Universal Journal of Applied Mathematics & Computation **2 (2014), 16-34**
www.papersciences.com

A Post-Fitting Algorithm for High Precision Complex Polynomial Root Finding

Peter Strobach
AST-Consulting Inc., Bahnsteig 6, 94133 Röhrnbach, Germany
peter_strobach@gmx.de

Abstract
A noniterative algorithm for root refinement of univariate polynomials with real or complex coefficients is introduced. The method uses a convolutional model which is fitted onto the coefficient sequence of a given polynomial. If initialized with root estimates from a conventional polynomial root finding algorithm like POLZEROS, the algorithm can double the number of accurate digits of these root estimates. Simulation results are shown for several types of polynomials which typically occur in signal and array processing. For instance, random coefficient polynomials up to degree $n = 64000$, where we reduced the absolute root errors of the POLZEROS root-finder by a factor of approximately 1000, or the roots of a complex chirp polynomial of degree $n = 2000$, where we reduced the absolute root errors by a factor of 10000 using the new method. Fortran subroutines of this high precision root refinement algorithm for real or complex polynomials are available upon request.

2010 Mathematics Subject Classification: 12D05, 12D10.
Key Word and Phrases
Polynomial Root Finding · Root Refinement · Aberth-Ehrlich Method · POLZEROS · Convolution.

1. Introduction

Root finding can be a demanding problem depending on the coefficient characteristics of a given polynomial. Random coefficient polynomials can be rooted up to virtually unlimited degrees as a consequence of the limited variance of their coefficients. On the other hand, Wilkinson-type polynomials [1] can be rooted only up to a degree of $n = 20$ in a standard double precision floating-point environment because of the enormous growth of their coefficients.

Our target are high-degree polynomials with moderate coefficient variance. Polynomials of this kind typically occur in signal and array processing [2]-[10]. Examples are the z-transform polynomials of long random sequences, for instance in problems of detecting signals in noise. The roots of these polynomials are tightly grouped around the unit circle in the z-plane. Examples of this kind can be found in seismic signal processing, in sonar and radar, and in speech processing.

The degrees of polynomials in these areas can be excessively high. Hence finding the roots of these polynomials is a computationally demanding problem. Multiprecision root finders like MPsolve [11] cannot be applied here because of excessive runtimes. Standard double precision root finders are sufficiently fast but inaccurate.

One of the top double precision root finders is POLZEROS [12], a modern and reliable implementation of the Aberth-Ehrlich method [13], [14]. We can demonstrate experimentally that the accuracy of the root estimates obtained from POLZEROS can be improved dramatically for typical signal and array processing type polynomials when the root estimates are passed through a properly constructed root refinement algorithm.

We propose a complex variant of polynomial fitting for root refinement. The principle is known from [15], [16] and [17]. In its root refinement variant, the method approximates a given "clean" coefficient sequence of a degree n polynomial with a model sequence formed as the convolution of a degree-1 desired root subpolynomial and a degree $n-1$ complementary subpolynomial. Initially, the desired root subpolynomial is formed using a given coarse root estimate. The complementary subpolynomial is initially determined in the least squares sense. The fitting step then optimizes both the desired root and the complementary subpolynomial sequences simultaneously for a perfect match with the given clean input polynomial sequence. Hereby, a root estimate of maximum accuracy is obtained. The procedure is repeated independently for each root of interest.

Typical improvements in root accuracy obtained by the method are a factor of 1000 reduction in absolute root errors for random coefficient polynomials of degree $n = 64000$, or typically 3 additional accurate mantissa digits in this case. Another example shown are the roots of a complex chirp polynomial of degree $n = 2000$, where we obtain 4 additional accurate mantissa digits. In all cases, we used `POLZEROS` as initial root finder. The overall runtime of `POLZEROS` with post refinement is about twice the runtime of the plain `POLZEROS` algorithm, which is one of the fastest root finding algorithms in double precision arithmetic known at this time.

This paper is organized as follows. In Section 2, we develop the post-fitting algorithm for complex polynomial root refinement. In Section 3, we extend our technique to polynomials with real coefficients. A special technique for handling complex conjugate root pairs is introduced. Detailed numerical examples demonstrating the effectiveness of our post fitting refinement method are shown in Section 4. Section 5 presents the conclusions.

2. The Complex Post-Fitting Refinement Algorithm

Consider the problem of calculating all roots $\{z_k, k = 1, 2, \ldots, n\}$ of the following monic polynomial $A(z)$ of degree n:

$$A(z) = z^n + a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_{n-1} z + a_n \quad , \tag{2.1}$$

with complex coefficients $\{a_k, k = 1, 2, \ldots, n\}$. We assume that we have a set of coarse root estimates $\{\hat{z}_k, k = 1, 2, \ldots, n\}$ at our disposal. A good choice for computing these initial root estimates is `POLZEROS` [12]. This program is an implementation of a simultaneous root-finder that can be traced back to algorithms of O. Aberth [13] and L.W. Ehrlich [14]. The Fortran code `POLZEROS` of the Aberth-Ehrlich (ABE) algorithm can be downloaded from `http://bezout.dm.unipi.it/`. This program and method is very fast and stable in a sense that difficult scenarios are handled without excessively large errors. Hence the method is a good initial root estimator.

In most cases of interest, `POLZEROS` produces its errors in a way suitable for a substantial correction by appropriate postprocessing refinement. A tricky algorithm for achieving this is next developed. For this purpose, we decompose the given $A(z)$ into the product of a linear factor $C(z)$ and a reduced degree polynomial $B(z)$ as follows,

$$A(z) = B(z)C(z) \quad , \tag{2.2}$$

where:

$$B(z) = z^{n-1} + b_1 z^{n-2} + \cdots + b_{n-2} z + b_{n-1} \quad , \tag{2.3}$$

$$C(z) = z + c \quad , \tag{2.4}$$

and $-c$ is a root of $A(z)$. We apply this model to every root of $A(z)$. Consequently, we have n

cases constituted by:

$$c = -z_k\,;\quad k = 1, 2, \ldots, n \quad . \tag{2.5}$$

Suppose we have a set of initial root estimates $\{\hat{z}_k, k = 1, 2, \ldots, n\}$ from an algorithm like `POLZEROS` at our disposal. Then we will be able to construct a highly effective refinement algorithm for computing updates Δc_k so that

$$\overline{z}_k = \hat{z}_k - \Delta c_k\,;\quad k = 1, 2, \ldots, n \quad , \tag{2.6}$$

where $\{\overline{z}_k\}$ is a refined set of root estimates with highest possible accuracy. The algorithm for computing the necessary root updates $\{\Delta c_k, k = 1, 2, \ldots, n\}$ is next developed. Given the initial setting

$$c = -\hat{z}_k\,;\quad k = 1, 2, \ldots, n \quad , \tag{2.7}$$

we define a fitting error polynomial $E(z)$ as follows:

$$E(z) = A(z) - B(z)C(z) \quad . \tag{2.8}$$

The goal is a modification of the c that constitutes $C(z)$ and the coefficients of $B(z)$ simultaneously in a way so that the coefficients of $E(z)$ are minimized. This gives rise to a nonlinear optimization or "fitting" problem, because a polynomial product $B(z)C(z)$ is fitted onto a given polynomial $A(z)$. Polynomial products are formed by convolution of the underlying coefficient sequences. Hence error equation (2.8) can be posed in terms of the following convolutional model fitting problem:

$$\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \\ e_n \end{bmatrix} = \begin{bmatrix} a_1 - c \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} - \begin{bmatrix} 1 & & & \\ c & 1 & & \\ & c & \cdot & \\ & & \cdot & \cdot \\ & & & \cdot\ 1 \\ & & & c \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix} \quad , \tag{2.9}$$

where the $\{e_k,\ k = 1, 2, \ldots, n\}$ are the coefficients of error polynomial $E(z)$. Convolutional models have proven useful in special areas of polynomial root finding [12], [13]. Let us rewrite (2.9) more compactly as follows:

$$\mathbf{e} = \mathbf{a}_c - \mathbf{C}\mathbf{b} \quad . \tag{2.10}$$

Notice from (2.9) that $\mathbf{C}$ is a $n \times n - 1$ bidiagonal matrix with a subdiagonal constituted by parameter c.

In each of the n cases, we assume that c is initialized with an initial coarse root estimate according to (2.7). The optimization procedure is computed for each of the given $\{\hat{z}_k, k = 1, 2, \ldots, n\}$ to obtain the refined set $\{\overline{z}_k, k = 1, 2, \ldots, n\}$. A first step towards the refined set is an optimization of the b-parameters in (2.9) in the least squares sense. For this purpose, (2.9) is premultiplied with a sequence of $n-1$ complex Givens plane rotations formally represented by an $n \times n$ unitary matrix $\mathbf{G}$ as follows:

$$\mathbf{G}\mathbf{e} = \mathbf{G}\mathbf{a}_c - \mathbf{G}\mathbf{C}\mathbf{b} \quad . \tag{2.11}$$

The $n-1$ complex plane rotors in $\mathbf{G}$ must be adjusted so that the c-subdiagonal in $\mathbf{C}$ is annihilated by unitary transformations. Consequently, we obtain:

$$\mathbf{G}\mathbf{C} = \begin{bmatrix} \mathbf{R} \\ 0 \ldots\ 0 \end{bmatrix} \quad , \tag{2.12}$$

where $\mathbf{R}$ is an upper-right bidiagonal triangular matrix. Moreover, define:

$$\mathbf{G}\mathbf{a}_c = \begin{bmatrix} \mathbf{d} \\ \delta \end{bmatrix} \quad . \tag{2.13}$$

Assuming that $\mathbf{C}$ admits a QR-factorization

$$\mathbf{C} = \mathbf{Q}\mathbf{R} \;\; ; \quad \mathbf{Q}^H\mathbf{Q} = \mathbf{I} \quad , \tag{2.14}$$

we can see that (2.12) implies the following structured representation of $\mathbf{G}$:

$$\mathbf{G} = \begin{bmatrix} \mathbf{Q}^H \\ \mathbf{q}^H \end{bmatrix} \quad , \tag{2.15}$$

where $\mathbf{q}$ spans the one-dimensional null space of $\mathbf{C}$. Consequently, (2.11) can be rewritten as follows:

$$\begin{bmatrix} \mathbf{Q}^H\mathbf{e} \\ \mathbf{q}^H\mathbf{e} \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ \delta \end{bmatrix} - \begin{bmatrix} \mathbf{R}\mathbf{b} \\ 0 \; \ldots \; 0 \end{bmatrix} \quad . \tag{2.16}$$

Determining $\mathbf{b}$ so that

$$\mathbf{R}\mathbf{b} = \mathbf{d} \quad , \tag{2.17}$$

implies that $\mathbf{Q}^H\mathbf{e} = [0 \cdots 0]^H$, and therefore $\mathbf{b}$ constitutes a least squares solution of overdetermined problem (2.10), as desired.

Finally, it remains to determine the elementary complex Givens plane rotors of $\mathbf{G}$ so that the bidiagonal matrix $\mathbf{C}$ is transformed into $\mathbf{R}$ according to (2.12) using elementary complex Givens plane rotations $\mathbf{\Gamma}$. Each of these elementary transformations must be adjusted so that the following complex nulling problem is solved:

$$\begin{bmatrix} r_1 \\ 0 \end{bmatrix} = \mathbf{\Gamma} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \quad , \tag{2.18}$$

where $\mathbf{\Gamma}$ is the 2×2 complex Givens plane rotor:

$$\mathbf{\Gamma} = \begin{bmatrix} \alpha & \beta^* \\ -\beta & \alpha \end{bmatrix} \;\; ; \quad \mathbf{\Gamma}^H\mathbf{\Gamma} = \mathbf{I} \quad , \tag{2.19}$$

$$\rho = \sqrt{|c_1|^2 + |c_2|^2} \quad , \tag{2.20}$$

$$\alpha = \frac{|c_1|}{\rho} \quad , \tag{2.21}$$

$$\beta = \alpha \frac{c_2}{c_1} \quad . \tag{2.22}$$

We are working downwards sucessively from top-left to bottom-right along the main diagonal and subdiagonal of $\mathbf{C}$ in (2.12) with c_1 denoting a main diagonal element and c_2 denoting the corresponding subdiagonal element, respectively. Then the r_1 of (18) is the resulting main diagonal element of $\mathbf{R}$ in (2.12).

Once we have computed the vector $\mathbf{b}$, we can proceed with a computation of the desired update Δc required in refinement step (2.6). For this purpose, consider the following linear Taylor series expansion of the fitting error $\mathbf{e}$ of (2.9) or (2.10),

$$\overline{\mathbf{e}}(\mathbf{p}) = \mathbf{e}(\mathbf{p}_0) + \mathbf{F}(\mathbf{p}_0)(\mathbf{p} - \mathbf{p}_0) \quad , \tag{2.23}$$

where $\mathbf{p}_0$ is an initial parameter vector comprising both the least squares estimated $\mathbf{b}$ of (2.17) as well as the c obtained via initialization according to (2.7):

$$\mathbf{p}_0 = \begin{bmatrix} \mathbf{b} \\ c \end{bmatrix} \quad , \tag{2.24}$$

and $\mathbf{F}(\mathbf{p}_0)$ denotes the Jacobian of the fitting error $\mathbf{e}$. We can easily verify from (2.9), (2.10) that:

$$\begin{aligned} \mathbf{F}(\mathbf{p}_0) &= \left[\frac{\partial \mathbf{e}}{\partial b_1} \; \frac{\partial \mathbf{e}}{\partial b_2} \cdots \frac{\partial \mathbf{e}}{\partial b_{n-1}} \,\middle|\, \frac{\partial \mathbf{e}}{\partial c} \right] \\ &= - \begin{bmatrix} 1 & & & & 1 \\ c & 1 & & & b_1 \\ & c & \cdot & & b_2 \\ & & \cdot\;\cdot & & \cdot \\ & & & \cdot\; 1 & \cdot \\ & & & c & b_{n-1} \end{bmatrix} \quad . \end{aligned} \tag{2.25}$$

A comparison of (2.25) with (2.9) and (2.12) reveals that a transformation of $\mathbf{F}(\mathbf{p}_0)$ with exactly the same sequence of elementary complex Givens plane rotations $\mathbf{G}$ as in (2.11) will transform $\mathbf{F}(\mathbf{p}_0)$ into an upper-right triangular matrix of the following form:

$$\mathbf{G}\mathbf{F}(\mathbf{p}_0) = - \begin{bmatrix} \mathbf{R} & \mathbf{g} \\ 0 \cdots 0 & \gamma \end{bmatrix} \quad . \tag{2.26}$$

Denoting the desired updating vector as:

$$\Delta\mathbf{p} = \begin{bmatrix} \Delta\mathbf{b} \\ \Delta c \end{bmatrix} = \mathbf{p} - \mathbf{p}_0 \quad , \tag{2.27}$$

we can see that the linearized error equation (2.23) can be rewritten as follows:

$$\overline{\mathbf{e}}(\mathbf{p}) = \mathbf{e}(\mathbf{p}_0) + \mathbf{F}(\mathbf{p}_0)\Delta\mathbf{p} \quad . \tag{2.28}$$

By setting

$$\overline{\mathbf{e}}(\mathbf{p}) = [0 \cdots 0]^H \quad , \tag{2.29}$$

we obtain an $n \times n$ system of linear equations for the vector of updates $\Delta\mathbf{p}$:

$$-\mathbf{F}(\mathbf{p}_0)\Delta\mathbf{p} = \mathbf{e}(\mathbf{p}_0) \quad , \tag{2.30}$$

where $\mathbf{e}(\mathbf{p}_0) = \mathbf{e}$ is the error vector of (2.10) computed using LS-parameter vector $\mathbf{b}$ of (2.17) and the initial c of (2.7). To solve (2.30), recall from (2.26) and (2.16) that this system appears conveniently transformed into upper-right banded triangular form as follows:

$$\begin{bmatrix} \mathbf{R} & \mathbf{g} \\ 0 \cdots 0 & \gamma \end{bmatrix} \begin{bmatrix} \Delta\mathbf{b} \\ \Delta c \end{bmatrix} = \mathbf{G}\mathbf{e} = \begin{bmatrix} \mathbf{Q}^H\mathbf{e} \\ \mathbf{q}^H\mathbf{e} \end{bmatrix} = \begin{bmatrix} \mathbf{d} - \mathbf{R}\mathbf{b} \\ \delta \end{bmatrix} \quad . \tag{2.31}$$

Since we are performing only a single iteration, we are not interested in computing $\Delta\mathbf{b}$ explicitly. Only the updating parameter Δc is computed as follows:

$$\Delta c = \frac{\delta}{\gamma} \quad , \tag{2.32}$$

where

$$\delta = \mathbf{q}^H \mathbf{e} \quad , \tag{2.33}$$

$$\gamma = \mathbf{q}^H \begin{bmatrix} 1 \\ \mathbf{b} \end{bmatrix} \quad . \tag{2.34}$$

However, $\mathbf{q}$ is never formed explicitly. We obtain δ as the bottom component of filtered sequence $\mathbf{e}$ and obtain γ as the bottom component of filtered sequence $\left[1,\ \mathbf{b}^H\right]^H$, where the $n-1$ elementary complex Givens rotors of $\mathbf{G}$ constitute an orthogonal filter [18] acting on these sequences. More explicitly, a filtering of the $\mathbf{e}$-sequence comprises the following transformations:

$$\begin{bmatrix} e_1' \\ e_2' \end{bmatrix} = \begin{bmatrix} \alpha_1 & \beta_1^* \\ -\beta_1 & \alpha_1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \quad , \tag{2.35}$$

$$\begin{bmatrix} e_k'' \\ e_{k+1}' \end{bmatrix} = \begin{bmatrix} \alpha_k & \beta_k^* \\ -\beta_k & \alpha_k \end{bmatrix} \begin{bmatrix} e_k' \\ e_{k+1} \end{bmatrix} \ ; \ \ k = 2, 3, \ldots, n-1 \ \ , \tag{2.36}$$

or compactly:

$$e_1' = e_1 \quad , \tag{2.37}$$

$$e_k' = -\beta_{k-1} e_{k-1}' + \alpha_{k-1} e_k \ ; \ \ k = 2, 3, \ldots, n \ \ . \tag{2.38}$$

Finally, the δ of (2.33) is obtained as the bottom component of the filtered $\mathbf{e}$-sequence as follows:

$$\delta = e_n' \quad . \tag{2.39}$$

In the same fashion, an evaluation of (2.34) yields:

$$b_1' = -\beta_1 + \alpha_1 b_1 \quad , \tag{2.40}$$

$$b_k' = -\beta_k b_{k-1}' + \alpha_k b_k \ ; \ \ k = 2, 3, \ldots, n-1 \ \ , \tag{2.41}$$

with

$$\gamma = b_{n-1}' \quad . \tag{2.42}$$

The root updating parameter Δc is then computed according to (2.32) and the root is finally updated according to (2.6). This completes the complex root refinement algorithm.

In summary, for each given initial root estimate $\hat{z}_k$, this refinement algorithm comprises the following steps:

- Initialize c according to (2.7).
- Compute the LS-solution vector $\mathbf{b}$ via (2.12), (2.13), and (2.17), using (2.18) - (2.22).
- Compute the error vector $\mathbf{e}$ from (2.9).
- Run the filters (2.37) - (2.39) and (2.40) - (2.42).
- Calculate Δc according to (2.32).
- Finally calculate the refined root $\overline{z}_k$ according to (2.6).

This refinement algorithm is available in terms of a Fortran subroutine `post_fit`, which is a part of an overall package `fit_polzeros` for high-precision complex polynomial root finding. This overall package also calls the original `POLZEROS` program [12] as an initial root estimator.

3. A Step to Real Coefficient Polynomials

The given `fit_polzeros` algorithm can also be applied to polynomials with real coefficients. In this case, the results do not automatically reflect the fact that complex roots must appear in complex conjugate pairs. Moreover, real root approximations have absolutely small but nonvanishing imaginary parts when a root-finder for complex polynomials is used.

These problems can be eliminated by means of complex conjugate root averaging. This method transforms approximate complex conjugate pairs into perfect complex conjugate pairs and removes small imaginary parts from real root estimates.

Suppose we have given a set of refined complex root estimates $\{\overline{z}_k\}$ from an algorithm like `fit_polzeros` at our disposition. A related set $\{\overline{z}_k^*\}$ is computed by complex conjugation. Next consider two elements z_1 and z_2 of set $\{\overline{z}_k\}$ and a related pair of elements of set $\{\overline{z}_k^*\}$:

$$z_1,\, z_2 \in \{\overline{z}_k\} \quad , \tag{3.1}$$

$$z_1^*,\, z_2^* \in \{\overline{z}_k^*\} \quad . \tag{3.2}$$

The following model assumptions will hold in cases where z_1 and z_2 represent a complex conjugate root pair:

$$z_1 = z_0 + \epsilon_1 \quad , \tag{3.3}$$

$$z_2 = z_0^* + \epsilon_2 \quad . \tag{3.4}$$

z_0 represents the true complex root and ϵ_1 and ϵ_2 are independent and generally different complex estimation errors. This results in the following undesirable property:

$$z_2 \neq z_1^* \quad . \tag{3.5}$$

We will now eliminate this annoying problem that often occurs with complex root estimators when applied to real coefficient polynomials. Recall the related roots of the complex conjugate set (3.2). According to model assumptions (3.3) and (3.4), these related complex conjugate roots must attain the following form:

$$z_1^* = z_0^* + \epsilon_1^* \quad , \tag{3.6}$$

$$z_2^* = z_0 + \epsilon_2^* \quad . \tag{3.7}$$

Now the set $\{\overline{z}_k^*\}$ is mapped into a sorted set $\{\overline{w}_k\}$ so that:

$$\sum_{k=1}^{n} |\overline{z}_k - \overline{w}_k| = min \quad , \tag{3.8}$$

and an averaged set $\{\hat{\overline{z}}_k\}$ is computed via:

$$\hat{\overline{z}}_k = \frac{1}{2}\left(\overline{z}_k + \overline{w}_k\right)\ ;\ \ k = 1, 2, \ldots, n \quad . \tag{3.9}$$

The shortest distance association sort of (3.8) will ensure that root estimates like (3.3), (3.4) with complex conjugates (3.6), (3.7) will be averaged as follows:

$$\hat{z}_1 = \frac{1}{2}\left(z_1 + z_2^*\right) = z_0 + \frac{1}{2}\left(\epsilon_1 + \epsilon_2^*\right) = z_0 + \hat{\epsilon}_0 \ , \tag{3.10}$$

$$\hat{z}_2 = \frac{1}{2}\left(z_2 + z_1^*\right) = z_0^* + \frac{1}{2}\left(\epsilon_1^* + \epsilon_2\right) = z_0^* + \hat{\epsilon}_0^* \ , \tag{3.11}$$

where

$$\hat{\epsilon}_0 = \frac{1}{2}\left(\epsilon_1 + \epsilon_2^*\right) \tag{3.12}$$

is an averaged estimation error. The averaged root estimates $\hat{z}_1$ and $\hat{z}_2$ will now be strictly related by complex conjugation, as desired. Moreover, operation (3.9) can be interpreted as a variant of phased averaging. The true information z_0 will add up “in voltage” while the statistically independent random complex estimation errors will add up “in power”. Hence another welcome side-effect of this operation is an improvement of the signal-to-noise ratio of the root estimates by a factor of 3 dB. In the case of real coefficient polynomials, we can apply the method to `fit_polzeros`, but also to the `POLZEROS` algorithm.

Finally, let us study the effect of this “complex phased averaging” operation (3.9) in the presence of real root estimates. For this purpose, suppose again that we pick out two samples z_1 and z_2 of set $\{\overline{z}_k\}$. In the case of two real roots, these samples satisfy the following model assumptions:

$$z_1 = r_1 + \epsilon_1 \quad , \tag{3.13}$$

$$z_2 = r_2 + \epsilon_2 \quad , \tag{3.14}$$

where r_1 and r_2 represent the ideal real roots, and the ϵ’s are statistically independent complex errors. Consequently, the related complex conjugate set elements will appear in the form:

$$z_1^* = r_1 + \epsilon_1^* \quad , \tag{3.15}$$

$$z_2^* = r_2 + \epsilon_2^* \quad . \tag{3.16}$$

As a consequence of the fact that complex conjugation does not alter a real number, it will turn out that complex phased averaging according to (3.9) will result in the following averaged real root estimates:

$$\hat{z}_1 = \frac{1}{2}\left(z_1 + z_1^*\right) = r_1 + \frac{1}{2}\left(\epsilon_1 + \epsilon_1^*\right) = r_1 + \hat{\epsilon}_1 \quad , \tag{3.17}$$

$$\hat{z}_2 = \frac{1}{2}\left(z_2 + z_2^*\right) = r_2 + \frac{1}{2}\left(\epsilon_2 + \epsilon_2^*\right) = r_2 + \hat{\epsilon}_2 \quad , \tag{3.18}$$

where:

$$\hat{\epsilon}_1 = \frac{1}{2}\left(\epsilon_1 + \epsilon_1^*\right) \quad , \tag{3.19}$$

$$\hat{\epsilon}_2 = \frac{1}{2}\left(\epsilon_2 + \epsilon_2^*\right) \quad , \tag{3.20}$$

are real errors, because the annoying complex error components cancel out perfectly, as desired.

4. Examples

In this section, we demonstrate the effectiveness of `fit_polzeros` for two important classes of polynomials, namely high-degree random coefficient polynomials and high-degree low coefficient variance deterministic polynomials like, for instance, the z-transforms of complex chirp sequences and the z-transfer functions of linear phase finite impulse response (FIR) filters.

As a performance criterion we use the *absolute root error* of root number k denoted by

$$E_k = |z_k - \overline{z}_k| \quad , \tag{4.1}$$

where z_k is a root obtained from a quadruple precision reference algorithm and $\overline{z}_k$ denotes a root estimate, obtained either from `fit_polzeros` or from `POLZEROS`. E_k is computed for every estimated root, even if tens of thousands of roots are evaluated. These absolute root errors are either concatenated and displayed like a time series, or they are statistically evaluated in terms of the probability that an E_k exceeds a reference error. These "exceeding error" curves can be interpreted as the *Receiver Operating Characteristics* (ROC) [19] of a root estimator as they display the probability that an absolute root error exceeds a reference error level which plays the role of a discriminator threshold. These "exceeding error" or ROC curves are most instructive in the evaluation of root estimation errors.

4.1 Complex Random Coefficient Polynomials. In the first series of experiments, the coefficients are generated as a complex nearly Gaussian distributed zero-mean white noise random processes. Polynomials of this kind are encountered in several important application areas, such as geophysical exploration, underwater acoustics, radar signal processing, and time series analysis in general. As a consequence of the limited variance of their coefficients, polynomials of this class can be rooted up to virtually almost unlimited degrees. Moreover, we shall observe that the roots of these polynomials can be computed amazingly accurately if a proper algorithm is used, even in cases where it is practically impossible to operate with multiprecision algorithms like MPSolve [11], because of the excessive runtimes of multiprecision software.

Fig. 1 shows the root scatter plot of a typical $n = 1000$ complex random coefficent polynomial.

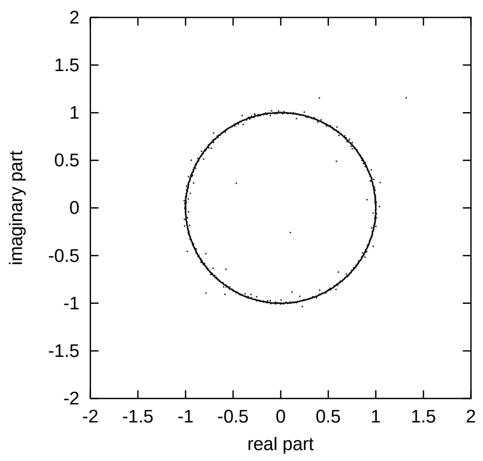

Figure 1: Typical root distribution in the complex random coefficients scenario of degree $n = 1000$.

We can see that most of the roots are grouped very tightly around the unit circle in the z-plane in cases of high-degree polynomials with low coefficient variance. However, some roots may lie far from the unit circle to accomodate the arithmetic and geometric means of all roots, as constituted by the first and the last elements of the coefficient sequence.

In the first experiment, we generate 10 statistically independent complex random coefficient polynomials of degree $n = 1000$ and estimate their roots using the `POLZEROS` algorithm on the one hand, and the `fit_polzeros` algorithm on the other hand. In both cases, we compute the absolute root errors according to (4.1) and concatenate all these root errors of the 10 experiments to a virtual "time series" of 10000 absolute error samples. Fig. 2 shows these absolute root error sequences for the two algorithms under study. Throughout this paper, results obtained from the `POLZEROS` program are labeled by "ABE" while results from the new program `fit_polzeros` are labeled by "FIT".

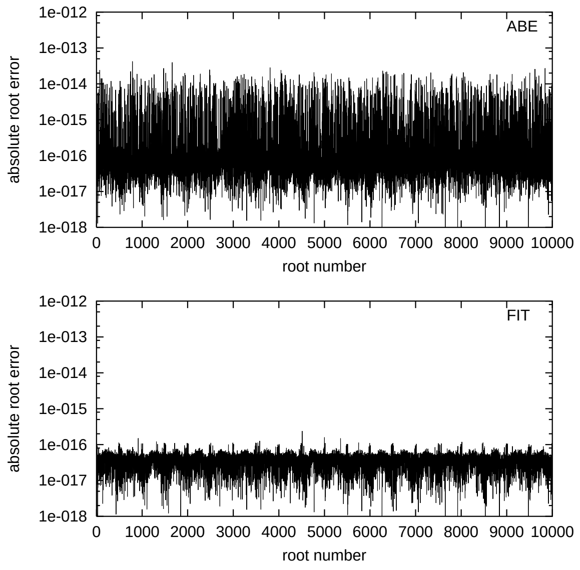

Figure 2: Absolute root errors of 10 trial runs of random coefficients scenario $n = 1000$. Results of POLZEROS labeled by ABE. Results of fit_polzeros labeled by FIT.

It is apparent from the absolute error sequences displayed in Fig. 2 that `POLZEROS` suffers from a relatively high variance of the root errors. However, in our tests all these root estimates are well located inside the inner basin of attraction of the post-fitting refinement algorithm described in Section 2 of this paper. This post-fitting algorithm is based on a linear Taylor series expansion of a fitting error. Consequently, its convergence in the inner basin of attraction is locally quadratic. When initialized with a root estimate from the `POLZEROS` algorithm, this post-fitter will approximately *double* the number of accurate digits in a single iteration in our tests with random coefficient polynomials. If a doubling of the number of accurate digits is prevented by the given numerical resolution, the algorithm produces root estimates with an E_k in the range of the machine precision. In the case of standard double precision arithmetic, we can expect values of $E_k \approx 1e - 16$ or lower. We can easily verify from Fig. 2, that this is well satisfied here. Moreover, this property is almost independent of the degree n of the polynomials under study. So this kind of post-fitting realizes almost perfectly an error bound for the class of complex random coefficient polynomials and similarly for the two other classes of tested polynomials.

This is also confirmed by the corresponding exceeding error curves for this experiment, as displayed in Fig. 3. The curves in Fig. 3 show the probability that a root error exceeds a reference error level. For instance, the solid line curve in Fig. 3 confims that after post-fitting, the probability that an E_k exceeds a value of 1e-16 vanishes almost perfectly. On the other hand, the dashed line curve in Fig. 3 shows the typical error characteristics of the plain `POLZEROS` algorithm for this class of polynomials: The errors spread out to unnecessarily large values, with a ramp-like tail characteristics of the exceeding error curve which is very characteristic for this estimator.

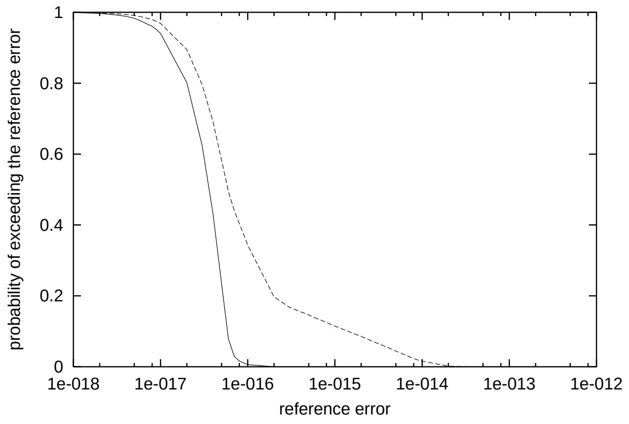

Figure 3: Probability that an absolute root error exceeds a reference error level in the random coefficients scenario $n = 1000$ of Fig. 2. *Dashed line*: POLZEROS. *Solid line*: fit_polzeros.

We can proceed with running some more experiments of this kind by successively increasing the polynomial degree n. The following cases have been studied:

- 32 statistically independent polynomials of degree $n = 2000$
- 16 statistically independent polynomials of degree $n = 4000$
- 8 statistically independent polynomials of degree $n = 8000$
- 4 statistically independent polynomials of degree $n = 16000$
- 2 statistically independent polynomials of degree $n = 32000$
- 1 polynomial of degree $n = 64000$

In each of these cases, we obtain concatenated error sequences of 64000 samples each, amenable to a statistical evaluation. Fig. 4 shows the exceeding error curves for each of these 6 cases, individually displayed for the `POLZEROS` and the `fit_polzeros` algorithms, where the 6 curves for `POLZEROS` appear as dashed lines and the 6 curves for `fit_polzeros` appear as solid lines in Fig. 4.

The most instructive result of this experiment is that the 6 individual curves of the exceeding errors of `fit_polzeros` for polynomials of increasing degrees are almost identical and match almost perfectly. This shows that the estimation accuracy of this algorithm is largely *independent* of the polynomial degree for this class of complex random coefficient polynomials and that the errors appear almost perfectly bounded by the machine precision level of $\approx 1e{-}16$. This property will not hold for every algorithm. For instance, we can see here from the dashed curves of the

POLZEROS exceeding errors, that these curves again show the typical ramp-like tail characteristics, and additionally, grow successively to larger average errors with increasing degree n.

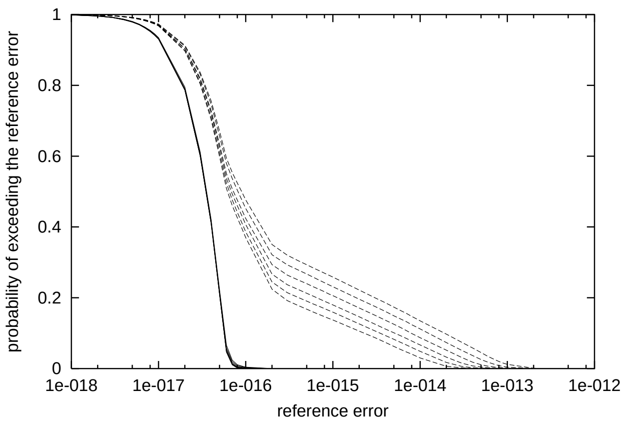

Figure 4: Probability that an absolute root error exceeds a reference error level in random coefficient scenarios of growing degree. Cases $32 \times n = 2000$, $16 \times n = 4000$, $8 \times n = 8000$, $4 \times n = 16000$, $2 \times n = 32000$, and $1 \times n = 64000$ examined. *Dashed lines*: POLZEROS. *Solid lines*: fit_polzeros.

Fig. 5 shows the underlying E_k sequences for the two algorithms `POLZEROS` and `fit_polzeros` in the case of the $n = 64000$ polynomial.

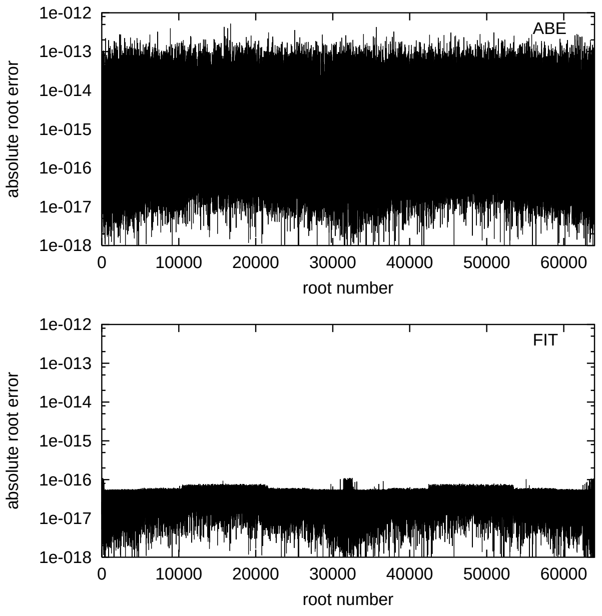

Figure 5: Absolute root errors of one trial run of random coefficient scenario $n = 64000$. Results of POLZEROS labeled by ABE. Results of fit_polzeros labeled by FIT.

We can see that the largest errors of `POLZEROS` in this case are in the range of 1e-13, while the errors of `fit_polzeros` appear almost perfectly upper bounded by a value of 1e-16. Hence post-fitting decreases the estimation errors at least by a factor of 1000 in the best cases in this example of a high-degree complex random coefficient polynomial.

4.2 Complex Linear Chirp Signal. Fig. 6 shows the real and imaginary parts of a complex linear chirp signal of duration 2001 samples with partial Hanning nose/tail taper and a normalized angular frequency ranging from $\overline{\omega} = 0.08\pi$ up to $\overline{\omega} = 0.48\pi$. We are interested in computing all roots of the z-transform of this complex chirp, giving rise to a complex root finding problem of degree $n = 2000$.

Fig. 7 displays the roots of the z-transform of the given complex linear chirp signal computed by `fit_polzeros`. We can see that in this case, the roots appear again very tightly lined up around the unit circle in the z-plane.

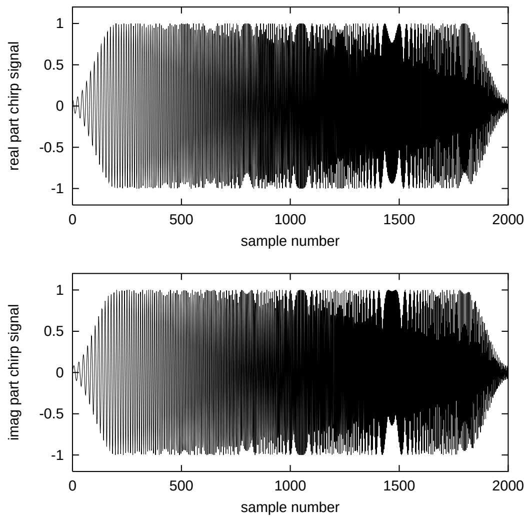

Figure 6: Real and imaginary parts of a complex chirp signal of duration 2001 samples.

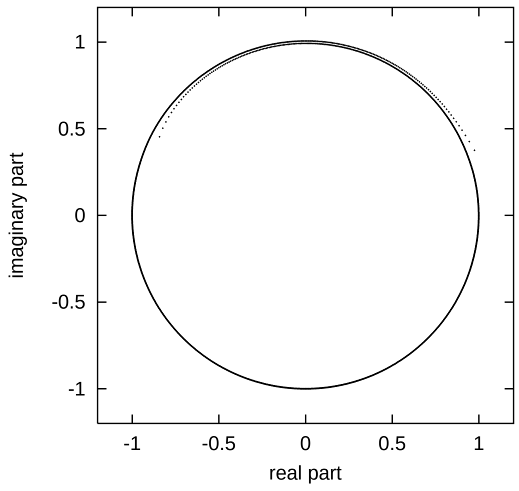

Figure 7: Root distribution of the $n = 2000$ complex linear chirp z-transform polynomial.

Fig. 8 shows the corresponding absolute root error sequences for the two algorithms `POLZEROS` and `fit_polzeros`.

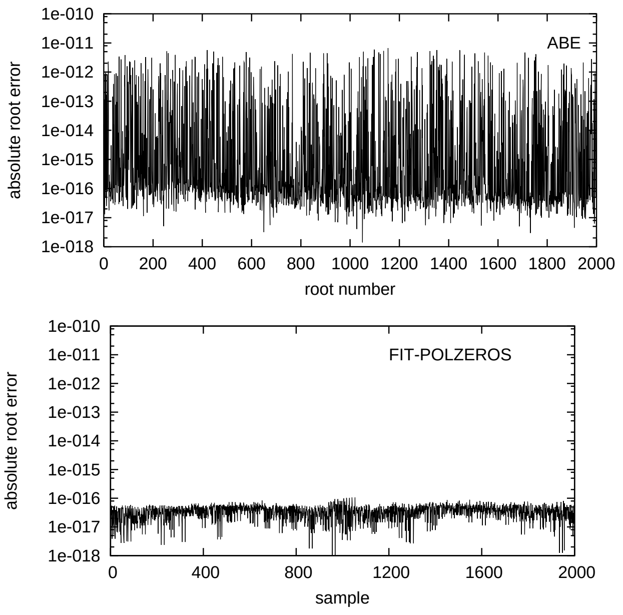

Figure 8: Absolute root errors of an $n = 2000$ complex linear chirp z-transform polynomial. Results of POLZEROS labeled by ABE. Results of fit_polzeros labeled by FIT.

We can see that `fit_polzeros` will upper bound the absolute roots errors by $\approx 1e - 16$ again, while the errors of `POLZEROS` are excessively high, as expected. This observation is also confirmed by the corresponding exceeding error curves as displayed in Fig. 9.

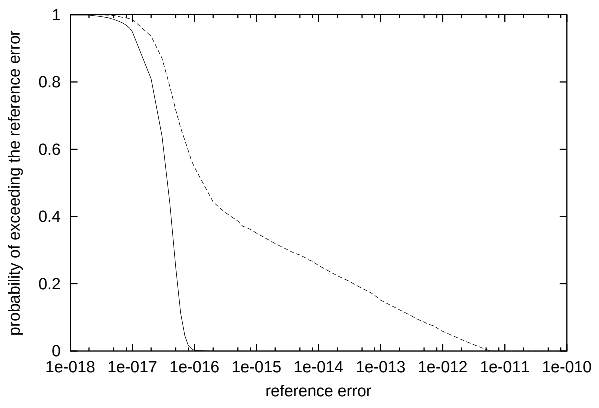

Figure 9: Probability that an absolute root error exceeds a reference error level in the complex chirp scenario $n = 2000$. *Dashed line*: POLZEROS. *Solid line*: fit_polzeros.

A comparison of the exceeding error characteristics of `POLZEROS` in this case of the complex chirp (Fig. 9) with the exceeding error curves of `POLZEROS` in the case of the random coefficient polynomial as shown in Fig. 4 reveals that we must accept an additional substantial increase of error level in the case of POLZEROS, if a chirp polynomial instead of a random coefficient polynomial is used. The estimation accuracy of `fit_polzeros`, on the other hand, remains totally unaltered by this change from a random coefficient to a deterministic chirp coefficient sequence. In summary, we conclude that in this example, a post-fitting refinement will improve the root estimates by a factor of more than 10000 in the best cases.

4.3 Linear FIR Filter z-Transfer Functions. We are now moving to real coefficient polynomials. Practically interesting cases here are impulse responses of high-degree linear phase FIR filters and the roots of their z-transforms [20]. Fig. 10 shows the impulse response of a linear phase FIR lowpass filter of duration 101 samples.

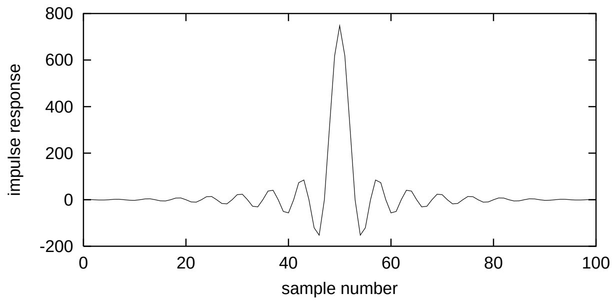

Figure 10: Impulse response of an $n = 100$ linear phase FIR lowpass filter.

The computation of the roots of the corresponding z-transfer function of this filter id an $n = 100$ real polynomial root finding problem. In all cases of this kind with *real* coefficient polynomials, we apply complex phased averaging, as described in Section 3, to both the POLZEROS and `fit_polzeros` estimation results. The so extended programs are consequently renamed `r_POLZEROS` and `r_fit_polzeros`, respectively.

Fig. 11 shows the corresponding root scatter plot of the z-transfer function of this linear phase FIR lowpass filter of order $n = 100$, computed using `r_fit_polzeros`. We can see that this root scatter plot shows the typical tube-like characteristics of the root locations in the passband of the filter.

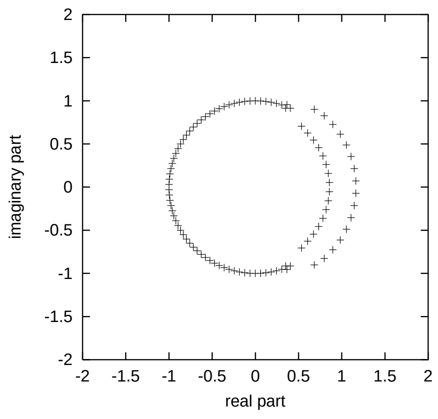

Figure 11: Roots of the z-transform of the $n = 100$ linear phase FIR lowpass filter impulse response.

Fig 12 shows the corresponding sequences of absolute root errors, as obtained using `r_POLZEROS` on the one hand, and `r_fit_polzeros` on the other hand as root estimators.

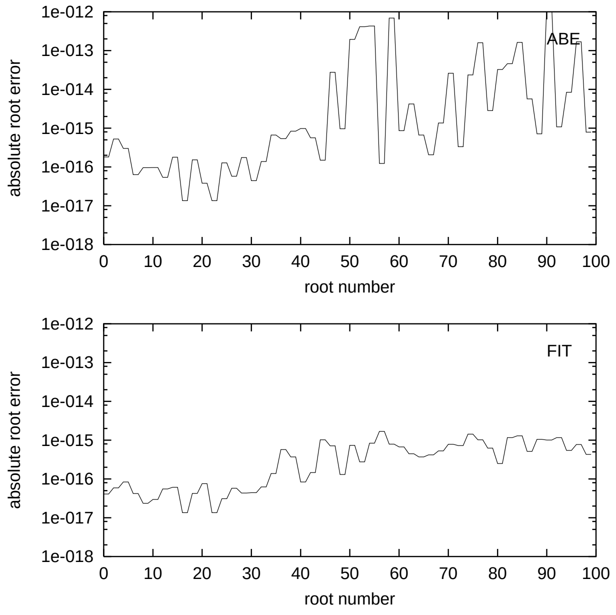

Figure 12: Absolute root errors in the $n = 100$ linear phase FIR lowpass filter scenario. Results of r_POLZEROS labeled by ABE. Results of r_fit_polzeros labeled by FIT.

We can see that these impulse response sequences no longer share the bounded variance characteristics of the coefficients as well as the former random and chirp sequences. Hence, as a consequence of higher coefficient variances, the root estimation errors will grow to larger values. The absolute root estimation errors grow to maximum values of $\approx 1e - 15$ in the case of `r_fit_polzeros`, and even to maximum values of $\approx 1e - 12$ in the case of `r_POLZEROS`. However, a substantial reduction of the estimation errors is again achieved by using the post-fitting refinement concept.

A second experiment of this kind with a linear phase FIR lowpass of much higher degree $n = 1000$ is finally examined. Fig. 13 shows the corresponding impulse response of duration 1001 samples.

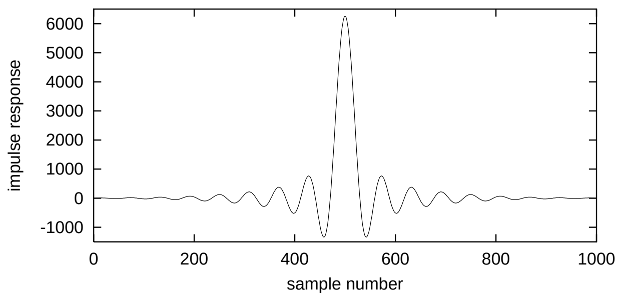

Figure 13: Impulse response of an $n = 1000$ linear phase FIR lowpass filter.

Fig. 14 shows the roots of the corresponding z-transfer function of this filter, computed using `r_fit_polzeros`.

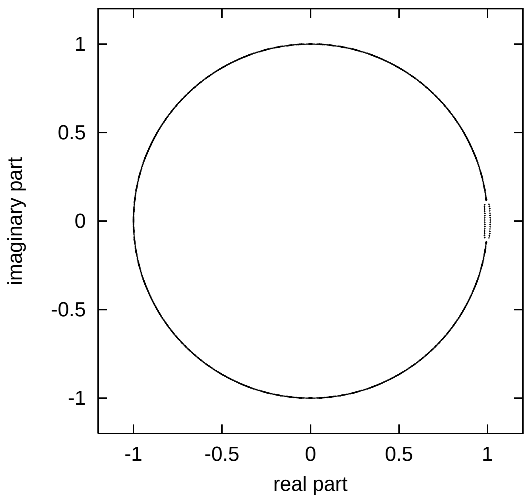

Figure 14: Roots of the z-transform of the $n = 1000$ linear phase FIR lowpass filter impulse response.

A comparison with the root scatter plot of the $n = 100$ filter of Fig. 11 reveals that this higher degree filter has a much lower relative bandwidth, characterized by a short and narrow tube of passband roots around $z = 1$ in the z-plane. The stopband roots are all perfectly lined up on the unit circle in the z-plane of Fig. 14.

Fig. 15 shows the corresponding absolute root error sequences. These sequences confirm an impressive improvement of estimation accuracy by a factor of 10000 approximately for `r_fit_polzeros` over the plain `r_POLZEROS` algorithm.

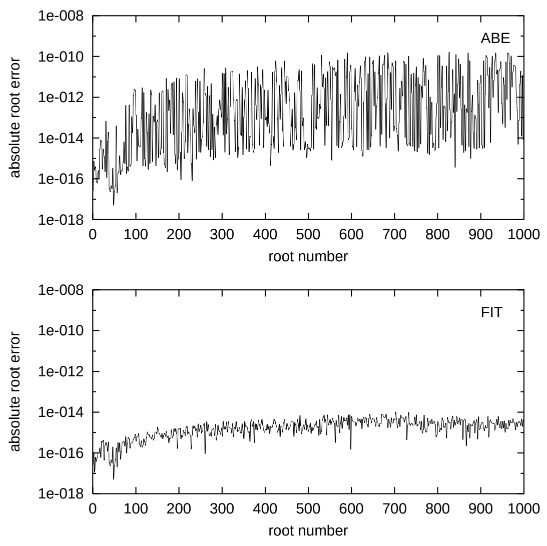

Figure 15: Absolute root errors in the $n = 1000$ linear phase FIR lowpass filter scenario. Results of r_POLZEROS labeled by ABE. Results of r_fit_polzeros labeled by FIT.

4.4 Runtimes. A final concern are the runtimes of the two algorithms under comparison. As a rule of thumb, we found that the runtime of `fit_polzeros` is approximately by a factor of 2.0 - 2.5 higher than the runtime of the conventional `POLZEROS` algorithm. Table 1 shows the practical runtimes for the random coefficients scenario of growing degrees. In the other cases, we observed very similar runtimes. This indicates that the overall runtime of the algorithms will not depend significantly on the characteristics the particular coefficient set as long as we stay within this class of high-degree polynomials with relatively limited coefficient variations, as typical in application areas like signal processing.

Table 1: Runtimes (in seconds) of the plain POLZEROS (ABE) and fit_polzeros (FIT) algorithms for rooting of one complex random coefficient polynomial of degree n.

degree n	1000	2000	4000	8000	16000	32000	64000
ABE	0.123	0.483	1.89	7.34	29.36	117.6	474
FIT	0.300	1.136	4.50	13.89	62.26	256.7	1219
factor	2.44	2.35	2.37	1.89	2.12	2.18	2.57

5. Conclusions

We introduced the principle of complex polynomial fitting as a useful tool for a post-refinement of complex root estimates. The method requires coarse root estimates from a basic root finding algorithm as inputs. We found that `POLZEROS` is well suited as initial estimator, because the algorithm is fast, reliable and sufficiently accurate for the initialization of the root estimates inside the inner basin of attraction of the post-fitting refinement algorithm. Experiments have revealed that the post-fitting refinement step significantly improves the estimation accuracy in many cases of practical interest. The deeper reasons for the demonstrated performance of `fit_polzeros` are twofold: On the one hand, both the error equation (2.9) as well as the Jacobian (2.25) are only linear functions of the underlying coefficients. This results in a relatively large inner basin of attraction with the consequence that the algorithm adapts well even in cases of relatively inaccurate initial root estimates. On the other hand, the δ and γ parameters used for computing the update Δc according to (2.32) are both computed by orthogonal filters (2.37)-(2.39) and (2.40)-(2.42), and *not* by the transversal filters (2.33) and (2.34). This seemingly minor implementation detail is in reality the key to the demonstrated numerical accuracy of `fit_polzeros`, particularly in cases of excessively high polynomial degrees n.

References

1. *Wilkinson J. H.,* 'The evaluation of the zeros of ill-conditioned polynomials Part I', *Numerische Mathematik,* ***1*** *(1959), 150-166.*
2. *Steiglitz K. and Dickinson B.,* 'Phase unwrapping by factorization', *IEEE Trans. on Acoustics, Speech, and Signal Processing,* ***30*** *(1982), 984-991.*
3. *Weiss A.J., Willsky A.S., and Levy B.C.,* 'Nonuniform array processing via the polynomial approach', *IEEE Trans. Aerosp. Electron. Syst.,* ***25*** *(1989), 48-55.*
4. *Ahlen A. and Sternad M.,* 'Optimal deconvolution based on polynomial methods', *IEEE Trans. on Acoustics, Speech, and Signal Processing,* ***37*** *(1989), 217-226.*
5. *Tabrikian J. and Messer H.,* 'Source localization in a waveguide using polynomial rooting', *IEEE Trans. on Signal Processing,* ***44*** *(1996), 1861-1871.*

6. *Dowlut N. and Mankias A.,* 'A polynomial rooting approach to super-resolution array design', *IEEE Trans. on Signal Processing,* ***48*** *(2000), 1559-1569.*
7. *Charge P., Wang Y., and Saillard J.,* 'A non-circular sources direction finding method using polynomial rooting', *Signal Processing,* ***81*** *(2001), 1765-1770.*
8. *Weiss A.J. and Friedlander B.,* 'Range and bearing estimation using polynomial rooting', *IEEE J. Ocean Engr.,* ***18*** *(2002), 130-137.*
9. *Sturmel N., Alessandro C., and Doval B.,* 'A comparative evaluation of the zeros of z-transform representation for voice source estimation', *in Proc. INTERSPEECH, (2007), 558-559.*
10. *Jensen J.R., Christensen M.G., and Jensen S.H.,* 'Fundamental frequency estimation using polynomial rooting of a subspace-based method', *in Proc. EUSIPCO, (2010), 502-506.*
11. *Bini D.A. and Fiorentino G.,* 'MPSolve homepage', *http://www.dm.unipi.it/clusterpages/mpsolve/index.htm.*
12. *Bini D.A.,* 'Numerical computation of polynomial zeros by means of Aberth's method', *Numerical Algorithms,* ***13*** *(1996), 179-200.*
13. *Aberth O.,* 'Iteration methods for finding all zeros of a polynomial simultaneously', *Mathematics of Computation,* ***27*** *(1973), 339-344.*
14. *Ehrlich L.W.,* 'A modified Newton method for polynomials', *Communications of the ACM,* ***10*** *(1967), 107-108.*
15. *Strobach P.,* 'The fast quartic solver', *Journal of Computational and Applied Mathematics,* ***234*** *(2010), 3007-3024.*
16. *Strobach P.,* 'Solving Cubics by polynomial fitting', *Journal of Computational and Applied Mathematics,* ***235*** *(2011), 3033-3052.*
17. *Strobach P.,* 'A fitting algorithm for real coefficient polynomial rooting', *Journal of Computational and Applied Mathematics,* ***236*** *(2012), 3238-3255.*
18. *Dewilde P.,* 'Orthogonal filters: A numerical approach to filtering theory', *In P. Fuhrmann (ed.), Lecture Notes in Control and Information Sciences, Springer-Verlag, (1984), 253-267.*
19. *VanTrees H. L.,* Detection, Estimation, and Modulation Theory: Part 1, *John Wiley & Sons, New York, 2001.*
20. *Rabiner L. and Gold B.,* Theory and Application of Digital Signal Processing, *Prentice Hall Inc., Englewood Cliffs, N. J., 1975.*