearing in mind that the Ehrlich–Aberth method (31) is used as the base for generating methods (32) and (33), we shall refer to these methods as the Ehrlich–Aberth type.

Consider next another new sequence of simultaneous iterative methods originating from the Chebyshev method (15) using $\mathcal{AG}$ (29). Substituting $u(x)$ by $u_i = 1/y_{1,i}$ and $C_2(x)$ by $C_{2,i} = \frac{1}{2}(y_{1,i} + y_{2,i}/y_{1,i})$ in (15), we obtain the fourth-order method

$$H_{4,i}(x_i) = x_i - \frac{\mu_i}{2y_{1,i}^3}(3y_{1,i}^2 + \mu_i y_{2,i}) \quad (i \in \boldsymbol{I}_\nu). \tag{34}$$

In a similar manner to the above, using the iterative formula (34) and $\mathcal{AG}$ (29), we derive the fifth-order simultaneous method

$$H_{5,i}(x_i) = x_i - \frac{3\mu_i y_{1,i}(3y_{1,i}^2 + \mu_i y_{2,i})}{4y_{1,i}^4 - 3\mu_i y_{1,i}^2 y_{2,i} - 3\mu_i^2 y_{2,i}^2 + \mu_i^2 y_{1,i} y_{3,i}} \quad (i \in \boldsymbol{I}_\nu). \tag{35}$$

It is worth mentioning that $\mathcal{AG}$ (29) can also be applied for the acceleration of iterative interval methods for the simultaneous inclusion of polynomial multiple roots in complex interval arithmetic. Since this procedure is essentially identical to the case of simultaneous methods in ordinary complex arithmetic, we will not discuss interval methods in this paper.

5. Computational aspects

This section begins with a short discussion about the efficiency of considered methods. Knowledge of computational efficiency is of particular interest in designing a package of root-solvers. More details about this topic may be found in [18, Ch. 6]. The efficiency of an iterative method can be successfully estimated using the *efficiency index* given by

$$E = \frac{\log r}{\theta},$$

where r is the order of convergence of the method under study and θ is the computational "cost" per iteration; see [1] and [19, Ch. 1].

In the case of iterative methods for solving nonlinear equations of the form $f(x) = 0$, the computational cost θ is approximately equal to the number of new function evaluations of f and its derivatives per iteration. Since one-point methods for finding a single multiple root, considered in Section 3, require r function evaluations to reach the order r, their efficiency index is expressed by a simple formula

$$E(r) = \frac{\log r}{r}.$$

Since $E'(r) = (1 - \log r)/r^2$, the function $E(r)$ attains its maximum for $r = e \approx 2.718$. Therefore, it turns out that cubically convergent one-point iterative methods possess the highest computational efficiency.

Estimation of the computational efficiency of iterative methods for the simultaneous determination of multiple roots of polynomials is a more complex task. Beside the evaluation of a polynomial and its derivatives at ν $(\leq n)$ points (where ν is the number of distinct roots and n is the polynomial degree), evaluations of sums and products can appear. This means that we must count the total numbers of basic arithmetic operations and take into consideration the processor time needed for the execution of these operations.

An efficient approach that shows a good agreement with real CPU (central processor unit) time deals with certain operation *weights* depending on the processor execution time for each of the basic operations; see [20] for details. Let w_{as}, w_m and w_d denote these weights for addition + subtraction, multiplication and division, respectively. Furthermore, let AS_ν, M_ν and D_ν denote the number of additions + subtractions, multiplications and divisions per one iteration for all ν roots of a given polynomial. Then the computational cost $\theta(\nu)$ can be calculated approximately as $\theta = \theta(\nu) = w_{as}AS_\nu + w_m M_\nu + w_d D_\nu$. Hence, the efficiency index is given by

$$E_\nu = \frac{\log r}{\theta(\nu)} = \frac{\log r}{w_{as}AS_\nu + w_m M_\nu + w_d D_\nu}. \tag{36}$$

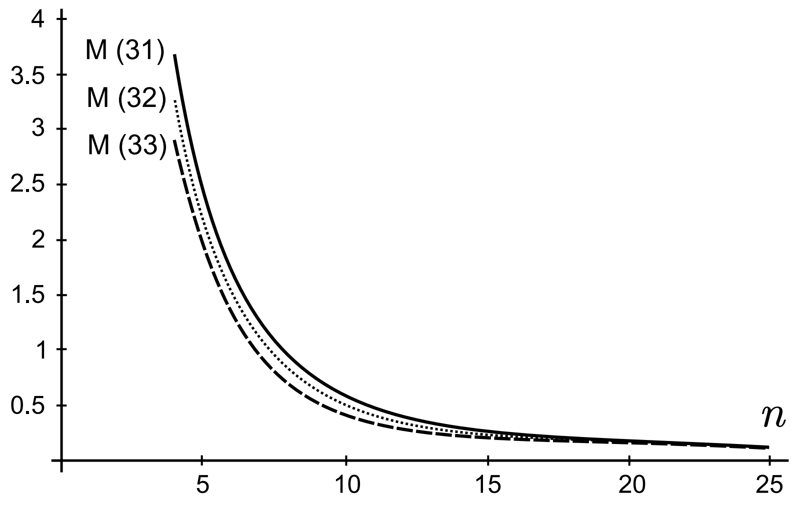


Fig. 1. Computational efficiency of the simultaneous methods (31)–(33).

Table 1
Errors of approximations.

Methods	$\|x^{(1)} - \alpha\|$	$\|x^{(2)} - \alpha\|$	$\|x^{(3)} - \alpha\|$	$\|x^{(4)} - \alpha\|$
The Newton method N_f(10)	1.32	1.09	0.90	0.75
The Halley-like method φ_3(12)	5.77(−2)	2.29(−6)	1.42(−19)	3.44(−59)
The second-order method ψ_2(21)	1.93(−1)	3.44(−4)	1.94(−12)	3.49(−37)
The third-order method ψ_3(22)	1.12(−1)	3.69(−5)	1.20(−15)	4.11(−47)
The fourth-order method φ_4^*(16)	2.28(−2)	8.53(−12)	6.24(−59)	1.30(−294)
Osada's method (17)	8.34(−2)	6.91(−6)	3.94(−18)	7.26(−55)
Accelerated Osada's method (18)	3.15(−2)	5.35(−11)	7.51(−55)	4.11(−274)
Ostrowski's method (19)	5.05(−2)	1.53(−6)	4.27(−20)	9.26(−61)
Accelerated Ostrowski's method (20)	1.23(−3)	2.94(−19)	2.29(−97)	2.48(−421)

As presented in [20] (see, also, [18, Ch. 6]), the ranking list of methods created according to (36) closely matches the ranking list made on the basis of real CPU time.

The estimation of the computational efficiency of simultaneous root-methods is more complicated in the case of multiple roots since, in addition to the polynomial degree n, the number of distinct roots ν must be taken into account as an important new parameter. Thus, we will simplify our analysis of efficiency and restrict our attention to simultaneous methods for finding only simple roots, realized in real arithmetic. To demonstrate, we will consider the iterative methods (31)–(33), generated by $\mathcal{AG}$ (29) starting from the Ehrlich–Aberth method (31).

It is commonly accepted that the weights appearing in (36) are proportional to the number of cycles in the execution of the four basic operations. To compare the simultaneous methods (31)–(33), we used the numbers of cycles (necessary for the evaluation of weights appearing in (36)) given in [21]. The numbers of basic operations for these three methods are $AS_n = rn^2 + O(n)$, $M_n = (2r-4)n^2 + O(n)$, and $D_n = n^2 + O(n)$, where $r = 3, 4, 5$ denotes the order of convergence. The (scaled) dependence of efficiency of the methods (31)–(33) on the polynomial degree n is graphically displayed in Fig. 1. We observe that method (31) is the most efficient and that method (32) is more efficient than (33). The same order remains unchanged for other weights. However, the differences among the efficiency indices are hardly noticeable, especially for polynomials of a high degree; see Fig. 1. To be more precise, we calculated relative efficiency indices $[(E_i - E_j)/E_j] \cdot 100$ (in %) and found that the Ehrlich–Aberth method (31) is about 10% more efficient than method (32) and about 23% more efficient than method (33).

We tested the considered methods in examples of functions having multiple roots. To illustrate, we have selected nine methods to solve two nonlinear equations and three simultaneous methods of the Ehrlich–Aberth type for finding polynomial roots.

Example 1. We considered the function

$$f(x) = x \sin x - 2(\sin(x/\sqrt{2}))^2$$

which has the root $\alpha = 0$ of the sixth order (not entirely obvious!). Starting from the initial approximation $x_0 = 1.6$, we carried out four iterations and obtained the following results given in Table 1. Instead of $A \times 10^{-h}$, a shorter notation $A(-h)$ is used in Tables 1–3.

Table 2
Errors of approximations.

Methods	$\|x^{(1)}-\alpha\|$	$\|x^{(2)}-\alpha\|$	$\|x^{(3)}-\alpha\|$	$\|x^{(4)}-\alpha\|$
The Newton method N_f(10)	0.21	0.13	0.0787	0.0502
The Halley-like method φ_3(12)	2.86(−3)	2.85(−9)	2.72(−27)	2.36(−81)
The second-order method ψ_2(21)	8.52(−2)	6.54(−3)	4.47(−5)	2.11(−9)
The third-order method ψ_3(22)	8.25(−3)	5.04(−7)	1.12(−19)	1.25(−57)
The fourth-order method φ_4^*(16)	1.61(−2)	1.31(−8)	7.53(−33)	8.26(−130)
Osada's method (17)	0.196	1.90(−2)	1.63(−5)	1.03(−14)
Accelerated Osada's method (18)	0.218	2.59(−3)	3.55(−11)	1.23(−42)
Ostrowski's method (19)	1.96(−2)	3.21(−6)	1.45(−17)	1.35(−51)
Accelerated Ostrowski's method (20)	9.89(−3)	2.39(−9)	7.70(−36)	8.27(−142)

Table 3
Norms of the errors of approximations.

Methods	$\epsilon^{(1)}$	$\epsilon^{(2)}$	$\epsilon^{(3)}$	$\epsilon^{(4)}$
The Ehrlich–Aberth method (31)	7.06(−2)	7.14(−5)	5.86(−14)	3.25(−41)
The Ehrlich–Aberth-like method (32)	1.88(−2)	6.07(−9)	1.60(−35)	4.08(−145)
The Ehrlich–Aberth-like method (33)	5.21(−3)	3.30(−13)	8.12(−67)	3.91(−347)

Example 2. We applied the same methods as in Example 1 to find improved approximations to the root $\alpha = 1$ of the multiplicity $m = 3$ of the polynomial

$$f(x) = (x-1)^3(x^2+2x+5)^2(x+5)(x^5+x+1).$$

We chose $x_0 = 0.6$ as the initial approximation. Results of the four iterations are given in Table 2.

From Tables 1 and 2 we observe that the Newton method (10) converges very slowly, in accordance with the fact that this method converges linearly in the case of multiple roots. The convergence behavior of the methods ψ_2 and ψ_3, which do not require any information on the order of multiplicity, is satisfactory. The convergence rate of methods with known multiplicities matches with the theoretical result very well. Analyzing the results of Example 2, we note that Osada's method (17) and its modification (18) converge slowly at the beginning of the iterative process. However, these methods reach the expected convergence orders 3 and 4 in later iterations.

Example 3. We applied the simultaneous methods (31)–(33) of the Ehrlich–Aberth type for finding all multiple roots $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_6)$ of the polynomial

$$P(x) = (x+3)(x-2\mathrm{i})^3(x^2+4x+5)^2(x^2-4x+5)^2.$$

The roots of this polynomial are $\alpha_1 = -3$, $\alpha_2 = 2\mathrm{i}$, $\alpha_{3,4} = -2 \pm \mathrm{i}$, $\alpha_{5,6} = 2 \pm \mathrm{i}$ with multiplicities $\mu_1 = 1$, $\mu_2 = 3$, $\mu_3 = \mu_4 = \mu_5 = \mu_6 = 2$. The following complex numbers were taken as initial approximations

$$x_1^{(0)} = -3.3 + 0.2\mathrm{i}, \qquad x_2^{(0)} = 0.3 + 2.3\mathrm{i}, \qquad x_3^{(0)} = -2.3 + 1.2\mathrm{i},$$
$$x_4^{(0)} = -2.3 - 1.2\mathrm{i}, \qquad x_5^{(0)} = 2.3 + 1.2\mathrm{i}, \qquad x_6^{(0)} = 2.3 - 1.2\mathrm{i}.$$

As a measure of accuracy of the approximations obtained, we calculated Euclid's norm

$$\epsilon^{(m)} := \|\boldsymbol{x}^{(m)} - \boldsymbol{\alpha}\|_2 = \left(\sum_{i=1}^{6} |z_i^{(m)} - \alpha_i|^2\right)^{1/2} \qquad \left(\boldsymbol{x}^{(m)} = \left(x_1^{(m)}, \ldots, x_6^{(m)}\right);\, m = 0, 1, \ldots\right).$$

The errors $\epsilon^{(k)}$ are given in Table 3. All tested methods converge rapidly with the convergence rate which coincides very well with the theoretical results.

Finally, a few words about iterative methods with a known multiplicity. Most of the papers treating such methods begin with the phrase "Let α be a root of f with the given multiplicity $m, \ldots$," with no information how to provide the exact m. Comparing iterative methods of the same convergence speed that deal (i) with a known multiplicity (e.g., (11) and (12)) and (ii) with no information on multiplicity (such as (21) and (22)), we observe that the latter ones require one more function evaluation. This is the price to pay to spare one from long calculations. Thus, we avoid the strenuous procedure of finding the order of multiplicity. Moreover, most algorithms to determine the order of multiplicity may lead to mutually opposite requirements. We shall illustrate this fact in the following two multiplicity-finding methods.

(I) Traub [1, p. 154] showed that

$$m \approx \frac{\log|f(x)|}{\log|u(x)|} \tag{37}$$

when x is very close to a root of f.

Table 4
Approximate multiplicity.

Approximations $x \longrightarrow$	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Traub's formula (37)	3.636	3.773	3.913	4.059	4.213	4.381	4.572	4.912
Lagouanelle's formula (38)	5.728	5.791	5.846	5.893	5.931	5.961	5.983	5.996

(II) Lagouanelle [22] derived the following approximate formula

$$m \approx \frac{f'(x)^2}{f'(x)^2 - f(x)f''(x)}, \tag{38}$$

assuming again that x is very close to a root of f.

Both the methods demand a very close approximation to calculate a multiplicity of high accuracy. On the other hand, to find a very close approximation to a multiple root, it is necessary to use precise multiplicity. However, both the requirements cannot be attained at the same time. Taking into account the opposite demands mentioned and additional calculations to find multiplicity, in those cases where we have not provided an accurate multiplicity (or we are not convinced that the calculated multiplicity is a genuine one; see the example below), it is sometimes better to apply a method which does not explicitly require the order of multiplicity (such as (21) or (22)), in spite of its lower computational efficiency arising from an additional function evaluation per iteration.

To emphasize the considered dilemma, we present a numerical example concerning the practical calculation of the order of multiplicity m by Traub's formula (37) and Lagouanelle's formula (38). We considered the function $f(x) = x\sin x - 2(\sin(x/\sqrt{2}))^2$ from Example 1 having the root $\alpha = 0$ of the sixth order. Approximate values of its multiplicity are given in Table 4 for different approximations x to the root. In practice, the calculated values are rounded to the nearest integer. However, in the case of Traub's formula rounding to 4 or 5 would yield incorrect results.

Finally, in addressing the accurate calculation of the order of multiplicity, we note that Johnson and Tucker [23] recently proposed an efficient quadrature approach to find the number of roots inside a given rectangle and to calculate their multiplicities. Their method is based on the argument principle and supported by the use of validated integration of contour integrals.

References

[1] J.F. Traub, Iterative Methods for the Solution of Equations, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.
[2] B. Jovanović, A method for obtaining iterative formulas of higher order, Mat. Vesnik 9 (24) (1972) 365–369.
[3] E. Schröder, Über unendlich viele algorithmen zur Auflösung der Gleichungen, Math. Ann. 2 (1870) 317–365.
[4] M.R. Farmer, G. Loizou, An algorithm for the total, or partial, factorization of a polynomial, Math. Proc. Cambridge Philos. Soc. 82 (1977) 427–437.
[5] M.S. Petković, D. Herceg, On rediscovered iteration methods for solving equations, J. Comput. Appl. Math. 107 (1999) 275–284.
[6] G.W. Stewart, On infinitely many algorithms for solving equations. Available by the corresponding author, ftp://thales.cs.umd.edu/pub/reports/imase.ps.
[7] X. Buff, C. Hendriksen, On König's root-finding algorithms, Nonlinearity 16 (2003) 989–1015.
[8] E.R. Vrscay, W.J. Gilbert, Extraneous fixed points, basin boundaries and chaotic dynamics for Schröder and König rational iteration functions, Numer. Math. 52 (1998) 1–16.
[9] M.S. Petković, L.D. Petković, Đ. Herceg, On Schröder's families of root-finding methods, J. Comput. Appl. Math. 233 (2010) 1755–1762.
[10] N. Osada, An optimal multiple root-finding method of order three, J. Comput. Appl. Math. 51 (1994) 131–133.
[11] C. Chun, H.J. Bae, Neta, New families of nonlinear third-order solvers for finding multiple roots, Comput. Math. Appl. 57 (2009) 1574–1582.
[12] C. Chun, B. Neta, A third-order modification of Newton's method for multiple roots, Appl. Math. Comput. 211 (2009) 474–479.
[13] M. Frontini, E. Sormani, Modified Newton's method with third order of convergence and multiple roots, J. Comput. Appl. Math. 156 (2003) 345–354.
[14] B. Neta, New third order nonlinear solvers for multiple roots, Appl. Math. Comput. 202 (2008) 162–170.
[15] M.S. Petković, L.D. Petković, Construction of zero-finding methods by Weierstrass functions, Appl. Math. Comput. 184 (2007) 351–359.
[16] O. Aberth, Iteration methods for finding all zeros of a polynomial simultaneously, Math. Comp. 27 (1973) 339–344.
[17] L.W. Ehrlich, A modified Newton method for polynomials, Commun. ACM 10 (1967) 107–108.
[18] M.S. Petković, Iterative Methods for Simultaneous Inclusion of Polynomial Zeros, Springer-Verlag, Berlin, Heidelberg, New York, 1989.
[19] J.M. McNamee, Numerical Methods for Roots of Polynomials, Part I, Elsevier, Amsterdam, 2007.
[20] G.V. Milovanović, M.S. Petković, On computational efficiency of the iterative methods for the simultaneous approximation of polynomial zeros, ACM Trans. Math. Software 12 (1986) 295–306.
[21] J. Fujimoto, T. Ishikawa, D. Perret-Gallix, High precision numerical computations, Technical report, ACCP-N-1, May 2005.
[22] J.L. Lagouanelle, Sur une métode de calcul de l'ordre de multiplicité des zéros d'un polynôme, C. R. Acad. Sci. Paris Sér. A 262 (1966) 626–627.
[23] T. Johnson, W. Tucker, Enclosing all zeros of an analytic function—A rigorous approach, J. Comput. Appl. Math. 228 (2009) 418–423.