2002

TR-2002003: Univariate Polynomial Root-Finding with a Lower Computational Precision and Higher Convergence Rates

V. Y. Pan

Univariate Polynomial Root-Finding with a Lower Computational Precision and Higher Convergence Rates

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

May 29, 2002

Abstract

Univariate polynoimial root-finding is an oldest classical problem, which is still an important research topic, due to its impact on computational algebra and geometry. The Weierstrass (Durand–Kerner) approach and its variations are most popular practical choices for simultaneous approximation of all roots of a polynomial, but these methods require computations with a high multiple precision. We apply some novel techniques of structured matrix computations to avoid this serious deficiency, thus giving decisive acceleration to the approach. We also show two ways (based on the Lagrange interpolation formula and on Newton's iteration for the eigenproblem for a generalized companion matrix) to unifying the derivation of the Weierstrass (Durand–Kerner) algorithm (having quadratic convergence) and its extensions having convergence rates $4, 6, 8, \ldots$, and we study application of the inverse power iteration to (generalized) companion matrix for polynomial root-finding.

*Supported by NSF Grant CCR 9732206 and PSC CUNY Award 66383-0032

Key words: polynomial root-finding, computational precision, Weierstrass (Durand–Kerner) algorithm, higher order root-finders.

2000 Math. Subject Classification: 65H05, 65Y20, 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 resolved in [P95],[P01a],[P02], where optimal (up to polylog factors) computational time bounds were reached under both arithmetic and Boolean computational models. The algorithms in [P95],[P01a],[P02], however, are quite involved, which complicates their 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 (Erlich / 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 they have deficiencies as well: they all require computations with a high precision. This causes substantial slowdown, so matrix methods based on the QR algorithms become competitive. The latter methods (see [F90], [TT94], [EM95], [MV95a], [MV95b], [F01], and the bibliography therein) use the order of n^3 flops per iteration (versus $O(n^2)$ in the W(D–K) iteration and its modifications) but can be performed with single precision.

In the present paper, we exploit the association with structured matrices

to devise a novel version of the W(D–K) iteration with the same convergence properties where computations are performed with a lower (single) precision. The approach can be extended to yield a similar effect for the above cited and other known variations of the W(D–K) iteration.

Our second innovation is a simple unified derivations of the W(D–K) iteration and its extensions having convergence rates $4, 6, 8, \ldots$.

Furthermore, we revisit the shifted inverse power iteration and apply it to the Frobenius or generalized companion matrices associated with the input polynomial (in Section 5). The latter quite natural but yet unused approach can be also applied as an independent root-finder particularly for approximating a single zero or a few zeros. It uses $O(n)$ flops per iteration step (for a selected zero), and we insure fast convergence by supplying simple explicit expression for the eigenvector of a generalized companion matrix in terms of its eigenvalues and their approximations. The derivation of this expression 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 6.

Apart from the latter subjects in Sections 5 and 6, we organize our presentation as follows. In the next section we recall the W(D–K) algorithm, based on Lagrange interpolation formula and then devise higher order root-finders. In Section 3 we recall Trummer's generalized problem of Cauchy matrix-by-vector multiplication and relate it to the W(D–K) algorithm. In Section 4 we modify the W(D–K) algorithm to decrease the precision of computing.

Acknowledgements: I thank S. Fortune, B. Mourrain, and E.E. Tyrtyshnikov for preliminary discussions on matrix methods for polynomial root-finding and X. Wang for helping me to simplify my original proof of equation (2.6).

2 The Weierstrass (Durand–Kerner) iteration

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 converges quadratically and only requires computation of the values $p(s_i)$ and $q'(s_i)$, $i = 1, \ldots, n$.

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

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

where $p'(x)$ is 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

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

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:

$$s_i - z_i = d_i/(1 + u_i) + O(\Delta^4), \tag{2.9}$$

where $u_i = \sum_{j \neq i} d_j/(s_i - d_i - s_j), i = 1, \ldots, n$.

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

$$s_i - z_i = d_i/(1 + v_i) + O(\Delta^6), \tag{2.10}$$

where $v_i = \sum_{j \neq i} d_j/(s_i - d_i/(1 + u_i) - s_j), i = 1, \ldots, n$.

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

$$s_i - z_i = d_i/(1 + w_i) + O(\Delta^8), \tag{2.11}$$

where $w_i = \sum_{j \neq i} d_j/(s_i - d_i/(1+v_i) - s_j), i = 0, \ldots, n$, 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. Computing $y_1, \ldots, y_n$ for given d_j, h_i and s_j is still simpler than computing $d_1, \ldots, d_n$ for given $p(x)$ and $s_1, \ldots, s_n$.

3 Trummer's generalized problems and the W(D–K) iteration

Each W(D–K) step requires multipoint evaluation of $p(x)$ and $q'(x)$. Next, we relate this task to Trummer's generalized problem, which can be solved by using the celebrated Fast Multipole Algorithm of [GR87]. This is not our final destination but a preliminary step on the path towards a modification of the W(D–K) iteration and its extensions, where the computations are performed with a lower (single) precision. We present this modification in Section 4.

Definition 3.1. *Hereafter* $\mathbf{v} = (v_i)_{i=1}^n$ *denotes the vector* $(v_1, \ldots, v_n)^T$, $\mathbf{1} = (1)_{i=1}^n$ *denotes the vector filled with ones,* $(M)_{i,j} = m_{i,j}$ *denotes the* (i,j)*-th entry of a matrix* M, I *is the* $n \times n$ *identity matrix,* $D_{\mathbf{v}}$ *denotes* $\mathrm{diag}(v_1, \ldots, v_n)$, *and* $\mathrm{diag}(M) = \mathrm{diag}(m_{i,i})_{i=1}^n$ *for a matrix* $M = (m_{i,j})_{i,j=1}^n$.

Definition 3.2. *Let us write*

$$C_k(\mathbf{u}, \mathbf{v}) = \left(\frac{1}{(u_i - v_j)^k} \right)_{i,j=1}^n;$$

$$C(\mathbf{u}, \mathbf{v}) = C_1(\mathbf{u}, \mathbf{v});$$

$$C_k(\mathbf{u}) = (c_{i,j}(\mathbf{u}))_{i,j=1}^n, \quad \textit{where} \quad c_{i,j}(\mathbf{u}) = \begin{cases} 0 & \textit{for } i = j \\ \frac{1}{(u_i - u_j)^k} & \textit{otherwise} \end{cases},$$

$$C(\mathbf{u}) = C_1(\mathbf{u}).$$

$C_k(\mathbf{u},\mathbf{v})$ and $C_k(\mathbf{u})$ for a fixed positive integer k are said to be *Cauchy matrices* (of degree k). In this paper we once encounter $C_2(\mathbf{u},\mathbf{v})$ in Section 4, otherwise $k=1$. The problems of multiplication by a vector of Cauchy matrices $C_k(\mathbf{u},\mathbf{v})$ and $C_k(\mathbf{u})$ are said to be *Trummer's generalized* and *Trummer's problems*, respectively. Their straightforward solution uses $O(n^2)$ flops. The algorithm of [GGS87] (see also [G88]) computes the solution by using $O(n\log^2 n)$ flops but leads to numerical stability problems. Approximate solution is computed by the numerically stable *Fast Multipole algorithm* of [GR87] using $O(n)$ flops (in terms of n). (The flop count also depends on the approximation error bound, on which we refer the reader to [GR87].) Hereafter, we refer to this algorithm as the *FM algorithm*. Every iteration step (2.9)–(2.12) involves the solution of Trummer's (generalized) problem.

Let us next follow [PLST93], [PZHY97], [PACLS98], [P01, page 91] to reduce the evaluation of the polynomials $p(x)$ and $q'(x)$ at the points $x=s_i$, $i=1,\dots,n$, to Trummer's generalized problems.

Definition 3.3. *Write $V(\mathbf{u})$ to denote an $n\times(n+1)$ Vandermonde matrix $(u_i^j)_{i=1,j=0}^{n,n}$, defined by its second column $\mathbf{u}=(u_i)_{i=1}^n$. Write*

$$q_{\mathbf{v}}(x)=\prod_{i=0}^{n}(x-v_i),\quad \mathbf{q}(\mathbf{t},\mathbf{v})=(q_{\mathbf{v}}(t_i))_{i=1}^n,\quad q'(\mathbf{v})=(q'_{\mathbf{v}}(v_i))_{i=0}^n. \tag{3.1}$$

Recall that the polynomial values $r(\mathbf{u})=(\sum_{i=0}^n r_i u_j^i)_{j=1}^n$ are given by the product $V(\mathbf{u})\mathbf{r}$, $\mathbf{r}=(r_i)_{i=0}^n$. Furthermore, recall the known expression (cf. [P01, page 91])

$$V(\mathbf{u})=D_{q(\mathbf{u},\mathbf{w})}C(\mathbf{u},\mathbf{w})D^{-1}_{q'(\mathbf{w})}V(\mathbf{w}), \tag{3.2}$$

which holds for any pair of vectors $\mathbf{u}=(u_i)_{i=1}^n$, $\mathbf{w}=(w_i)_{i=0}^n$, where $u_i\neq w_j$ for all pairs i and j, $w_h\neq w_j$ if $h\neq j$. (3.2) can be viewed as the matrix version of the Lagrange interpolation formula.

To multiply $V(\mathbf{u})$ by a vector $\mathbf{p}$, we fix any vector $\mathbf{w}$ at our convenience and recursively compute the vectors $\mathbf{x}=V(\mathbf{w})\mathbf{p}$, $\mathbf{y}=D^{-1}_{q'(\mathbf{w})}\mathbf{x}$, $\mathbf{z}=C(\mathbf{u},\mathbf{w})\mathbf{y}$, $V(\mathbf{u})\mathbf{p}=D_{q(\mathbf{u},\mathbf{w})}\mathbf{z}$. For example, choose the vector $\mathbf{w}=(\gamma\exp(2\pi k\sqrt{-1}/N))_{k=0}^{n-1}$, made up of $n+1$ distinct scaled N^{th} roots ot 1, $n<N=2^h<2n+2$, where γ is a scalar of our choice. Then the vector $\mathbf{x}$ is computed via FFT at N points by using $O(n\log\ n)$ flops; the vectors $\mathbf{y}$ and $V(\mathbf{u})\mathbf{p}$ (for given $\mathbf{x}$ and $\mathbf{z}$) are obtained in $2n+1$ flops by the diagonal

scaling of $\mathbf{x}$ and $\mathbf{z}$, and the vector $\mathbf{z}$ is computed from $\mathbf{y}$ by solving Trummer's generalized problem.

In this way, the computation of the values of $p(s_i)$ and $q'(s_i)$, $i = 1, \ldots, n$ can be reduced essentially to computing the coefficients of $q(x) = q_{\mathbf{s}}(x)$ (which can be done in $O(n \log^2 n)$ flops [P01, page 75], although with possible numerical stability problems; alternatively, we may directly compute $q_i(s_i), i = 1, \ldots, n$, in $O(n^2)$ flops, based on (2.4)), to FFT, and to the solution of Trummer's generalized problem with a matrix $C(\mathbf{s}, \mathbf{w})$ where the option of choosing any vector $\mathbf{w}$ (or any scalar γ in the sample choice of $\mathbf{w}$ above) can be used to simplify the problem. The cost of performing these algorithms is dominated by the cost of the solution of Trummer's (generalized) problem and the computation of the coefficients of $q(x) = q_{\mathbf{s}}(x)$ or the values of $q_i(s_i)$, that is, ranges from $O(n \log^2 n)$ to $O(n^2)$ flops. If the polynomial $p(x)$ is given by a black box subroutine for its evaluation, then the overall computational cost should cover the order of n invocations of this subroutine.

4 Reduction to lower precision computations

Suppose that s_i are given with the precision of b bits and that we seek improvement towards t_i by $O(l)$ bits. Then the straightforward computation of the values $p(s_i), q_i(s_i)$, and their ratios requires precision of the order of nb bits. Our next task is to compute the ratios by using roughly as many flops (up to a small constant factor) and $O(l)$ bit precision.

Let Δf denote $f^{new} - f^{old}$ where f may stand for the values d_i, $p(s_i)$, and $q_i(s_i)$, whereas the superscripts "old" and "new" mark the input and output values of the current iteration, respectively. To improve numerical stability of the computations in the W(D–K) iteration, we compute first Δf and then f^{new} as $f^{old} + \Delta f$, instead of computing f^{new} directly. Let us specify. Write

$$\begin{array}{c} \Delta s_i = \sigma_i, \quad \Delta p_i = p(s_i + \sigma_i) - p(s_i), \\ \Delta q_i = q_i(s_i + \sigma_i) - q_i(s_i), \quad \sigma = \max_i |\sigma_i|. \end{array} \tag{4.1}$$

Recall (2.4), and obtain that

$$\Delta d_i = \frac{p(s_i + \sigma_i)}{q_i(s_i + \sigma_i)} - \frac{p(s_i)}{q_i(s_i)} = \frac{p(s_i)(1 + \Delta p_i / p(s_i))}{q_i(s_i)(1 + \Delta q_i / q(s_i))} - \frac{p(s_i)}{q_i(s_i)},$$

so

$$\frac{\Delta d_i}{d_i} = \Delta d_i / \frac{p(s_i)}{q_i(s_i)} = \frac{\Delta p_i/p(s_i) - \Delta q_i/q_i(s_i)}{1 + \Delta q_i/q_i(s_i)} d_i, \quad i = 1, \ldots, n.$$

Now examine the values $\Delta p_i/p(s_i)$ and $\Delta q_i/q_i(s_i)$. Recall that $q(x)$ and $q_i(x)$ change when we change s_j into $s_j + \sigma_j$ for all j, that is,

$$q^{new}(x) = \prod_{j=1}^{n}(x - s_j - \sigma_j),$$

$$q_i^{new}(x) = \prod_{j \neq i}(x - s_j - \sigma_j)$$

(see (2.3),(2.4)). Therefore,

$$\Delta q_i/q_i(s_i) = (q_i(s_i + \sigma_i)/q_i(s_i)) - 1 = \prod_{j \neq i}(s_i - s_j + \sigma_i - \sigma_j)/(s_i - s_j) - 1 =$$

$$\prod_{j \neq i}(1 + \frac{\sigma_i - \sigma_j}{s_i - s_j}) - 1 =$$

$$\sum_{j \neq i} \frac{\sigma_i - \sigma_j}{s_i - s_j} + O(\sigma^2) =$$

$$\sigma_i \sum_{j \neq i} \frac{1}{s_i - s_j} - \sum_{j \neq i} \frac{\sigma_j}{s_i - s_j} + O(\sigma^2). \tag{4.2}$$

This reduces the computation to Trummer's problems of multiplication of the Cauchy matrix $C(\mathbf{s})$ by the vectors $\mathbf{1} = (1)_{j=1}^n$ and $(\sigma_j)_{j=1}^n$. As follows from (4.2), it is sufficient to perform the computations with a lower precision of the order of $\log(1/\sigma)$ bits to obtain the output values within the errors in $O(\sigma^2)$. By computing Δq_i directly, we also avoid the costly stages of computing the coefficients of $q'(x)$ and the values of $q'(x)$ and $p(x)$ for $x = s_i$ and $x = s_i + \sigma_i$ for all i.

A similar attempt of treating $\Delta p_i/p(s_i)$ leads to some difficulty. Indeed, we have

$$\frac{\Delta p_i}{p(s_i)} = \frac{p(s_i + \sigma_i) - p(s_i)}{p(s_i)} = \frac{p'(s_i)}{p(s_i)}\sigma_i + O(\sigma_i^2) =$$

$$\sigma_i \sum_{j=1}^{n} \frac{1}{s_i - z_j} + O(\sigma_i^2) = \sigma_i \sum_{j \neq i} \frac{1}{s_i - s_j - \sigma_j} + \frac{\sigma_i}{s_i - z_i} + O(\sigma_i^2),$$

where the term $\sigma_i/(s_i - z_i)$ is not easy to approximate within the error in $O(\sigma^2)$. We overcome the problem by using the approach of [PLST93],[PZHY97], [PACLS98],[P01, page 91].

That is, let us apply (3.1) and (3.2) for $\mathbf{u} = (s_i + \sigma_i)_{i=1}^n$ and for $\mathbf{u} = \mathbf{s}$ to evaluate $p(x)$ at the points $s_i + \sigma_i$ and s_i, $i = 1, \dots, n$, provided that $p(x)$ is given with its coefficient vector $\mathbf{p}$. Then again we fix a vector $\mathbf{w}$, e.g., a vector $\mathbf{w} = (w_j)_{j=0}^n = (\gamma \exp(2\pi j\sqrt{-1}/N))_{j=0}^{n-1}$ of scaled roots of 1, precompute the vector $\mathbf{y} = D_{q'(\mathbf{w})}^{-1} V(\mathbf{w})\mathbf{p} = (y_j)_{j=1}^n$, and obtain that

$$\Delta p_i = q_{\mathbf{w}}(s_i + \sigma_i) \sum_j \frac{y_j}{s_i + \sigma_i - w_j} - q_{\mathbf{w}}(s_i) \sum_j \frac{y_j}{s_i - w_j} =$$

$$(q_{\mathbf{w}}(s_i + \sigma_i) - q_{\mathbf{w}}(s_i)) \sum_j \frac{y_j}{s_i + \sigma_i - w_j} + q_{\mathbf{w}}(s_i) \sum_j (\frac{y_j}{s_i + \sigma_i - w_j} - \frac{y_j}{s_i - w_j}) =$$

$$((q'_{\mathbf{w}}(s_i) + O(\sigma_i)) \sum_j \frac{y_j}{s_i + \sigma_i - w_j} - q_{\mathbf{w}}(s_i) \sum_j \frac{y_j}{(s_i - w_j)^2 + O(\sigma)})\sigma_i.$$

By applying (3.1) and (3.2), we also obtain that

$$p(s_i) = q_{\mathbf{w}}(s_i) \sum_j \frac{y_j}{s_i - w_j}$$

and recall that $q'_{\mathbf{w}}(x)/q_{\mathbf{w}}(x) = \sum_{j=1}^n \frac{1}{x - w_j}$. Therefore,

$$\frac{\Delta p_i}{p(s_i)} = ((\sum_j \frac{1}{s_i - w_j} + O(\sigma)) \sum_j \frac{y_j}{s_i + \sigma_i - w_j} - \sum_j \frac{y_j}{(s_i - w_j)^2 + O(\sigma)})\sigma_i / \sum_j \frac{y_j}{s_i - w_j}.$$

Then again we avoid the evaluation of $q(s_i)$ and perform the computations with a lower precision of $\log(1/\sigma) + O(1)$ bits to obtain $\Delta p_i/p(s_i)$ with the error bound in $O(\sigma^2)$. By scaling the roots of 1 in the definition of $\mathbf{w}$, we keep the coordinates w_j at some distance from all approximations s_i to the roots of $p(x)$, thus increasing the denominators in the above expressions. Scaling can be updated if $|s_i - w_j|$ becomes small for some i and j.

Computing the vectors $(\sum_j \frac{1}{s_i - w_j})_{i=1}^n = C(\mathbf{s}, \mathbf{w})\mathbf{1}$, $(\sum_j \frac{y_j}{s_i - w_j})_{i=1}^n = C(\mathbf{s}, \mathbf{w})\mathbf{y}$, $(\sum_j \frac{y_j}{s_i + \sigma_i - w_j})_{i=1}^n = C((s_i + \sigma_i)_i, \mathbf{w})\mathbf{y}$, and $(\sum_j \frac{y_j}{(s_i - w_j)^2})_{i=1}^n = C_2(\mathbf{s}, \mathbf{w})\mathbf{y}$ amounts to solving Trummer's generalized problems. The overall number of flops required in this computation is asymptotically the same in the original unstabilized and the above stabilized methods, that is, $O(n^2)$ based on the

straightforward algorithms for Trummer's generalized problems and $O(n)$ based on the FM algorithms, but the precision of computing decreases to the level of $b = \max_i \log |s_i|$, instead of the original level of the order of nb.

Remark 4.1. *Our treatment of the ratio of $\Delta p_i/p(s_i)$ can be immediately extended to the ratios $\Delta p'_i/p'(s_i)$ and $\Delta(p'/p)_i/(p'(s_i)/p(s_i))$, and consequently, to lower precision computations for various extensions of the W(D–K) iteration such as Aberth's (Erlich's / Börsch–Supan's), Werner's, and Farmer–Loizou's iterations as well as our algorithms (2.9)–(2.11) in Section 2 having higher convergence rates.*

5 Refinement of the roots by the inverse power iteration

As an alternative or a complement to the W(D–K) and higher order iterations, one may refine the computed approximations to a root z_i by applying the shifted inverse power method to the associated *Frobenius (companion)* or *generalized companion* matrices. In this case every step of the shifted inverse power iteration uses only $O(n)$ flops, e.g., due to the (Sherman–Morrison–) Woodbury formula [GL96, page 50]. Let us further accentuate the power of this customary algorithm, which converges most rapidly when both an eigenvalue and the associated eigenvector are closely approximated. We assume that some initial approximations to the eigenvalues are available and extend them to the eigenvectors. That is, we derive some simple expressions for the eigenvectors via the eigenvalues z_i and their given approximations $s_i, i = 1, \ldots, n$.

In the case of a Frobenius (companion) matrix

$$F(\mathbf{p}) = \begin{pmatrix} 0 & 1 & & \\ & \cdot & \cdot & \\ & & \cdot & \cdot \\ & & & \cdot & 1 \\ -p_0 & -p_1 & & & -p_{n-1} \end{pmatrix},$$

where $\mathbf{p} = (p_i)_{i=0}^{n-1}, p(x) = x^n - \sum_{i=0}^{n-1} p_i x^i$ in (2.1), the eigenvectors form the (transposed) Vandermonde matrix $(z_i^j)_{i,j=0}^{n-1}$. For a generalized companion

matrix (see (5.1),(5.3) below), we are going to derive similar expressions. Our derivation is of independent interest because it reveals the corellation between the W(D–K) algorithm and Newton's method for the eigenproblem for the generalized companion matrices and enables an alternative derivation of (2.6).
We first recall the basic definition.

Definition 5.1. *(Cf. [E73],[C91],[C92].) For a polynomial $p(x)$ of (2.1) and n distinct values $s_1, \dots, s_n$, define a rank-one matrix $E_{\mathbf{d}}$ with diagonal entries $d_1, \dots, d_n$ of (2.4) and an associated $n \times n$* generalized companion matrix

$$C = C_{\mathbf{s},\mathbf{d}} = D_{\mathbf{s}} - E_{\mathbf{d}}. \tag{5.1}$$

Definition 5.1 leaves us with some freedom in choosing the matrix $E_{\mathbf{d}}$. In particular, Fiedler in [F90] proposes $E_{\mathbf{d}} = \sigma \mathbf{f}\,\mathbf{f}^T$, $\mathbf{f} = (f_i)_{i=1}^n$, $\sigma f_i^2 = d_i$, $i = 1, \dots, n$, for a fixed scalar σ, whereas Elsner in [E73] proposes

$$E_{\mathbf{d}} = \mathbf{1}\mathbf{d}^T. \tag{5.2}$$

Theorem 5.2. *(Cf. [C91].) For any pair of matrices C and $E_{\mathbf{d}}$ of Definition 5.1, we have*

$$\det(xI - C) = p(x). \tag{5.3}$$

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 $d_i q(x)/(x - s_i) = d_i q_i(x)$, $i = 1, \dots, n$.
Now (5.3) follows from the Lagrange interpolation formula (2.5). □

Now, suppose that $z_i \approx s_i$ for all i and $E_{\mathbf{d}}$ satisfies (5.2) and try to go from the eigenvalues/eigenvectors pair $(\mathbf{s}, I)$ of the matrix $D_{\mathbf{s}}$ to the eigenpair $(\mathbf{z}, I + Y)$ of the matrix $C = D_{\mathbf{s}} - E_{\mathbf{d}}$ of (5.1) (cf. [DS93]). We have $(D_{\mathbf{s}} - E_{\mathbf{d}})(I + Y) = (I + Y)D_{\mathbf{z}}$. It follows that

$$-E_{\mathbf{d}}(I + Y) + D_{\mathbf{s}}Y = YD_{\mathbf{s}} + D_{\mathbf{z}-\mathbf{s}} + YD_{\mathbf{z}-\mathbf{s}}. \tag{5.4}$$

Let us scale the matrix $I + Y$ of the right eigenvectors of $C = D_{\mathbf{s}} - E_{\mathbf{d}}$ to yield

$$y_{i,i} = 0 \text{ for all } i. \tag{5.5}$$

By equating the diagonal entries on both sides of (5.4) and applying (5.5), we obtain that

$$D_{\mathbf{z}-\mathbf{s}} = -\operatorname{diag}(E_{\mathbf{d}}(I + Y)),$$

that is,

$$z_i - s_i + d_i = -(E_{\mathbf{d}}Y)_{i,i}, i = 1, \ldots, n.$$

Substitute (5.2) and (5.5) and obtain that

$$z_i - s_i + d_i = -\sum_{j \neq i} d_j y_{j,i} \text{ for all } i. \tag{5.6}$$

Now, equate the off-diagonal entries (j, i) on both sides of (5.4) to obtain that

$$(YD_{\mathbf{z}} - D_{\mathbf{s}}Y)_{j,i} = -(E_{\mathbf{d}}(I + Y))_{j,i}, j \neq i,$$

that is (cf. (5.2) and (5.5)),

$$(s_j - z_i)y_{j,i} = (E_{\mathbf{d}}(I + Y))_{j,i} = d_i + \sum_{j \neq i} d_j y_{j,i}, j \neq i,$$

or equivalently,

$$-\sum_{j \neq i} d_j y_{j,i} = d_i + (z_i - s_j)y_{j,i}, j \neq i.$$

Substitute this into (5.6) and obtain that

$$y_{j,i} = \frac{z_i - s_i}{z_i - s_j}, j \neq i.$$

By combining the latter equations and (5.5), we express the matrix $I + Y$ via s_h and $z_h, h = 1, \ldots, n$, as the following Loewner matrix,

$$(I + Y)_{i,j} = \frac{z_i - s_i}{z_i - s_j} = 1 - \frac{s_i - s_j}{z_i - s_j}, i, j = 1, \ldots, n. \tag{5.7}$$

Combining (5.5)–(5.7) yields the equations

$$s_i - z_i = d_i + (z_i - s_i)\sum_{j \neq i} d_j/(z_i - s_j),$$

and we arrive at (2.6) in an alternative way.

6 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 n^{th} 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. 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 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 [KS94],[PHI98],[HSS01], and [BPa] on these and some other choices.

References

[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.

[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.

[G88] A. Gerasoulis, A Fast Algorithm for the Multiplication of Generalized Hilbert Matrices with Vectors, *Mathematics of Computation*, **50, 181**: 179–188, 1988.

[GGS87] A. Gerasoulis, M. D. Grigoriadis, L. Sun, A Fast Algorithm for Trummer's Problem, *SIAM Journal on Scientific and Statistical Computing*, **8, 1**: 135–138, 1987.

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

[GR87] L. Greengard, V. Rokhlin, A Fast Algorithm for Particle Simulation, *Journal of Computational Physics*, **73**: 325–348, 1987.

[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.

[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, Structured Matrices and Polynomials: Unified Superfast Algorithms, Birkhäuser/Springer, Boston/New York, 2001.

[P01a] V. Y. Pan, Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *Proc. Intern. Symposium on symbolic and Algorithmic Computation*, 253-267, ACM Press, New York, 2001.

[P02] V. Y. Pan, Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *Journal of Symbolic Computations (special issue)*, July 2002 (to appear).

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

[PLST93] V. Y. Pan, E. Landowne, A. Sadikou, O. Tiga, A New Approach to Fast Polynomial Interpolation and Multipoint Evaluation, *Computers & Mathematics (with Applications)*, **25, 9**: 25–30, 1993.

[PZHY97] V. Y. Pan, A. Zheng, X. Huang, Y. Yu, Fast Multipoint Polynomial Evaluation and Interpolation via Computations with Structured Matrices, *Annals of Numerical Mathematics*, **4**: 483–510, 1997.

[PACLS98] V. Y. Pan, M. AbuTabanjeh, Z. Chen, E. Landowne, A. Sadikou, New Transformations of Cauchy Matrices and Trummer's Problem, *Computers & Mathematics (with Applications)*, **35, 12**: 1–5, 1998.

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