Inverse Power and Durand-Kerner Iterations for Univariate Polynomial Root-Finding

D. A. Bini *
Dipartimento di Matematica, Università di Pisa
Via Buonarroti 2, 56127 Pisa, Italy
bini@dm.unipi.it

L.Gemignani †
Dipartimento di Matematica, Università di Pisa
Via Buonarroti 2, 56127 Pisa, Italy
gemignan@dm.unipi.it

V. Y. Pan ‡
Department of Mathematics and Computer Science
Lehman College of CUNY, Bronx, NY 10468, USA
vpan@lehman.cuny.edu

October 21, 2002

Abstract

Univariate polynomial root-finding is an oldest classical problem of mathematics and computational mathematics, which is still an important research topic, due to its impact on computational algebra and geometry. The Weierstrass (Durand–Kerner) approach and its variations as well as matrix methods based on the QR algorithm are among the most popular practical choices for simultaneous approximation of all roots of a polynomial. We propose an alternative application of the inverse power iteration to the Frobenius or generalized companion matrix for polynomial root-finding, demonstrate its effectiveness, and relate its study to unifying the derivation of the Weierstrass (Durand–Kerner) algorithm (having quadratic convergence) and its extensions having convergence rates $4, 6, 8, \ldots$.

*Supported by GNCS and by MIUR
†Supported by GNCS and by MIUR
‡Supported by NSF Grant CCR 9732206 and PSC CUNY Awards 66383-0032 and 64406-0033

Key words: polynomial root-finding, Weierstrass (Durand–Kerner) algorithm, higher order root-finders, companion matrices, matrix methods for root-finding, inverse power iteration.

2000 Math. Subject Classification: 65H05, 30C15, 65B99, 65E05

1 Introduction

Univariate polynomial root-finding is the four millenia old problem of mathematics and computational mathematics whose study throughout many centuries has made greatest impact on these fields (see [McN93], [McN97], [P97] for the bibliography and a survey). The problem still remains an important research topic, particularly due to its applications to algebraic and geometric computations. From the asymptotic computational complexity point of view, the problem was solved in [P95],[P01],[P02], where optimal (up to polylog factors) computational time bounds were reached under both arithmetic and Boolean computational models. The algorithms in [P95],[P01],[P02], however, are quite involved, which complicates their numerical analysis and practical implementation. The users presently prefer some easily implementable iterative algorithms. As a rule, rapid convergence of these algorithms is proved only locally, that is, where the roots are isolated from each other and are already approximated closely, but under the customary choices of initial approximations, rapid global convergence has been confirmed by statistics of extensive application of the algorithms in computational practice.

For simultaneous approximation of all n roots of a given polynomial, most popular practical choices are the Weierstrass (Durand–Kerner) iteration (hereafter we refer to it as the *W(D–K) iteration*) and its variations such as Aberth's (Ehrlich / Börsch–Supan's), Farmer–Loizou's and Werner's (see [PHI98] on derivation of such algorithms). Their convergence rate is quite high (it varies from 2 to 4), but the matrix methods based on the QR algorithms are still considered competitive. The latter methods (see [F90], [TT94], [EM95], [MV95a], [MV95b], [F01], and the bibliography therein) use the order of n^3 flops for all zeros (versus $O(n^2)$ per iteration in the W(D–K) iteration and its modifications) performed numerically with single precision.

In the present paper, our first innovation (in Section 2) is a simple unified derivations of the W(D–K) iteration and its extensions having convergence rates $4, 6, 8, \ldots$. Then in Sections 4 and 5, we revisit the inverse power iteration and apply it to the Frobenius or generalized companion matrices associated with the input polynomial. The latter quite natural but yet unexplored approach to approximating a single zero uses $O(n)$ flops per iteration step and exhibits fast convergence in our preliminary tests. We supply some details of application of the method to the Frobenius and generalized companion matrices; in particular, we show explicit expressions for the eigenvectors of a generalized companion matrix in terms of its eigenvalues and their approximations. The derivation of

the expressions turns out to be closely related to our derivation of the W(D–K) iteration and its cited extensions to iterations with higher convergence rates.

Practical polynomial root-finding and consequently practical application of the proposed algorithms should involve numerous important implementation issues, tricks and techniques (see, e.g., [BF00]), which are beyond the scope of this paper. We just briefly recall some effective policies for choosing initial approximations to the roots in Section 3.

Our analysis of the inverse power iteration can be easily extended to the eigencomputation for an important class of semisimple + diagonal matrices. In the case of the Frobenius and generalized companion matrices, the extension from approximating a single (absolutely smallest or largest) eigenvalue to all eigenvalues is simplified.

Acknowledgments: V. Pan thank S. Fortune, B. Mourrain, and E.E. Tyrtyshnikov for preliminary discussions on matrix methods for polynomial root-finding and X. Wang for providing help to simplify the original proof of equation (2.6).

2 The Weierstrass (Durand–Kerner) iteration and its higher order extensions

Given n distinct values $s_1, \ldots, s_n$ approximating the unknown distinct roots (zeros) $z_1, \ldots, z_n$ of a polynomial

$$p(x) = \prod_{i=1}^{n} (x - z_i), \tag{2.1}$$

the W(D–K) iteration consists in recursive computation of improved approximations

$$t_i = s_i - d_i, \quad i = 1, \ldots, n, \tag{2.2}$$

where

$$q(x) = \prod_{i=1}^{n} (x - s_i), \quad q_i(s_i) = q'(s_i), \quad i = 1, \ldots, n, \tag{2.3}$$

$$d_i = p(s_i)/q_i(s_i), \quad q_i(x) = q(x)/(x - s_i) = \prod_{j \neq i} (x - s_j). \tag{2.4}$$

The W(D–K) iteration has a local quadratic convergence and only requires computation of the values $p(s_i)$ and $q'(s_i)$, $i = 1, \ldots, n$. Hereafter we refer to d_i's as to the D–K corrections.

The iteration has been derived by applying Newton's method to the Viéte system of polynomial equations, relating the coefficients of $p(x)$ to the symmetric functions in its zeros $z_1, \ldots, z_n$. Moreover, the iteration can be written as

$$z_i^{new} = z_i^{old} - p(z_i^{old})/q'(z_i^{old}), i = 1, \ldots, n,$$

which is Newton's iteration

$$z_i^{new} = z_i^{old} - p(z_i^{old})/p'(z_i^{old}), i = 1, \ldots, n,$$

with $p'(x)$ replaced by its approximation $q'(x)$. Our next alternative derivation via the Lagrange interpolation formula,

$$p(x) = q(x) + \sum_{i=1}^{n} d_i q_i(x), \tag{2.5}$$

produces also iterations with the convergence rates $4, 6, 8, \ldots$.

Theorem 2.1. *Let $z_i \neq s_j$ for all $j \neq i$. Then we have*

$$s_i - z_i = d_i/(1 + \sum_{j \neq i} d_j/(z_i - s_j)), i = 1, \ldots, n. \tag{2.6}$$

Proof. Substitute $x = z_i$ into (2.5) and obtain that $q(z_i) + \sum_{j=1}^{n} d_j q_j(z_i) = 0$. If $z_i = s_i$, then $d_i = 0$, and (2.6) trivially holds. Otherwise, divide by $q(z_i)$, substitute (2.3) and (2.4), and obtain that

$$1 + \sum_{j=1}^{n} d_j/(z_i - s_j) = 0, \quad d_i/(s_i - z_i) = 1 + \sum_{j \neq i} d_j/(z_i - s_j).$$

Multiply both sides by $(s_i - z_i)/(1 + \sum_{j \neq i} d_j/(z_i - s_j))$ to obtain (2.6). □

Let us write

$$\Delta = \max_i \{|d_i| + |z_i - s_i|\}. \tag{2.7}$$

We immediately deduce from (2.6) that

$$\begin{aligned} z_i &= t_i + O(\Delta^2), \\ t_i &= s_i - d_i, \end{aligned} \qquad i = 1, \ldots, n \tag{2.8}$$

which shows quadratic local convergence of the W(D–K) iteration (2.2).

Substitute (2.8) on the right hand side of (2.6) and obtain an iteration of the fourth order:

$$\begin{aligned} z_i &= t_i + O(\Delta^4), \\ t_i &= s_i - d_i/(1 + u_i), \qquad i = 1, \ldots, n \\ u_i &= \textstyle\sum_{j \neq i} d_j/(s_i - d_i - s_j), \end{aligned} \tag{2.9}$$

Similarly substitute (2.9) into (2.6) and obtain an iteration of the sixth order:

$$\begin{aligned} z_i &= t_i + O(\Delta^6), \\ t_i &= s_i - d_i/(1 + v_i), \qquad i = 1, \ldots, n. \\ v_i &= \textstyle\sum_{j \neq i} d_j/(s_i - d_i/(1 + u_i) - s_j), \end{aligned} \tag{2.10}$$

Method	DK2	DK4	DK6	DK8
weighted cost over $\mathbb{R}$	$4n^2$	$3.5n^2$	$3.8n^2$	$4.3n^2$
weighted cost over $\mathbb{C}$	$12n^2$	$13.5n^2$	$16.2n^2$	$19n^2$

Table 1: Figure 1. Weighted cost of the higher order extensions.

Continue this pattern, substitute (2.10) into (2.6), obtain an iteration of the 8th order:

$$\begin{aligned} z_i &= t_i + O(\Delta^8), \\ t_i &= s_i - d_i/(1+w_i), \qquad\qquad i = 1,\ldots,n, \\ w_i &= \textstyle\sum_{j\neq i} d_j/(s_i - d_i/(1+v_i) - s_j), \end{aligned} \tag{2.11}$$

and so on. In this process, each increase of the convergence rate by two requires additional computation of the values

$$y_i = 1 + \sum_{j\neq i} d_j/(h_i - s_j), i = 1,\ldots,n, \tag{2.12}$$

where h_i are readily computable.

Observe that the cost of the W(D-K) iteration is just $4n^2 + O(n)$ arithmetic operations (ops), whereas the cost of each single higher order extension is $3n^2$ ops. If we assume that all the arithmetic operations have the same cost, then the arithmetic cost of the k-th higher order extension is $c(k) = (4+3k)n^2$ ops, $k = 0, 1, \ldots$. We may evaluate the performance of each iteration in terms of the weighted cost $c(k)\log_2(2k+2)$, where $(2k+2)$ is the order of convergence. This expression represents the ratio between the cost needed by the method to reach a give error bound ϵ in the approximation, and the cost of an ideal quadratically convergent method having unit cost per iteration needed to reach the same error bound as $\epsilon \to 0$. Table 1 shows that if we assume that the arithmetic operations have the same cost, then acceleration of the 4-th order is the best choice. On the other hand if we work with complex numbers, each complex addition costs two real ops, each complex multiplication costs 6 ops and complex division is performed with 11 ops. Therefore the approximation of the complex roots of a polynomial with real coefficients costs $12n^2 + O(n)$ ops for each step of the W(D-K) iteration, and the additional cost of $15n^2$ ops for each single higher order extension. In this case we deduce that the most efficient method still remains the W(D-K) iteration.

3 The choice of initial approximations

It is well-known from extensive numerical tests that as a rule the W(D–K) algorithm as well as its various extensions converge rapidly starting with a quite random set of initial approximations $s_1, \ldots, s_n$. A customary choice is $s_i = a\omega^{i-1}$,

$i = 1, \ldots, n$, where $\omega = \exp(2\pi\sqrt{-1}/n)$ is a primitive nth root of 1, $a/\max_i |z_i|$ is set to, say 1.5 or 2, and $|z_i|$ are the unknown zeros of $p(x)$. C. Carstensen in [C91a] proposes to choose $s_1, \ldots, s_n$ by using Gershgorin's discs, D. Bini [B96] proposes to choose the initial approximations along different circles whose radii are computed by means of Rouché theorem and Newton's polygon.
S.Fortune [F01] applies the QR algorithm to the Frobenius matrix $F(\mathbf{p})$ and uses the computed approximations to the eigenvalues as the initial approximations s_i. In spite of the order of n^3 flops involved, this single precision computation is quite fast, according to S.Fortune. Finally, a slower but reliable customary option is the continuation (or homotopy) approach, where one starts with a polynomial $p_{\tau_0}(x) = q(x) = \prod_j (x - s_j(\tau_0))$ with some fixed zeros $s_1(\tau_0), \ldots, s_n(\tau_0)$ and then recursively computes the zeros $s_j(\tau_i)$ for a sequence of polynomials $p_{\tau_i}(x) = \tau_i p(x) + (1 - \tau_i)q(x)$, $i = 0, 1, \ldots, K$, $\tau_0 < \tau_1 < \ldots < \tau_K = 1$ using the values $s_j = s_j(\tau_i)$ as the initial approximations to $t_j = s_j(\tau_{i+1})$, $j = 1, \ldots, n$. We refer the reader to [BF00], [KS94], [PHI98], [HSS01], and [BPa] on these and some other choices.

4 Generalized companion matrices and their eigenvectors

Definition 4.1. An $n \times n$ matrix $C = C(p(x))$ is a *generalized companion matrix* for a polynomial $p(x)$ in (2.1) if $\{z_1, \ldots, z_n\}$ is the set of the eigenvalues of C.

Thus root-finding for $p(x)$ amounts to the eigenvalue problem for C, where matrix methods can be applied. Next, we recall two most important classes of companion matrices C and express their eigenvectors via $z_1, \ldots, z_n$. In the next section we examine application of the inverse power method to these matrices, which can be viewed as an alternative or as a complement to the algorithms of Section 2.

Example 4.2. The Frobenius (companion) matrix $C = F(p(x))$ is given by

$$F(p(x)) = \begin{pmatrix} 0 & 1 & & \\ & & \ddots & \\ & & & 1 \\ -p_0 & -p_1 & \cdots & -p_{n-1} \end{pmatrix}$$

provided that $p(x) = x^n + \sum_{i=0}^{n-1} p_i x^i$. It has the eigenpairs $(z_i, \mathbf{v}_i)$, where

$$\mathbf{v}_i = (z_i^j)_{j=0}^{n-1}, \quad i = 1, \ldots, n. \tag{4.1}$$

♠ Victor, in def 4.1 we define the generalized companion matrices, therefore I inserted the term "generalized" in the sentence below, and I have removed the "emphasization" from "generalized companiona matrix" in the next definition 4.3

Next we define another important class of generalized companion matrices.

Definition 4.3. (Cf. [E73],[C91],[C92].) For a polynomial $p(x)$ of (2.1) and n distinct values $s_1, \ldots, s_n$, define a rank-one matrix $E_{\mathbf{d}}$ with diagonal entries $d_1, \ldots, d_n$ of (2.4) and an associated $n \times n$ *generalized companion matrix*

$$C = C_{\mathbf{s},\mathbf{d}} = D_{\mathbf{s}} - E_{\mathbf{d}}. \tag{4.2}$$

where $D_{\mathbf{s}}$ is a diagonal matrix with diagonal entries $s_1, s_2, \ldots, s_n$.

Definition 4.3 leaves us with some freedom in choosing the matrix $E_{\mathbf{d}}$. In particular, Fiedler in [F90] proposes $E_{\mathbf{d}} = \sigma \mathbf{f}\mathbf{f}^T$, $\mathbf{f} = (f_i)_{i=1}^n$, $\sigma f_i^2 = d_i$, $i = 1, \ldots, n$, for a fixed scalar σ, whereas Elsner in [E73] proposes

$$E_{\mathbf{d}} = \mathbf{1}\mathbf{d}^T, \tag{4.3}$$

where $\mathbf{1} = (1, 1, \ldots, 1)^T$.

♠ Victor, do we really need to write the proof of theorem 4.4? It can be found in the literature.

Theorem 4.4. *(Cf. [C91].) For any pair of matrices C and $E_{\mathbf{d}}$ of Definition 4.3, we have*

$$\det(xI - C) = p(x). \tag{4.4}$$

Proof. Since every $k \times k$ submatrix of $E_{\mathbf{d}}$ is singular for $k > 1$, all terms in the expansion of $\det(xI - C) = \det(D_{x\mathbf{1}-\mathbf{s}} + E_{\mathbf{d}})$ vanish except for those including the products of at least $n-1$ diagonal entries of $D_{x\mathbf{1}-\mathbf{s}}$. That is, $\det(xI - C)$ is made up entirely of the terms of $\det(D_{x\mathbf{1}-\mathbf{s}}) = \prod_{i=1}^{n}(x - s_i) = q(x)$ and $d_i q(x)/(x - s_i) = d_i q_i(x)$, $i = 1, \ldots, n$. Now (4.4) follows from the Lagrange interpolation formula (2.5). □

Our next goal is the expressions for the eigenvectors of the matrix C in (4.2), (4.3) via the eigenvalues $z_1, \ldots, z_n$.

Theorem 4.5. *Let C be the matrix in (4.2), (4.3) where s_i are pairwise distinct, $\mathbf{d} = (d_i)$, $d_i = p(s_i)/\prod_{j \neq i}(s_i - s_j)$, $i = 1, \ldots, k \leq n$, and $p(x)$ is the polynomial (2.1) having pairwise distinct zeros $z_1, \ldots, z_n$. Then, if $s_i \neq z_j$, for $i, j = 1, \ldots, n$ for the right and left eigenvectors $\mathbf{u}_j$ and $\mathbf{v}_j$, respectively, of C we have*

$$\begin{array}{ll} C\mathbf{v}_j = z_j \mathbf{v}_j & \mathbf{v}_j = (1/(s_i - z_j)) \\ \mathbf{u}_j^T C = z_j \mathbf{u}_j^T & \mathbf{u}_j = (d_i/(s_i - z_j)) \end{array} \quad j = 1, \ldots, k. \tag{4.5}$$

Proof. From $C\mathbf{v}_j = z_j \mathbf{v}_j$ we obtain $\mathbf{v}_j = (D - z_j I)^{-1}\mathbf{1}\mathbf{d}^T\mathbf{v}_j$. Whence, up to the normalization factor $d^T v_j$ we have $\mathbf{v}_j = (D - z_j I)^{-1}\mathbf{1} = (1/(s_i - z_j))$. Similarly, from $\mathbf{u}_j^T C = \mathbf{u}_j^T z_j$ we obtain $\mathbf{u}_j = (d_i/(s_i - z_j))$. □

Corollary 4.6. *Equation (2.6) holds.*

Proof. Equate the i-th coordinates on both sides of the vector equation $C\mathbf{v}_j = z_j\mathbf{v}_j$, substitute $\mathbf{v}_j$ from (4.5) and obtain (2.6). $\square$

We conclude this section with the following simple observation.

Theorem 4.7. *Given a scalar z such that $p(z) \neq 0$, a vector $\mathbf{x}$, and a Frobenius matrix $C = F(p(x))$ or a generalized companion matrix C of Definition 4.3, the vectors $C\mathbf{x}$ and $(C - zI)^{-1}\mathbf{x}$ can be computed by using $O(n)$ flops.*

Hereafter write $D = D_{\mathbf{s}}, E = E_{\mathbf{d}}$. Let us apply S-M-W formula [GL96] to invert the matrix $C = D_s - \mathbf{1}\mathbf{d}^T$ of (4.2) and (4.3), that is,

$$(C - zI)^{-1} = (I + \tfrac{1}{1-\tau}(D - zI)^{-1}\mathbf{1}\mathbf{d}^T)(D - zI)^{-1}$$

$$\tau = \mathbf{d}^T D^{-1}\mathbf{1}$$

Thus we have

$$(C - zI)^{-1}\mathbf{v} = (D - zI)^{-1}\mathbf{v} + \tfrac{\sigma}{1-\tau}(D - zI)^{-1}\mathbf{1},$$

$$\sigma = \mathbf{d}^T(D - zI)^{-1}\mathbf{1}, \quad \tau = \mathbf{d}^T(D - zI)^{-1}\mathbf{1}.$$

Therefore we can compute $\mathbf{y} = (C - zI)^{-1}\mathbf{x}$ according to the following steps in n reciprocations, $4n$ multiplications and $4n$ additions.

Algorithm 4.8. [Shifted inverse power iteration for a g.c.m.]

STEP 1: compute $\mathbf{g} = (D - zI)^{-1}\mathbf{1}$;

STEP 2: compute $\mathbf{u} = \mathbf{g} * \mathbf{x}$ where $*$ denotes componentwise product of vectors;

STEP 3: compute $\tau = \sum_{i=1}^{n} d_i g_i$;

STEP 4: compute $\sigma = \sum_{i=1}^{n} d_i g_i$;

STEP 5: compute $\mathbf{y} = \mathbf{u} + \frac{\sigma}{1-\tau}\mathbf{g}$.

For complex data the overall cost amounts to $37n + O(1)$ arithmetic operations between real numbers.

5 Application of the inverse power method to a (generalized) companion matrix

Under the assumptions of Theorem 4.5, let z be a sufficiently close approximation to an eigenvalue z_j of a matrix C and let $\mathbf{v} = \sum_{i=1}^{n} a_i\mathbf{v}_i$, $\|\mathbf{v}\|_2 = 1$, where $\mathbf{v}_j$, $j = 1, \ldots, n$ are the eigenvectors of C and $a_i \neq 0$. Then the shifted inverse power iteration is defined as follows:

$$\mathbf{x}^{(0)} = \mathbf{v}, \quad \mathbf{y}^{(k)} = (C - zI)^{-1}\mathbf{x}^{(k-1)}, \quad \mathbf{x}^{(k)} = \mathbf{y}^{(k)}/\|\mathbf{y}^{(k)}\|_2, \tag{5.1}$$

$$z^{(k)} = \mathbf{x}^{(k)T} C \mathbf{x}^{(k)}, \quad k = 1, 2, \ldots. \tag{5.2}$$

The pair $(\mathbf{y}^{(k)}, z^{(k)})$ rapidly converges to an eigenvector/eigenvalue pair $(\mathbf{v}_j, z_j)$ (see [GL96, Section 7.6]) provided that for all $i \neq j$ the ratios $|z - z_j| \big/ |z - z_i|$ are substantially less than 1 and the ratios $|a_j/a_i|$ are not close to 0. By Theorem 4.7, each iteration step uses $O(n)$ flops. Initial approximations z can be computed as in Section 3.

Practical statistics shows that random initial eigenvector is usually a good choice for $\mathbf{v}$ (see [GL96]). Having some initial information about the eigenvalues, we may further improve this choice based on (4.1) or (4.5). If s_i is close to z_i but not equal to z_i, then (4.5) suggests the choice of the i-th coordinate vector as $\mathbf{v}$. For $C = F(p(x))$, one may consider the choice of $\mathbf{v} = (\sum_{i=1}^{n} z_i^k)_{k=0}^{n-1}$, where the power sums of all the zeros z_i can be computed in $O(n \log n)$ flops. As soon as a zero z_i of $p(x)$ is closely approximated, one may reapply the algorithm to the deflated polynomial $p(x)/(x - x_i)$.

♠ Victor, is it really possible to extend theorem 4.5 to D+SS matrices? We do not see how it can be done.

Finally, Theorem 4.5 can be easily extended to the important class of semi-separable + diagonal matrices [EG97], [MCVH01]; consequently, the same inverse power algorithm can be extended to the eigencomputation for these matrices.

Remark 5.1. The convergence of the shifted inverse power algorithm can be accelerated if the shift value z is replaced at each step by the current approximation $z^{(k)}$ of the eigenvalue. In this way the local convergence to simple eigenvalues becomes superlinear.

Remark 5.2. In the shifted power method we may replace equation (5.2) with the following less expensive formula for the eigenvalues approximation:

$$z^{(k)} = (Cy^{(k)})_i / y_i^{(k)}$$

where the subscript i is such that $y_i \neq 0$. For the matrix $C = D - E$ of (4.2) and (4.3) the latter formula turns into

$$z^{(k)} = s_i - (\sum_{j=1}^{n} d_j y_j^{(k)}) / y_i^{(k)}. \tag{5.3}$$

5.1 The case of generalized companion matrices

Now, we consider a more specific implementation of the shifted inverse power method for the matrix $C = D - E$ of (4.2) and (4.3). Observe that for the matrix $C = D - E$ the deflation of each approximated eigenvalue can be performed automatically. Let $s_1, \ldots, s_n$ be the initial approximations to the eigenvalues and

$$d_i = \frac{p(s_i)}{\prod_{j \neq i}(s_i - s_j)}$$

be the Durand-Kerner corrections. Let us denote z the computed eigenvalue of C, and without loss of generality assume that s_n is the initial approximation closest to z (recall that we can sort the initial approximations in any order).

Remark 5.3. Let $\hat{\mathbf{s}} = (\hat{s}_i) = (s_1, \ldots, s_{n-1}, z)^T$ and $\hat{\mathbf{d}} = (\hat{d}_i)$ be the vector of the Durand-Kerner corrections corresponding to $\hat{\mathbf{s}}$ such that

$$\hat{d}_i = \frac{p(\hat{s}_i)}{\prod_{j \neq i}(\hat{s}_i - \hat{s}_j)}. \tag{5.4}$$

Since $p(z) = 0$, then the last row of $C_{\hat{\mathbf{s}},\hat{\mathbf{d}}}$ is $(0, \ldots, 0, z)$. Therefore the $(n-1) \times (n-1)$ leading principal submatrix of $C_{\hat{\mathbf{s}},\hat{\mathbf{d}}}$ coincides with the generalized companion matrix associated with the deflated polynomial $p(x)/(x-z)$ and with the vector $(s_1, \ldots, s_{n-1})$. This matrix is defined by the $(n-1)$-dimensional vectors $(s_1, \ldots, s_{n-1})^T$, and by the Durand-Kerner corrections $(\hat{d}_1, \ldots, \hat{d}_{n-1})^T$. The components $s_1, \ldots, s_{n-1}$ are available whereas the values $\hat{d}_1, \ldots, \hat{d}_{n-1}$ must be computed. From (5.4) we immediately deduce that

$$\hat{d}_i = d_i \frac{s_i - s_n}{s_i - z}, \quad i = 1, \ldots, n-1, \tag{5.5}$$

which provides a simple means for performing deflation with $2(n-1)$ additions, $(n-1)$ divisions and $(n-1)$ multiplications.

According to Remark 5.3 the approximation of the eigenvalues of $C_{\mathbf{s},\mathbf{d}}$ can be performed by applying the shifted inverse power method to a sequence of $k \times k$ matrices $C^{(k)}$ for $k = n, n-1, \ldots, 2$, where $C^{(k)}$ is the matrix obtained by updating $C^{(k+1)}$ according to (5.4) once the inverse power method, applied to $C^{(k+1)}$ has arrived at convergence. The eigenvalue of the matrix $C^{(1)}$ is just $s_1 - d_1$. The resulting algorithm, with the dynamical choice of the shift value performed according to Remark 5.1, is described below:

Algorithm 5.4. [Computing all the eigenvalues of a g.c.m.]

INPUT: An integer n and the vectors $\mathbf{s} = (s_1, \ldots, s_n)^T$, $\mathbf{d} = (d_1, \ldots, d_n)^T$; the maximum number of iterations N; an error bound ϵ.

OUTPUT: Approximations $(\xi_1, \ldots, \xi_n)$ to the eigenvalues of the matrix $C_{\mathbf{s},\mathbf{d}}$.

COMPUTATION:

sort $s_1, \ldots, s_n$, so that $|s_i| \leq |s_j|$ for $i > j$; apply the same permutation to $d_1, \ldots, d_n$.

`For` $j = n, n-1, \ldots, 1$ `do`

`If` $|d_j| > |s_j|$ `then` set $z = s_j(1+\epsilon)$, $\mathbf{v} = \mathbf{e}_j$, `else` set $z = s_j - d_j$, $\mathbf{v} = ((s_j - z)/(s_i - z))$ (compare with (4.5))

Set $\nu = 0$, $err = 1$.

`While` $err > \epsilon$ `and` $\nu < N$ `do`

$\nu = \nu + 1$

Apply a single shifted inverse power step with $\mathbf{x} = \mathbf{v}$ according to Algorithm 4.8 and get $\mathbf{y}$

set $z_{new} = s_j - (\sum_i d_i y_i)/y_j$ (compare with (5.3))

set $err = |(z_{new} - z)/z_{new}|$, $z = z_{new}$

`End while`

`If` $err < \epsilon$ `Then` output $\xi_j = z$, `else` output `Failure.`

Choose the s_i closest to ξ_j (compare with the next Remark 5.5), set $s_j = \xi_j$ and exchange s_j with s_i and d_j with d_i.

Update the Weierstrass (D-K) corrections $d_i = d_i * (s_i - s_j)/(s_i - z)$, $i = 1, \ldots, j-1$ by means of (5.5).

`End do`

Output $\xi_1 = s_1 - d_1$

Remark 5.5. The choice of the starting vector for the shifted inverse power iteration is mainly heuristic. In many cases a relatively large value of d_j does not correspond to a great distance from s_j to the closest eigenvalue. Moreover, if $|d_j| \leq |s_j|$ then the modulus of the D-K correction is smaller than the modulus of the approximation and it is likely that the approximation s_j is not much far from an eigenvalue. Once ξ_j have been computed, the algorithm selects the approximation s_i closest to ξ_j. Actually, this is not the best strategy. What we have really implemented is the closeness condition $\min(|s_i - \xi_j| + 2||s_i| - |\xi_j||)$. In fact, when applying the shifted inverse power method for computing zeros of polynomials (see the next section), we may have "approximations" $s = \rho e^{i\theta}$ to some root $\rho' e^{i\theta'}$ where $\rho \approx \rho'$ are very large and θ may be much different from θ', say, $\theta = 0$, $\theta' = \pi$. In this case the approximation $s = \rho e^{i\theta}$ would be closer to an eigenvalue with very small modulus than to $\rho' e^{i\theta'}$. However, the inverse power algorithm generally provides a better convergence to $\rho' e^{i\theta'}$ if using the shift $z = \rho e^{i\theta}$.

5.2 An algorithm for computing the zeros of a polynomial

The inverse power method applied to a generalized companion matrix can be used for implementing an efficient polynomial rootfinder in the same fashion of [F01].

We first recall the following useful result of [B96] which allows us to say if a given approximation ξ of a zero of $p(x)$ is zero of a slightly perturbed polynomial. This will be used as stop condition for terminating the iterations.

Theorem 5.6. *Let $p(x)$ be the polynomial of (2.1) and ξ a complex number. Denote with $fl(p(\xi))$ the value obtained by computing $p(\xi)$ by means of Horner rule*

$$\begin{aligned} &u_0 = p_n, \\ &u_{i+1} = \xi u_i + p_{n-i-1}, \quad i = 0, \ldots, n-1, \\ &p(\xi) = u_n \end{aligned}$$

in floating point arithmetic with machine precision μ. If

$$|\mathrm{fl}(p(\xi))| \leq \delta \sum_{i=0}^{n} |a_i||\xi|^i, \qquad \delta = (12n+3)\mu, \tag{5.6}$$

then there exists a polynomial $\tilde{p}(x) = \sum_{i=0}^{n} \tilde{a}_i x^i$ such that $\tilde{a}_i = a_i(1+\epsilon_i)$, $|\epsilon_i| \leq \delta$, and $\tilde{p}(\xi) = 0$. If the inequality (5.6) is not satisfied then for any polynomial $\tilde{p}(x)$ such that $|\epsilon_i| \leq \frac{1}{3}\delta$ it holds $\tilde{p}(\xi) \neq 0$.

If equation (5.6) is satisfied we say that ξ is a δ-approximated zero of $p(x)$.

We use also the criterion of [B96] for selecting initial approximation to the zeros of $p(x)$ based on Rouché theorem and on Newton's polygon. For more details on this criterion we refer the reader to [B96].

Algorithm 5.7. [Approximating plynomial zeros]

INPUT: The degree n and the coefficients $a_0, \ldots, a_n$ of the polynomial (2.1)

OUTPUT: Approximations $\xi_1, \ldots, \xi_n$ to the zeros of $p(x)$ such that (5.6) is satisfied for $\xi = x_i$, $i, 1, \ldots, n$.

COMPUTATION:

Compute initial approximations $s_1, \ldots, s_n$ to the zeros of $p(x)$ by means of the criterion of [B96]. Set $m = n$, $\delta = (12n+3)\mu$, where μ is the machine precision.

`While` $m > 0$ `do`

compute the D-K corrections $d_1, \ldots, d_n$ defined by (2.4) and check if s_i is a δ-approximated zero of $p(x)$.

Sort s_i so that the δ-approximated components are at the bottom and the components which are not yet δ-approximated are ordered with nonincreasing modulus.

Denote with m the number of components which are not yet δ-approximated.

Apply the shifted inverse power method (Algorithm 5.4) to the $m \times m$ generalized companion matrix $C_{\mathbf{s},\mathbf{d}}$ defined by $s_1, \ldots, s_m$, $d_1, \ldots, d_m$, and output approximations $\xi_1, \ldots, \xi_m$. Set $s_i = \xi_i$, $i = 1, \ldots, m$.

`End While`

5.3 Numerical experiments

The algorithm 5.7 has been implemented in Fortran 90 (the program can be downloaded at `www.dm.unipi.it/~bini`) and compared with the simple Durand-Kerner iteration implemented in the Gauss-Seidel style. In order to avoid overflow in the computation of the product $\prod_{j=1,\ j\neq i}^{n}(x_i - x_j)$, we have replaced this product with its logarithm both in our algorithm and in the Durand-Kerner method. The algorithm has been tested with the following set of polynomials:

- *Roots of the unity*: $x^n - 1$. All the zeros are well conditioned.

- *Mignotte-like polynomial*: $x^n + (100x - 1)^3$. This is a particular class of polynomials obtained by modifying certain polynomials introduced by Maurice Mignotte which almost reach the Mahler bound to the separation of the roots. The polynomials which we have considered in this class have three zeros clustered around 1/100 and, therefore, very ill-conditioned, the remaining ones are roughly displaced along a circle.

- *Unbalanced zeros*: $x^n + 10^{100}x^{n-3} + 10^{100}x^3 + 10^{-200}$. Three zeros of this polynomial have very large moduli and three zeros have very small moduli. Here it is crucial to use the starting criterion of [B96].

- *Mandelbrot polynomial*: The polynomial is recursively defined by means of the relation

  $$\begin{aligned} &m_0(x) = 1 \\ &m_{i+1}(x) = zm_i(x)^2 - 1, \quad i = 0, 1, \ldots, k-1, \quad n = 2^k - 1. \end{aligned}$$

  The value of the Mandelbrot polynomial at a point is computed by means of the above relations in $O(\log_2 n)$ ops by a suitable subroutine. In order to avoid overflow/underflow situations the program computes the logarithm of $p(x)$. The stop condition is obtained by computing an upper bound to the relative error in the floating point computation of $p(x)$, by using a criterion similar to the one of Theorem 5.6.

  The zeros of this class of polynomials determine the Mandelbrot set. Therefore they are clustered in a fractal and this feature makes it difficult to approximate all the zeros.

The results are reported in tables 2-5 where n denotes the degree of the polynomial; cpu is the cpu time needed by our algorithm (on the left) and by Durand-Kerner algorithm (on the right) run over a laptop with a Celeron^{TM} cpu; sweeps denotes the number of times that a new generalized companion matrix is generated and that its eigenvalues are computed; w-iter denotes the overall weighted number of shifted inverse power iterations, where the weight is m/n with m the size of the matrix to which the iteration is applied; finally, iter denotes the overall number of iterations of the Durand-Kerner method (a whole sweep performed on all the n approximations counts n iterations).

From the experiments that we have performed it results that our implementation is significantly faster than the Durand-Kerner method. Moreover the performances of our algorithm get better as the degree of the polynomial increases. This makes our approach a valid tool for replacing the QR iteration technique used by S. Fortune in the implementation of his polynomial rootfinder. In fact, we remark that our algorithm can compute all the eigenvalues of a generaligzed companion matrix in $O(n^2)$ ops per step, whereas the QR iteration has a cost of $O(n^3)$ ops per step.

n	cpu	sweeps	w-iter.	cpu	iter.
20	0.03	1	52	0.02	130
50	0.05	1	190	0.05	682
100	0.09	1	251	0.1	658
200	0.35	2	684	0.4	1537
500	1.9	2	1767	15.2	22585
1000	9.5	2	4394	12.8	9675
2000	37.0	2	6012	349.1	126431

Table 2: Polynomial $x^n - 1$

n	cpu	sweeps	w-iter.	cpu	iter.
20	0.01	1	99	0.01	206
50	0.02	2	204	0.12	1364
100	0.11	2	333	0.14	900
200	0.4	2	844	0.6	2066
500	1.6	2	1165	7.7	11018
1000	9.3	2	4196	47.4	34671
2000	25.6	1	3053	342.4	44156

Table 3: Polynomial $x^n + (100x - 1)^3$

n	cpu	sweeps	w-iter.	cpu	iter.
15	0.01	2	131	0.01	192
31	0.02	2	477	0.05	623
63	0.12	3	1050	0.21	2009
127	0.79	6	3993	1.24	7177
255	4.3	12	10685	9.8	28404
511	31.5	26	40555	78.7	116974
1023	265	58	167149	623	453734

Table 4: Mandelbrot polynomial

n	cpu	sweeps	w-iter.	cpu	iter.
20	0.02	2	72	0.02	224
50	0.04	2	189	0.07	514
100	0.09	2	253	0.11	598
200	0.4	2	597	0.39	1376
500	2.9	2	2403	3.6	5511
1000	10.4	2	3438	10.3	7834
2000	51.5	2	9103	94.6	36154

Table 5: Polynomial $x^n + 10^{100}x^{n-3} + 10^{100}x^3 - 10^{-200}$

References

[B96] D. A. Bini, Numerical computation of polynomial zeros by means of Aberth's method, *Numerical Algorithms*, 13 (1996), no. 3-4, 179–200.

[BF00] D. A. Bini, G. Fiorentino, Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder, *Numerical Algorithms*, **23**: 127–173, 2000.

[BPa] D. A. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Vol.2: Fundamental and Practical Algorithms*, Birkhäuser, Boston, to appear.

[C91] C. Carstensen, On a Linear Construction of Companion Matrices, *Linear Algebra and its Applications*, **149**: 191-214, 1991.

[C91a] C. Carstensen, Inclusion of the Roots of a Polynomial Based on Gershgorin's Theorem, *Numerische Math.*, **59**: 349-360, 1991.

[C92] C. Carstensen, On Grau's Method for Simultaneous Factorization of Polynomials, *SIAM J. of Numerical Analysis*, **29, 2**: 601-613, 1992.

[DS93] J.J. Dongarra, M. Sidani, A Parallel Algorithm for the Nonsymmetric Eigenvalue Problem, *SIAM J. Sci. Computing*, **14**: 242-269, 1993.

[E73] L. Elsner, A Remark on Simultaneous Inclusions of the Zeros of a Polynomial by Gershgörin Theorem, *Numer. Math.*, **21**: 425-427, 1973.

[EG97] Y. Eidelman, I.Gohberg, Inversion Formulas and Linear Complexity Algorithm for Diagonal Plus Semiseparable Matrices, *Computers and Math. (with Applications)*, **33**: 69-79, 1997.

[EM95] A. Edelman, H. Murakami, Polynomial Roots from Companion Matrix Eigenvalues, *Mathematics of Computation*, **64**: 763-776, 1995.

[F90] M. Fiedler, Expressing a Polynomial as the Characteristic Polynomial of a Symmetric Matrix, *Linear Algebra and Its Applications*, **141**: 265–270, 1990.

[F01] S. Fortune, Polynomial Root Finding Using Iterated Eigenvalue Computation, *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC'01)*, 121–128, ACM Press, New York, 2001.

[GL96] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 3rd edition, The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[HSS01] J. Hubbard, D. Schleicher, S. Sutherland, How to Find All Roots of Complex Polynomials by Newton's Method, *Inventiones Math.*, **146**: 1-33, 2001.

[KS94] M-hi Kim, S. Sutherland, Polynomial Root-finding Algorithms and Branched Covers, *SIAM J. on Computing*, **23, 2**: 415–436, 1994.

[McN93] J. M. McNamee, Bibliography on Roots of Polynomials, *J. Comp. Appl. Math.*, **47**: 391-394, 1993.

[McN97] J. M. McNamee, A Supplementary Bibliography on Roots of Polynomials, *J. Computational Applied Mathematics*, **78, 1**, 1997, http://www.elsevier.nl/homepage/sac/cam/mcnamee/index.html.

[MCVH01] N. Mastronardi, S. Chandrasekaran, S. Van Huffel, Fast and Stable Two-Way Algorithm for Diagonal Plus Semi-Separable Systems of Linear Equations, *Numer. Linear Algebra Appl.*, **8, 1**: 7-12, 2001.

[MV95a] F. Malek, R. Vaillancourt, Polynomial Zerofinding Iterative Matrix Algorithms, *Computers and Math. (with Applications)*, **29, 1**: 1–13, 1995.

[MV95b] F. Malek, R. Vaillancourt, A Composite Polynomial Zerofinding Matrix Algorithm, *Computers and Math. (with Applications)*, **30, 2**: 37–47, 1995.

[P95] V. Y. Pan, Optimal (up to Polylog Factors) Sequential and Parallel Algorithms for Approximating Complex Polynomial Zeros, *Proc. 27th Ann. ACM Symp. on Theory of Computing*, 741–750, ACM Press, New York, May, 1995.

[P97] V. Y. Pan, Solving a Polynomial Equation: Some History and Recent progress, *SIAM Review*, **39, 2**, 187–220, 1997.

[P01] V. Y. Pan, Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *Proc. Intern. Symposium on Symbolic and Algorithmic Computation (ISSAC'01)*, 253-267, ACM Press, New York, 2001.

[P02] V. Y. Pan, Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *Journal of Symbolic Computations*, **33, 5**, 701-733, 2002.

[PHI98] M. S. Petkovic, D. Herceg, S. Illic, Safe Convergence of Simultaneous Method for Polynomials Zeros, *Numer. Algorithms*, **17**: 313-331, 1998.

[TT94] K. C. Toh, L. N. Trefethen, Pseudozeros of Polynomials and Pseudospectra of Companion Matrices, *Numerische Math.*, **68**: 403–425, 1994.