Effective fast algorithms for polynomial spectral factorization

D. A. Bini*, G. Fiorentino*, L. Gemignani* and B. Meini*

November 30, 2001

Abstract

Let $p(z)$ be a polynomial of degree n having zeros $|\xi_1| \le \ldots \le |\xi_m| < 1 < |\xi_{m+1}| \le \ldots \le |\xi_n|$. This paper is concerned with the problem of efficiently computing the coefficients of the factors $u(z) = \prod_{i=1}^{m}(z-\xi_i)$ and $l(z) = \prod_{i=m+1}^{n}(z-\xi_i)$ of $p(z)$ such that $a(z) = z^{-m}p(z) = (z^{-m}u(z))l(z)$ is the spectral factorization of $a(z)$. To perform this task the following two-stage approach is considered: First we approximate the central coefficients $x_{-n+1}, \ldots x_{n-1}$ of the Laurent series $x(z) = \sum_{i=-\infty}^{+\infty} x_i z^i$ satisfying $x(z)a(z) = 1$; then we determine the entries in the first column and in the first row of the inverse of the Toeplitz matrix $T = (x_{i-j})_{i,j=-n+1,n-1}$ which provide the sought coefficients of $u(z)$ and $l(z)$. Two different algorithms are analyzed for the reciprocation of Laurent polynomials. One algorithm makes use of Graeffe's iteration which is quadratically convergent. Differently, the second algorithm directly employs evaluation/interpolation techniques at the roots of 1 and it is linearly convergent only. Algorithmic issues and numerical experiments are discussed.

Keywords: Spectral factorization, polynomial factorization, Graeffe method, Laurent polynomial inversion
AMS Subject Classification: 65H05, 12Y05, 12D05

1 Introduction

Let $p(z)$ be a polynomial of degree n with complex coefficients,

$$\begin{array}{l} p(z) = \sum_{i=0}^{n} p_i z^i = p_n \prod_{i=1}^{n}(z-\xi_i), \\ |\xi_1| \le \cdots \le |\xi_m| < 1 < |\xi_{m+1}| \le \cdots \le |\xi_n| \end{array} \tag{1}$$

and define its factors

$$u(z) = \prod_{i=1}^{m}(z-\xi_i) = \sum_{i=0}^{m} u_{m-i} z^i, \quad l(z) = \prod_{i=m+1}^{n}(z-\xi_i) = \sum_{i=0}^{n-m} l_i z^i \tag{2}$$

such that $p(z) = p_n l(z) u(z)$, and both $l(z)$ and $u_R(z) = z^m u(z^{-1}) = \sum_{i=0}^{m} u_i z^i$ have zeros of modulus greater than 1. Under these assumptions, the factorization of $p(z)$ provides the spectral factorization $a(z) = z^{-m}p(z) = p_n u_R(z^{-1}) l(z)$

*Dipartimento di Matematica Università di Pisa, via Buonarroti 2, 56127 Pisa (bini@dm.unipi.it, fiorent@dm.unipi.it, gemignan@dm.unipi.it, meini@dm.unipi.it)

of the Laurent polynomial $a(z) = \sum_{i=-m}^{n-m} a_i z^i$ with $a_i = p_{i+m}$, $-m \le i \le n-m$. Let us define $\rho = \max\{|\xi_m|, |1/\xi_{m+1}|\}$ the *isolation index* of the two factors.

Approximating the coefficients of the factors $l(z)$ and $u(z)$ given the coefficients of $p(z)$ or, equivalently, of $a(z)$ is a problem which has both a theoretical and a practical interest. Several authors have investigated the asymptotic complexity of this computation [18, 17, 12, 14], more numerical issues have been considered in [7, 13], a detailed comparison of more practical algorithms has been performed in [20] for the specific case where $a(z)$ is a real symmetric Laurent polynomial. Applications are shown in [20, 6, 13, 16].

More recently, the problem has been dealt in terms of Toeplitz matrix computations and manipulation of Laurent series [3, 4, 2] and several algorithms have been devised based on the following linearization of the problem [3, 2]:

Theorem 1 *Under the assumptions (1) there exists and is unique the Laurent series $x(z) = \sum_{i=-\infty}^{\infty} x_i z^i$ such that $x(z)a(z) = 1$ and $\sum_{i=-\infty}^{\infty} |x_i| < +\infty$. Moreover, if $T = (x_{i-j})_{i,j=1,q}$ is the $q \times q$ Toeplitz matrix associated with $x(z)$, where $q > \max\{m, n-m\}$ and $\boldsymbol{l} = (l_0, l_1, \ldots, l_{n-m}, 0, \ldots, 0)^T$, $\boldsymbol{u} = (u_0, u_1, \ldots, u_m, 0, \ldots, 0)^T$, are the coefficient vectors of the polynomials $u(z)$ and $l(z)$ padded with zeros to size q, then $T\boldsymbol{l} = (a_{n-m}u_0)^{-1}\boldsymbol{e}_1$, $\boldsymbol{u}^T T = (a_{n-m}l_0)^{-1}\boldsymbol{e}_1^T$.*

In this paper we deepen the analysis of methods for polynomial factorization based on Theorem 1 that is, relying on the reduction to inverting a Laurent polynomial and to solving two Toeplitz linear systems as shown in the following general scheme:

INPUT: The coefficients $a_{i-m} = p_i$ of $a(z) = \sum_{i=-m}^{n-m} a_i z^i$ and $\epsilon > 0$.

STAGE 1: Choose $q > \max\{m, n-m\}$ and approximate the coefficients $x_{-q}, \ldots, x_0, \ldots, x_q$ of the Laurent series $x(z)$ such that $a(z)x(z) = 1$ within the precision ϵ.

STAGE 2: Define $T = (x_{i-j})_{i,j=1,q}$, and solve the Toeplitz linear systems $T\boldsymbol{l} = \boldsymbol{e}_1$, $\boldsymbol{u}^T T = \boldsymbol{e}_1^T$.

OUTPUT: Approximations of the coefficients $u_0, \ldots, u_m, l_0, \ldots, l_{n-m}$ of $u(z)$ and $l(z)$.

Observe that stage 1 of the above scheme is equivalent to compute the entries in the first $2q+1$ diagonals of the inverse of the bi-infinite Toeplitz matrix associated with $a(z)$. The remaining stage 2 consists of solving a finite Toeplitz system and thus can be performed by means of the customary algorithms, like the fast, superfast, or iterative techniques (see [11]) at an arithmetic cost ranging from $O(n \log n)$ to $O(n^2)$ operations (ops).

The problem of performing stage 1 is investigated in detail in the next sections. More precisely, in section 3 we analyze an algorithm for stage 1, introduced in [2], based on the evaluation/interpolation technique at the roots of 1. We provide an adaptive criterion for dynamically choosing the number of Fourier points and for the control of the approximation error. In section 3 we revisit Graeffe's iteration and apply it to Laurent polynomials. In particular we define the concept of scaled Graeffe sequence and introduce the concept of balanced Graeffe sequence for its computation by improving and simplifying the algorithm of [3]. The error and the complexity analysis of our algorithm shows that an ϵ-approximation of $x_{-q}, \ldots, x_0, \ldots, x_q$ at stage 1 may be computed at the cost of $O(n \log n(\log\log \epsilon^{-1} - \log\log \rho^{-1}))$ floating-point operations performed with $O(n(|\log\log \rho^{-1}| + \log n))$ bits of precision provided that $\log \epsilon^{-1} = O(n \log n)$.

In section 4 we present the results of numerical experiments. From the tests that we have performed it results that the factorization algorithms are reliable and robust as long as a splitting annulus exists. The time complexity of performing stage 1 is quite negligible with respect to the time needed for solving two Toeplitz systems. This fact makes more important the design and the implementation of effective and reliable Toeplitz solvers. It is interesting to observe that, not only the algorithm based on Graeffe iteration is the one which reaches the best asymptotic complexity bound up to date, but even in the practical experiments it is very effective and it allows to deal with polynomials of degree up to 10^6 in a very low CPU time. Despite its theoretical low convergence the algorithm based on evaluation/interpolation is practically effective for well conditioned factorizations and in this case provides the lowest CPU time. Some issues related to the numerical stability are also discussed.

2 Inverting a polynomial by means of FFT.

Given a Laurent polynomial $a(z) = z^{-m}p(z) = \sum_{i=-m}^{n-m} a_i z^i$ whose zeros ξ_i satisfy condition (1), in order to accomplish stage 1 and 2 of the general scheme for polynomial factorization described in the introduction we need to approximate $2q+1$ coefficients $x_{-q}, \ldots, x_q$ of the Laurent series $x(z)$ such that

$$x(z)a(z) = 1, \quad x(z) = \sum_{i \in \mathbb{Z}} x_i z^i. \tag{3}$$

To this purpose, observe that under condition (1) the Fourier coefficients x_i of $x(z)$ decay exponentially to zero as $|i|$ approaches infinity so that we may look at the Laurent series $x(z)$ as at a polynomial plus a small error term. This suggests an algorithm to directly approximate $x(z)$ as the reciprocal of $a(z)$ by means of the following evaluation/interpolation strategy:

Algorithm 1 – evaluation/interpolation

INPUT: The coefficients a_i, $-m \le i \le n-m$, of the polynomial $a(z)$ having zeros ξ_i, $1 \le i \le n$ verifying (1); an integer q and a given tolerance $\epsilon > 0$.

OUTPUT: Approximations $\tilde{x}_i$ within the error bound ϵ of the $2q+1$ central coefficients x_i of the Laurent polynomial $x(z)$ satisfying (3).

Computation:

STAGE 1: Chose N "large enough".

STAGE 2: Evaluate $a(z)$ at the N-th roots of 1 to obtain $w_i = a(\omega_N^i)$, $i = 0, \ldots, N-1$.

STAGE 3: Compute $t_i = 1/w_i$, $i = 0, \ldots, N-1$.

STAGE 4: Interpolate (ω_N^i, t_i), $i = 0, \ldots, N-1$ and obtain coefficients $s_j = \frac{1}{N}\sum_{i=0}^{N-1} \omega_N^{-ij} t_i$, $j = 0, \ldots, N-1$.

STAGE 5: Output $\tilde{x}_j = s_{j \bmod N}$, $j = -N/2, \ldots, N/2-1$ to recover the desired coefficients.

The effectiveness of the above algorithm relies on the following result [2] which follows from the Cauchy integral representation of the Laurent coefficients of a meromorphic function [9].

Theorem 2 *Under the hypotheses (1) and (3), let* $|\xi_m| < r < 1 < R < |\xi_{m+1}|$ *and* $\rho = \max\{r, \frac{1}{R}\}$. *There exists a constant* η *such that for any even* N

$$|\tilde{x}_j - x_j| \le \eta\rho^{N-|j|}, \quad j = -N/2, \ldots, N/2-1, \tag{4}$$

where $\tilde{x}_j = \dfrac{1}{N}\displaystyle\sum_{i=0}^{N-1} \frac{\omega_N^{-ij}}{a(\omega_N^i)}$.

The only delicate part in the otherwise straightforward Algorithm 1 is the choice of a suitable N, which clearly depends both on the desired accuracy ϵ and on the separation factor ρ. The inequality in equation (4), depending on a priori unknown parameters, cannot be directly used to determine the correct value of N for Algorithm 1. Nevertheless, under some quite reasonable heuristic assumptions we may construct an adaptive algorithm based on the evaluation/interpolation technique. Let us denote $e_i(N) = \tilde{x}_i(N) - x_i$, where $\tilde{x}_i(N) = \tilde{x}_i$ are the values obtained with N interpolation points, moreover, set $\varepsilon_i(N) = \tilde{x}_i(N) - \tilde{x}_i(2N)$. If we assume that there exists a positive constant $\delta < \eta$ independent of N such that $\delta\rho^N \le \max_{-q\le i\le q} |\tilde{x}_i - x_i| \le \eta\rho^N$ and, moreover, that $|e_i(N)| \ge |e_i(2N)|$ for any $N \ge N_0$ and for any i with $-q \le i \le q$, then we have $|e_i(N)| - |e_i(2N)| \le |\varepsilon_i(N)| \le |e_i(N)| + |e_i(2N)|$ from which it follows that $(\delta - \eta\rho^N)\rho^N \le \max_{-q\le i\le q} |\varepsilon_i(N)| = \varepsilon(N) \le \eta\rho^N(1+\rho^N)$. Therefore, we have

$$\frac{\delta - \eta\rho^{2N}}{\eta(1+\rho^N)}\rho^N \le \frac{\varepsilon(2N)}{\varepsilon(N)} \le \frac{\eta}{\delta - \eta\rho^{2N}}\rho^N(1+\rho^{2N}).$$

This provides us with the following heuristic strategy:

STAGE 1: Chose $N = 2^k$, $k = \lceil \log_2 n \rceil$ and compute $\tilde{x}_i(N)$ by means of the evaluation/interpolation technique.

STAGE 2: Set $M = 2N$ and compute $\tilde{x}_i(M)$.

STAGE 3: Evaluate $\varepsilon(N)$; if $\varepsilon(N) < \epsilon$, where $\epsilon > 0$ is a given error bound, then stop. Otherwise set $N = M$ and continue from stage 2.

The overall computational cost of such a scheme for the approximation of the central coefficients of $x(z)$ is therefore given by $N \log_2 N$, for $N = O(\log \epsilon^{-1} / \log \rho^{-1})$, N being the maximum size used in the doubling scheme.

3 Graeffe's iterations

A better way to compute $x(z)$ such that $a(z)x(z) = 1$ is obtained by means of Graeffe iterations. Graeffe's method has recently received a renewed attention: it is used for the inversion of circulant matrices [1], for reciprocating power series [19, 3] and for factoring polynomials and analytic functions [12, 4, 3, 14]. The origins of Graeffe's iteration as well as the analysis of its convergence properties are well described in Householder's book [10].

If we multiply both sides of the identity $a(z)x(z) = 1$ by $a(-z)$ we obtain

$$a_1(z^2)x(z) = a(-z) \tag{5}$$

where $a_1(z^2) = a(z)a(-z)$. Now, if we split $x(z)$ and $a(z)$ into their even and odd parts: $x(z) = x^+(z^2) + zx^-(z^2)$, $\quad a(z) = a^+(z^2) + za^-(z^2)$, we may rewrite equation (5) as $x^+(z)a_1(z) = a^+(z)$, $\quad x^-(z)a_1(z) = -a^-(z)$. From the latter equation we may finally recover $x^+(z)$ and $x^-(z)$ as:

$$x^+(z) = a^+(z)x_1(z), \quad x^-(z) = -a^-(z)x_1(z) \tag{6}$$

where $x_1(z)$ is such that $x_1(z)a_1(z) = 1$. The latter equation is of the same nature of the original one and, hence, it can be recursively solved in exactly the same way. Therefore, one can proceed by generating the Graeffe sequence $\{a_i(z)\}$ formed from the Laurent polynomials $a_i(z)$, $i = 1, 2 \ldots, k$, defined by $a_{i+1}(z^2) = a_i(z)a_i(-z)$ with $a_0(z) = a(z)$. If $a_k(z)$ is easily invertible, we compute $x_k(z) = 1/a_k(z)$ and from $x_i(z)$ we recover $x_{i-1}(z)$ for $i = k, k-1, \ldots, 1$ by using (6) adjusted to the i-th step.

For the purpose of avoiding overflow problems which might occur in the construction of the Graeffe polynomial sequence, it is useful to introduce the *normalized Graeffe sequence* $\{a_i(z)\}_i$ given by

$$a_0(z) = \frac{a(z)}{\| a(z) \|_\infty}, \quad \tilde{a}_{i+1}(z^2) = a_i(z)a_i(-z), \quad a_{i+1}(z) = \frac{\tilde{a}_{i+1}(z)}{\| \tilde{a}_{i+1}(z) \|_\infty}, \tag{7}$$

where $\| p(z) \|_\infty$ is the of maximum modulus of the coefficient of $p(z)$.

We also point out that the above recursion scheme is always eventually stopped since, under hypothesis (1), the normalized Graeffe sequence is such that $\lim_i a_i(z) = 1$. In fact, from (7) it holds that the zeros of $a_i(z)$ are $\xi_j^{2^i}$, $j = 1, \ldots, n$. Therefore, recalling that the coefficients of a polynomial are the elementary symmetric functions of its zeros, we find that $|a_i^{(j)}/a_0^{(j)}| = O(|\xi_{m+i} \cdots \xi_m|^{2^j})$ for $i < 0$ and $|a_i^{(j)}/a_0^{(j)}| = O(1/(|\xi_{m+1} \cdots \xi_{m+i}|^{2^j})$ for $i > 0$. Hence, the normalized Graeffe sequence $\{a_i(z)\}$ tends to the identity function with double exponential speed, i.e., at the i-th step we obtain:

$$\| a_i(z) - 1 \|_\infty = O(|\xi_m|^{2^i} + |\xi_{m+1}|^{-2^i}) = O(\rho^{2^i}).$$

So, after at most $k = O(\log\log\epsilon^{-1} - \log\log\rho^{-1})$ steps, we obtain that the condition $\| a_k(z) - 1 \|_\infty < \epsilon$ is fulfilled. Then, we may stop the forward recursion, replacing $a_k(z)$ and $x_k(z) = 1/a_k(z)$ with 1 and start the "back substitution phase" recovering by means of (6) the coefficients of $x_{k-1}(z), \ldots, x_1(z)$ and finally of $x(z)$.

By carefully analyzing the outlined recursion scheme it turns out that the back-substitution stage is equivalent to the following identity:

$$a_{k+1}(z^{2^{k+1}}) = a(z)[a(-z)a_1(-z^2)a_2(-z^4) \cdots a_k(-z^{2^k})]$$

which can be easily proved by induction over k. This implies that $x_j(z) = a_j(-z)a_{j+1}(-z^2) \cdots a_k(-z^{2^{k-j}})$, from which one finds that in order to compute the $2q+1$ central coefficients of $x(z)$ it suffices to keep $2q+1$ central coefficients of $x_j(z)$, for any j. This property is used to simplify the resulting recursive scheme shown below where, given a Laurent power series $b(z)$ and an integer k, we denote by $\mathrm{Central}_k[b(z)]$ the Laurent polynomial $\sum_{i=-k}^{k} b_i z^i$ formed by the $2k+1$ central coefficients of $b(z)$.

Algorithm 2 – normalized Graeffe's iteration

INPUT: The coefficients a_i, $-m \le i \le n-m$, of the polynomial $a(z)$ having zeros ξ_i, $1 \le i \le n$ verifying (1); an integer q and a given tolerance $\epsilon > 0$.

OUTPUT: Approximations $\tilde{x}_i$ within the error bound $O(\epsilon)$ of the $2q+1$ central coefficients x_i of the Laurent polynomial $x(z)$ satisfying (3).

Computation:

Stage 1 (Normalized Graeffe) Set $a_0(z) = a(z)/ \parallel a(z) \parallel_\infty$,
Repeat for $i = 1, 2, \ldots$
$\tilde{a}_i(z) = a_{i-1}(\sqrt{z})a_{i-1}(-\sqrt{z})$,
$a_i(z) = \tilde{a}_i(z)/ \parallel \tilde{a}_i(z) \parallel_\infty$.
Until $\parallel a_i(z) - 1 \parallel_\infty \leq \epsilon$
Set $k = i$

Stage 2 (Back Substitution) Set $x_k(z) = a_k(z)$
For $i = k, k-1, \ldots, 1$
$x_{i-1}(z) = \text{Central}_q[a_{i-1}(z)x_i(z^2)]$
Output $\text{Central}_q[x_0(z)]$

This algorithm is very attractive from a computational point of view. In fact, at each step the amount of work essentially reduces to a polynomial multiplication which can be performed by means of very efficient and stable techniques based on the Fast Fourier Transform, on Karatsuba's algorithm, or on a combination of both [5]. In particular, the overall computational cost of Algorithm 2 is $O(n \log n(\log\log \epsilon^{-1} - \log\log \rho^{-1}))$ ops if FFT is used for polynomial multiplication.

In past years several authors [10, 8] have realized that the exponential growth of the coefficients of the polynomials generated by classical Graeffe's iteration is a very serious drawback for any floating-point implementation of this process. Approaches for taking under control this growth are suggested in [9, 8] and, more recently, a suitable renormalization technique has been proposed in [15] but, however, all these variants are time consuming as they cannot be implemented using superfast multiplication methods. In this way, up to day Graeffe's method for polynomial factorization is not generally indicated as an effective numerical algorithm and its use is rather restricted to the computational complexity framework . Our main claim which will be strengthened in the next section by means of numerical experiments is that Algorithm 2 is both practically very effective and, at the same time, it reaches the best record complexity estimates for polynomial factorization. In this respect, a quite interesting feature of the proposed scheme is that only polynomial multiplications are required to approximate $x(z)$ and, hence, no potentially numerically unstable operations like polynomial division or solving linear systems must be performed. Moreover, approximations of the sought coefficients are constructed by using the information returned by the scaled Graeffe's process step-by-step. Differently, the previously known implementations of Graeffe's method for polynomial factoring make use only of the coefficients of the last polynomial iterates whose relative precision can seriously be deteriorated.

A worst-case analysis reveals that the approximations of $x_{-q} \ldots x_q$ returned by Algorithm 2 are within an error bound $O(\epsilon)$ provided that computations are performed using $O(n(\log n + |\log\log \rho^{-1}|))$ bits of precision where $\log \epsilon^{-1} = O(n \log n)$.

A final remark is concerned with the possibility of dynamically operating suitable scalings of the variable z in order to improve the separation factor and, in this way to accelerate the convergence of Graeffe's process. If we appropriately scale the variable z by a constant θ, i.e., we replace $a(z)$ with $b(z) = a(\theta z)$ having

n	Steps	Res	t_G	t_{BS}	t_T
200	13	4e-15	1e-3	3e-3	2e-3
1000	15	5e-14	3e-2	6e-2	2e-2
4000	17	5e-13	8e-1	1.4	0.4
16000	19	2e-12	23	37	17

Table 1: Polynomial 1: Graeffe with customary multiplication

zeros $\eta_i = \xi_i/\theta$, $1 \le i \le n$, then the above algorithm may converge even if $b(z)$ has zeros on the unit circle or if the splitting circle is not the unit one. An interesting application of such property is to balance the polynomials so that the separation factor ρ in (4) is optimized, thus improving the numerical performance of the method. If $\theta = \sqrt{|\xi_m \xi_{m+1}|}$ then $|\eta_m| = 1/|\eta_{m+1}| = \sqrt{|\xi_m/\xi_{m+1}|} \le \rho$ so that the isolation factor for $b(z)$ is improved so that the corresponding normalized Graeffe sequence obtained from $b(z)$ will converge to 1 faster. Clearly, in common situations neither $|\xi_m|$ nor $|\xi_{m+1}|$ is known a priori and, so we need a different practical way to obtain the seeked balancing. This can be achieved by observing that $|a_{-1}^{(j)}| = O(|\xi_m|^{2^j})$ and $|a_1^{(j)}| = O(|\xi_{m+1}|^{-2^j})$. Therefore, if we set $r = \max\limits_{i<0} |a_i^{(j)}| / \max\limits_{i>0} |a_i^{(j)}|$, whenever we detect that $r \ll 1$ or $r \gg 1$ we need to rebalance, so we may set $\omega = \sqrt{|a_{-1}^{(j)}/a_1^{(j)}|}$, $\quad \tilde{a}_{j+1}(z) = a_j(\sqrt{\omega z})a_j(-\sqrt{\omega z})$. It is easily seen that the result of such scaling is the same as replacing $a(z)$ with $b(z) = a(\sqrt[2^j]{\omega} z)$ at the initial step, where $|\sqrt[2^j]{\omega}| \simeq \sqrt{|\xi_m \xi_{m+1}|}$.

4 Numerical Experiments

We report here the results of some numerical experiments performed with an implementation in Fortran 90 of the described algorithms where we used the TOMS729 package for solving Toeplitz systems.

The following two polynomials have been considered which significantly show the complementarity of the methods.

Poly 1: $(m + x + x^2 + ... + x^m)(1 + x + x^2 + ... + mx^m)$,
$p_i = p_{n-i} = m + i$, $i = 0, \ldots, m-1$, $p_m = m^2 + m$
$n = 2m, \quad \rho \approx 1 - 1/n$;

Poly 2: $(2 + x + x^2 + ... + x^m)(1 + x + x^2 + ... + 2x^m)$
$p_i = p_{n-i} = 2 + i$, $i = 0, \ldots, m-1$, $a_m = m + 4$
$n = 2m, \quad \rho \approx 1 - 1/n^3$.

In the tables, besides the maximum absolute approximation error *Err* in the coefficient, we report the number *Steps* of steps required by Graeffe's iteration, the logarithm *Log(N)* of the number of Fourier points needed by the evaluation/interpolation scheme, and the CPU time in seconds required in the computation of the Graeffe iteration t_G, by the back-substitution stage t_{BS}, by the Toeplitz matrix inversion t_T and for the evaluation/interpolation scheme t_{FFT}. From Table 2 we deduce that for the first class of polynomials, for which the spectral factorization problem is well conditioned in the sense of [2], both methods work effectively. Moreover, the method based on evaluation/interpolation

n	Steps	Err	t_G	t_{BS}	Log(N)	Err	t_{FFT}
1000	15	7e-14	2e-2	0.1	14	6.2e-13	1.4e-2
4000	17	3e-13	0.11	1.3	16	2.9e-12	9.7e-2
16000	19	8e-12	1.1	8.7	18	1.0e-10	0.63
64000	21	3e-11	6.0	42.3	20	1.9e-10	1.9
200000	23	1e-10	25.3	93	22	1.9e-9	9.0
1000000	25	9e-10	122.7	896	24	5.7e-9	42.3

Table 2: Polynomial 1: Graeffe with Fast polynomial multiplication/FFT

n	Steps	Err	t_G	t_{BS}	t_T	Log(N)	Err	t_{FFT}
200	21	4e-10	2e-3	2e-3	5e-4	19	1.7e-11	0.9
400	24	5e-9	6e-3	1e-2	5e-3	22	1.5e-12	25
1000	28	1e-7	5e-2	1e-1	2e-2	24	1.0e-6	37
2000	31	4e-6	0.2	0.5	8e-2	24	3.4e-5	38

Table 3: Polynomial 2: Graeffe vs. FFT

has a lower cost in terms of CPU time for the computation in the IEEE customary arithmetic. For the second polynomial, for which the problem is ill conditioned according to [2], the Graeffe iteration is much faster than the evaluation/interpolation method which requires a quite large number of points for convergence.

On the stability of Graeffe iterations. It is worth mentioning that in some cases the generation of the Graeffe sequence $a_k(z)$ may become a severely unstable process even though factoring the polynomial $a_0(z)$ is a well conditioned problem. This phenomenon happens when a polynomial with very ill-conditioned zeros is encountered in the Graeffe sequence. In fact, in this case, small perturbations of the coefficients of $a_k(z)$, say, generated by roundoff, may move the zeros far away from the unit circle changing the number of the zeros inside/outside the unit disk. Ill conditioning of the zeros is particularly severe for polynomials with multiple zeros where a perturbation ϵ in the coefficients turns into a variation $O(\epsilon^{1/m})$ in the zero of multiplicity m, leading to an asymptotical reduction of the number of accuracy digits by a factor $1/m$.

This happens for instance when the starting polynomial has several zeros with the same modulus and such that, under repeated squaring, they all collapse into the same point at some stage of the process. One of such polynomials, for instance, can be constructed as follows: start with the polynomial

$$q(y) = y^2 - (\frac{n}{n+1} + \frac{n+1}{n})y + 1 \tag{8}$$

whose zeros are $\xi_1 = \frac{n}{n+1} < 1 < \frac{n+1}{n} = \xi_2$, with a separation factor of roughly $1 - 1/n$. Then set $y = z^m$ with $m = 2^k$. The zeros of the resulting polynomial $p(z) = q(z^m)$ are given by all the m-th zeros of both ξ_1 and ξ_2, the separation factor becomes roughly $\rho = 1 - 1/(mn)$.

As shown in the previous numerical examples, separation factors like this do not constitute, in general, a serious problem for the convergence of the outlined Graeffe method. Nevertheless, in this wicked case the zeros of the Laurent polynomials $\{a_j(z)\}_j$ generated by the Graeffe sequence become more and more ill conditioned as j grows from 1 to k. In fact, at each step, the number of distinct zeros halves down to 2, so that $a_k(z)$ has the zeros ξ_1 and ξ_2 , each with multiplicity $m = 2^k$. So, after k iterations, the isolation ratio has only improved linearly by k bits but now, because of the multiple zeros, the number of significant bits in the zeros has decreased by the exponential factor 2^k. Consequently, for $k = 5$ the whole (double) IEEE machine precision is eaten up completely just in 5 iterations.

A similar phenomenon clearly happens whenever the starting polynomial has a factor resembling the one above. In fact, we encounter the amazing case of the polynomials (8) where m is an unconstrained integer. The Graeffe method works fine also for very high degrees (thousands) for many values of m but whenever a big enough power of 2 divides m the method breaks down after a few steps. An easy remedy for such cases is given by applying a random rotation of the complex variable and to consider the polynomial with complex coefficients $p(\omega x)$ for $\omega = e^{\boldsymbol{i}\theta}$ for a random θ. In this way, the arguments of the zeros of $a(z)$ will no more add up to form new multiple zeros in the sequence. In this way, however, the resulting computational cost of the Graeffe iterations roughly grows by a factor of 4 since we need to perform complex arithmetic. Numerical experiments with the random rotation show the effectiveness of this technique.

References

[1] D. Bini, G. Del Corso, G.Manzini, and L. Margara. Inversion of circulant matrices over Z_m. *Math. Comp.*, 70(235):1169–1182, 2001.

[2] D. A. Bini and A. Bottcher. Polynomial factorization through Toeplitz matrix computations. To appear on Linear Algebra Appl..

[3] D. A. Bini, L. Gemignani, and B. Meini. Computations with infinite Toeplitz matrices and polynomials. To appear on Linear Algebra Appl..

[4] D. A. Bini, L. Gemignani, and B. Meini. Factorization of analytic functions by means of Koenig's theorem and Toeplitz computations. *Numer. Math.*, 89:49–82, 2001.

[5] D. A. Bini and V. Pan. *Matrix and Polynomial Computations, Vol. 1: Fundamental Algorithms.* Birkhäuser, Boston, 1994.

[6] P. A. Van Dooren. Some numerical challenges in control theory. In *Linear Algebra for Control Theory*, volume 62 of *IMA Vol. Math. Appl.* Springer, 1994.

[7] L. Gemignani. Computing a factor of a polynomial by means of multishift LR algorithms. *SIAM J. Matrix Anal. Appl.*, 19(1):161–181, 1998.

[8] A. A. Grau. On the reduction of number range in the use of Graeffe process. *J. Assoc. Comput. Mach.*, 10:538–544, 1963.

[9] P. Henrici. *Applied and Computational Complex Analysis*, volume 1. Wiley, 1974.

[10] A. S. Householder. *The Numerical Treatment of a Single Nonlinear Equation.* McGraw-Hill Book Company, N.Y., 1970.

[11] T. Kailath and A. H. Sayed, editors. *Fast Reliable Algorithms for Matrices with Structure.* SIAM Publications, 1999.

[12] P. Kirrinnis. Partial fraction decompostion in $\mathbf{C}(z)$ and simultaneous Newton iteration for factorization in $\mathbf{C}[z]$. *J. Complexity*, 14(3):378–444, 1998.

[13] P. Kravanja and M. Van Barel. *Computing the zeros of analytic functions.* Number 1727 in Lectures Notes in Mathematics. Sringer, 2000.

[14] G. Malajovich and J. P. Zubelli. A fast and stable algorithm for splitting polynomials. *Comput. Math. Appl.*, 33(3):1–23, 1997.

[15] G. Malajovich and J. P. Zubelli. Tangent Graeffe iteration. Informes de Matemática Série B-119, IMPA, 1998. To appear on Numer. Math..

[16] M. F. Neuts. *Structured Stochastic Matrices of M/G/1 Type and Their Applications.* Dekker Inc., 1989.

[17] V. Y. Pan. Optimal and nearly optimal algorithms for approximating complex polynomial zeros. *Comput. Math. Appl.*, 31:97–138, 1996.

[18] A. Schönhage. Equation solving in terms of computational complexity. In *Proceedings of the International Congress of Mathematicians, Vol. 1, 2 (Berkeley, Calif., 1986)*, pages 131–153, Providence, RI, 1987. Amer. Math. Soc.

[19] A. Schönhage. Variations on computing reciprocals of power series. *Inform. Process. Lett.*, 74:41–46, 2000.

[20] G. Rodriguez T. N. T. Goodman, C. A. Micchelli and S. Seatzu. Spectral factorization of Laurent polynomials. *Adv. Comput. Math.*, 7:429–454, 1997.