2011

TR-2011004: Acceleration of Newton's Polynomial Factorization: Army of Constraints, Convolution, Sylvester Matrices, and Partial Fraction Decomposition

Victor Y. Pan

Acceleration of Newton's Polynomial Factorization: Army of Constraints, Convolution, Sylvester Matrices, and Partial Fraction Decomposition [*]

Victor Y. Pan

Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/

Abstract

We try to arm Newton's iteration for univariate polynomial factorization with greater convergence power by shifting to a larger basic system of multivariate constraints. The convolution equation is a natural means for a desired expansion of the basis for this iteration versus the classical univariate method, which is more vulnerable to foreign distractions from its convergence course. Compared to Viete's equations, the convolution equation directs the Newton's root-finding iteration to factorization (which is a task of independent interest) and enables approximation of a single root. Combining convolution with partial fraction decomposition (PFD) yields even a greater army of constraints. By linking PFD with Sylvester and generalized Sylvester matrices we extend to their inverses the celebrated formula by Gohberg and Semencul for Toeplitz matrix inversion. Furthermore, we accelerate the solution of Sylvester and generalized Sylvester linear systems in the important case where all but one of the basic polynomials defining the matrix have small degrees. This enables us to speed up Newton's convolution steps.

Key words: Newton's polynomial factorization, Army of constraints, Convolution, Sylvester matrices, PFD

---

[*] Supported by PSC CUNY Awards 62230–0040 and 63153–0041

1 Introduction

The problems of polynomial factorization and root-finding are classical and remain highly important for modern computations. Their study has begun at least four millennia ago and still remains extensive (see McNamee (1993, 1997, 2002 and 2007). The algorithms in Pan (1995, 1996, 2001a, and 2002) solve both problems (as well as the related problem of root isolation in the case of integer input coefficients) by using arithmetic and Boolean time which is optimal up to polylogarithmic factors in the input size. The algorithms extend the work of Schönhage (1982) whose solution was slower by the order of magnitude but served as a springboard for devising the cited nearly optimal algorithms.

Their basic step is approximate splitting of a polynomial of a degree n into the product of two factors; this step is repeated recursively in the divide and conquer fashion. Splitting begins slowly with computing crude initial approximate factors; then one rapidly refines them by applying Newton's iteration. Kirrinnis (1998) extended Newton–Schönhage's techniques to refine splitting into the m factors for any m, $2 \leq m \leq n$. As by-product his algorithm refines the associated approximate partial fraction decomposition of the polynomial reciprocal $1/p(x)$. Hereafter we will use the acronym "*PFD*" for "partial fraction decomposition". Splitting method for polynomial root-finding is attractive because besides (and actually prior to) obtaining roots it yields factorization of a degree n polynomial into the product of n linear factors, which is an important goal in its own right due to its applications to the time series analysis, Weiner filtering, noise variance estimation, covariance matrix computation, and the study of multi-channel systems (see Wilson (1969), Box and Jenkins (1976), Barnett (1983), Demeure and Mullis (1989 and 1990), Van Dooren (1994)).

Our examination reveals that in the case where the factors have degree one the Kirrinnis' algorithm is closely linked to Newton's classical iteration for approximating a root of a univariate equation. At this point we wish to state as a conjecture the following *Principle of Arming with Constraints*, for which we use the acronym *PAC*:

Suppose Newton's iteration has been applied to approximate a root r of an equation (in our case polynomial equation) beginning with a crude initial approximation. Then one can enhance convergence power by applying Newton's iteration to an appropriate system of multivariate equations (in our case algebraic equations) whose root set includes r. Moreover, the more equations and varaibles are involved in such a system, the greater convergence power can be achieved. The PAC can be extended to iterations extending Newton's as well.

We do not know if this principle has ever been formulated, but we observe clear, strong and consistent empirical support for it from the polynomial root-finders based on Viete's equations (such as Durand–Kerner iteration (due to Weierstrass (1903)), Aberth or Ehrlich–Aberth iteration (due to Börsch-Supan (1963))) and on matrix methods involving eigenvectors (see Pan and Zheng (2011) and the bibiliography therein).

Now by following this principle we employ the convolution equation as the basis for Newton's multivariate iteration for univariate polynomial factorization and root-finding. According to our preliminary tests this transition does enhance the convergence power compared to Newton's classical univariate iteration. Furthermore, we can yield an additional group of constrains by including equations (2.2) or (2.3) in Section 2.

In the next section we focus on employing the same link to PFDs to accelerate Newton's iteration steps for the convolution equation. Further link of the PFDs to Sylvester linear systems of equations enables us to extend the celebrated formula by Gohberg and Semencul (1972) from Toeplitz to Sylvester and generalized Sylvester matrices. Furthermore, this link also enables us to accelerate the solution of Sylvester and generalized Sylvester linear systems in the important case where all but one of the basic polynomials defining the matrix have small degrees. This implies a speed up of Newton's convolution steps.

Let us restate and summarize our contributions.

1. Our conjectured Principle of Arming with Constraints (PAC) could be tested as a guidance in designing iterative algorithms for a large class of problems. We pursue this principle for polynomial root-finding by shifting from the Newton's classical univariate iteration $z_i^{(k+1)} = z_i^{(k)} - \frac{p(z_i^{(k)})}{p'(z_i^{(k)})}$ for $k = 0, 1, \dots$ to Newton's multivariate iteration for the convolution equation $p = L_1 L_2$. This transition substantially increases the convergence power of the iteration according to our tests. One can try to add our equations (2.2) and (2.3) in Section 2 to expand a system of constraints (and variables) for the same task of univariate polynomial root-finding.

Unlike the Durand–Kerner's, Aberth's and other iterations based on Viete's equations, we can apply our process directly to factorization and to the approximation of even a single root.

2. We express the solution of a nonsingular Sylvester or generalized Sylvester system of n equations $S\mathbf{y} = \mathbf{b}$ via computing the last column of the inverse S^{-1} and performing $O(n \log n)$ additional field operations. This extends the celebrated formula in Gohberg and Semencul (1972) for Toeplitz inverses.

3. If a nonsingular Sylvester matrix $S = S(Q_1, Q_2)$ is defined by two polynomials Q_1 and Q_2 and if $\deg Q_i$ for $i = 1$ or $i = 2$ is in $O(1)$, that is bounded by a constant, then we can solve the linear system $S\mathbf{y} = \mathbf{b}$ in $O(n)$ field operations. The result also holds for nonsingular $n \times n$ generalized Sylvester matrices $S = S(Q_1, \dots, Q_m)$ where $\deg Q_1, \dots, \deg Q_{m-1}$ and m are in $O(1)$. These algorithms are simple and should have been known for a long while, but we are not aware of any previous results in this direction.

4. Solution of Sylvester linear systems is required, e.g., for polynomial root-finding and computing approximate polynomial GCDs and LCMs by means of Newton's iteration, and so our results imply acceleration of the known algorithms for some of these important problems. Here and hereafter we use the acronyms "GCD" for "greatest common divisor" and "LCM" for "least common multiple".

2 Computation of PFDs

Hereafter we write $u = u(x)$ to denote the polynomial with a coefficient vector $\mathbf{u}$, and we simplify our notation by writing u for a polynomial $u(x) = \sum_i u_i x^i$ wherever this causes no confusion.

M^T denotes the tanspose of a matrix or vector M. $[M_1|M_2| \ \ldots \ | \ M_s]$ is a $1 \times s$ block matrix with the blocks $M_1, \ldots, M_s$.

$\deg u$ denotes the degree of a polynomial $u = u(x)$. $\gcd(u, v)$ denotes the monic greatest common divisor of two polynomials $u = u(x)$ and $v = v(x)$; $\operatorname{lcm}(u, v)$ denotes their least common multiple.

Assume a polynomial

$$p = p(x) = \sum_{i=0}^{n} p_i x^i = p_n \prod_{j=1}^{n} (x - z_j), \quad p_n \neq 0. \tag{2.1}$$

Let m, n, $n_1, \ldots, n_m$ denote positive integers such that $2 \leq m \leq n$ and $n_1 + \ldots + n_m = n$. For a monic polynomial p of degree n and a polynomial $T = T(x)$ of degree at most $n - 1$ and coprime with p, we seek pairwise prime monic polynomials $L_1, \ldots, L_m$ and polynomials $V_1, \ldots, V_m$, deg $V_i <$ deg $L_i = n_i$, $i = 1, \ldots, m$, defining the factorization $p = L_1 \cdots L_m$ and the PFD

$$\frac{T}{p} = \frac{V_1}{L_1} + \cdots + \frac{V_m}{L_m}. \tag{2.2}$$

Multiply this PFD by p and obtain the equivalent polynomial equation

$$Q_1 V_1 + \cdots + Q_m V_m = T \tag{2.3}$$

where

$$Q_i = p/L_i, \ V_i = (T L_i / p) \mod L_i, \ i = 1, \ldots, m. \tag{2.4}$$

We can obtain the polynomials $W_i = (L_i/p) \mod L_i$ by solving the PFD problem for $T = 1$; then we can readily obtain $V_i = T W_i \mod L_i$ for all i.

Alternatively the coefficient vectors of the polynomials $V_1, \ldots, V_m$ can be obtained from a linear system of equations

$$S(Q_1, \ldots, Q_m)\mathbf{V} = \mathbf{T}. \tag{2.5}$$

Here $\mathbf{V}^T = (\mathbf{V}_1^T, \ldots, \mathbf{V}_m^T)$, $\mathbf{V}_j$ denotes the coefficient vectors of the polynomials V_j for $j = 1, \ldots, m$, $\mathbf{T}$ is the coefficient vector of T, so that $\mathbf{T} = \mathbf{e}_n = (0, \ldots, 0, 1)^T$ is the nth coordinate vector of dimension n for $T = 1$, the coefficient matrix $S(Q_1, \ldots, Q_m) = [C_{n-n_1}(Q_1) \ | \ \ldots \ | \ C_{n-n_m}(Q_m)]$ is the $1 \times m$ block matrix with the blocks $C_{n-n_1}(Q_1), \ldots, C_{n-n_m}(Q_m)$, and $C_k(w) =$

$$\begin{pmatrix} w_l & & & O \\ \vdots & \ddots & & \\ \vdots & \ddots & \ddots & \\ \vdots & \ddots & \ddots & w_l \\ w_0 & \vdots & \ddots & \vdots \\ \ddots & \ddots & \vdots & \\ & & \ddots & \vdots \\ O & & & w_0 \end{pmatrix}$$ denotes the kth convolution matrix of a polynomial $w(x) = \sum_{i=0}^{l} w_i x^i$, that is the $(k+l) \times (k+1)$ Toeplitz matrix defined by its first column $(w_l, w_{l-1}, \dots, w_0, 0 \dots, 0)^T$ and its first row $(w_l, 0, \dots, 0)$. $S(Q_1, \dots, Q_m)$ is a generalized Sylvester matrix, becoming Sylvester matrix $S(Q_1, Q_2) = S(L_2, L_1)$ in the case where $m = 2$. We recall that a Sylvester matrix is nonsingular if and only if its basic polynomials Q_1 and Q_2 are coprime.

Equations (2.4) enable us to express the solution of a generalized Sylvester linear system (2.5) defined by coprime polynomials $L_1, \dots, L_m$ and any vector $\mathbf{T}$ via its solution for $T = 1$. By introducing a random scalar parameter τ, we can express the solution of equation (2.5) for any T via the solution for $W = T - \tau$ and $T = \tau$ and observe that the polynomials W and p are coprime with a probability near one.

The extension of the solution from the special case where $T = 1$ to the case of general polynomial T via equations (2.4) takes $O((n \log n) \log m)$ arithmetic operations (see Problem 4.1 (POL·MODULI) in Bini and Pan (1994)). According to the two following theorems the cost bound decreases to $O(n)$ where $m = O(1)$ and the factors $L_1, \dots, L_{m-1}$ are pairwise coprime and have degrees in $O(1)$. The first theorem covers the simpler but important case where $m = 2$.

Theorem 2.1. *Suppose* $S(L_2, L_1)$ *denotes the* $n \times n$ *Sylvester matrix defined by two coprime polynomials* L_1 *and* L_2 *where* $d_1 = \deg L_1$, $d_2 = \deg L_2 = n - d_1$. *Then a Sylvester linear system* $S\mathbf{y} = \mathbf{T}$ *of* n *equations with this matrix can be solved by using* $O(n)$ *arithmetic operations.*

Proof. Perform the following steps, each taking $O(n)$ arithmetic operations.

Algorithm 2.1. Sylvester Solving.

1. *Compute the polynomial* $p = L_1 L_2$.
2. *Compute the polynomial* V_1 *by applying the equation* $V_1 = (TL_1/p) \mod L_1$ *in (2.4),* $T = T(x)$ *denoting the polynomial with the coefficient vector* $\mathbf{T}$.
3. *Compute the polynomial* $T - V_1 L_2$.
4. *Compute the polynomial* $V_2 = (T - V_1 L_2)/L_1$ *(cf. equation (2.3)). Output the vector* $\mathbf{V} = [\mathbf{V}_1^T, \mathbf{V}_2^T]^T$. *(Here* $\mathbf{V}_i$ *for* $i = 1, 2$ *denotes the coefficient vector of the polynomial* V_i.*)*

□

Theorem 2.2. *Suppose* $S = S(Q_1, \ldots, Q_m)$ *is the* $n \times n$ *generalized Sylvester matrix defined by* m *polynomials* $L_1, \ldots, L_m$ *such that* $\gcd(L_1, \ldots, L_m) = 1$, $Q_i = p/L_i$, $i = 1, \ldots, m$, $p = \prod_{i=1}^{m} L_i$, $\deg p = n$, $\deg L_i = d_i$ *for* $i = 1, \ldots, m$, *and* d_i *is in* $O(1)$ *for* $i = 1, \ldots, m-1$. *Then a Sylvester linear system* $S\mathbf{y} = \mathbf{T}$ *of* n *equations with this matrix can be solved by using* $O(n)$ *arithmetic operations.*

Proof. Here is our algorithm supporting the theorem. One can readily verify that all of its seven steps take $O(n)$ arithmetic operations.

Algorithm 2.2. Generalized Sylvester Solving.

INITIALIZATION: Write $p_0 = Q_m$.

COMPUTATIONS.

1. *Compute the polynomials* $p_i = \mathrm{lcm}(p_{i-1}, Q_i) = p_{i-1}Q_i / \gcd(p_{i-1}, Q_i)$ *for* $i = 1.2, \ldots, m-1$. *(This step takes* $O(n)$ *arithmetic operations because* $\max_i \deg p_i \leq \deg Q_m$ *and because both* m *and* $\deg Q_m = n - d_m = \sum_{i=1}^{m-1} d_i$ *are in* $O(1)$*.) Observe that* $p_{m-1} = \mathrm{lcm}(Q_1, \ldots, Q_m) = p$ *because by assumption* $\gcd(L_1, \ldots, L_m) = 1$.
2. *Compute the polynomials* L_i *for* $i = 1, \ldots, m$ *by applying the equations* p/Q_i *in (2.4).*
3. *Compute the polynomials* V_i *for* $i = 1, \ldots, m-1$ *by applying the equations* $V_i = (TL_i/p) \mod L_i$ *in (2.4).*
4. *Compute the polynomial* $W_m = T - \sum_{i=1}^{m-1} Q_i V_i$.
5. *Compute the polynomial* $V_m = W_m/Q_m$ *(cf. equation (2.3)). Output the vector* $\mathbf{V} = [\mathbf{V}_1^T, \ldots, \mathbf{V}_m^T]^T$.

□

The first step of Algorithm 2.2 involves computation of $m-1$ LCMs or GCDs. The following algorithm replaces this with computing a single LCM or GCD in the case where L_m and L_j are coprime for some fixed j.

Algorithm 2.3. Generalized Sylvester Solving simplified.

1. *Compute the polynomial* $g_{j,m} = \gcd(Q_j, Q_m)$.
2. *Compute the polynomial* $L_j = Q_m/g_{j,m}$.
3. *Compute the polynomial* $p = L_j Q_j$ *(see (2.4)).*
4. *Compute the polynomials* L_i *for all* $i \neq j$ *by applying the equations* p/Q_i *in (2.4).*

5. Perform the last three steps as in Algorithm 2.2.

Remark 2.1. *Algorithm 2.2 computes the solution of a generalized Sylvester linear system for any right-hand side vector if* $\gcd(L_1, \dots, L_m) = 1$*, that is if the* m *basic polynomials have only constant common factors.*

Remark 2.2. *In the case where* $L_j = x - z_j$ *is a monic linear factor of* p *the above computations are simplified because* $V_j = T(z_j)/p'(z_j)$*.*

3 Factorization via PFD and Newton's Iteration

Assume sufficiently close approximations $l_i = l_i^{(0)}$ to L_i, $v_i = v_i^{(0)}$ to V_i in (2.2) for $i = 1, \dots, m$ and $l^{(0)} = l_1^{(0)} \cdots l_m^{(0)}$ to p satisfying the PFD

$$\frac{1}{l^{(k)}} = \frac{v_1^{(k)}}{l_1^{(k)}} + \cdots + \frac{v_m^{(k)}}{l_m^{(l)}} \text{ for } k = 0. \tag{3.1}$$

Kirrinnis (1998) preserves the PFD while he recursively improves the initial approximations by the polynomials $l_1^{(0)}, \dots, l_m^{(0)}$ to $L_1, \dots, L_m$. He sets

$$l_i^{(k+1)} = l_i^{(k)} + \Delta_i^{(k)},\ i = 1, \dots, m;\ k = 0, 1, \dots \tag{3.2}$$

and computes the Newton's corrections $\Delta_1^{(k)}$, …, $\Delta_m^{(k)}$ from the PFD

$$\frac{p - l^{(k)}}{l^{(k)}} = \frac{\Delta_1^{(k)}}{l_1^{(k)}} + \cdots + \frac{\Delta_m^{(k)}}{l_m^{(k)}},\ \deg\ \Delta_i^{(k)} < \deg\ l_i^{(k)},\ i = 1, \dots, m. \tag{3.3}$$

Alternatively one can first compute the PFD

$$\frac{1}{l^{(k)}} = \frac{v_1^{(k)}}{l_1^{(k)}} + \cdots + \frac{v_m^{(k)}}{l_m^{(k)}},\ \deg v_i^{(k)} < \deg\ l_i^{(k)},\ i = 1, \dots, m$$

and then apply equation (2.4) to recover the correction values so that

$$\Delta_i^{(k)} = (v_i^{(k)} p) \bmod l_i^{(k)},\ i = 1, \dots, m. \tag{3.4}$$

In the case where $l_i^{(k)} = x - z_i^{(k)}$ is a monic linear factor, we arrive at the Newton's correction $\Delta_i^{(k)} = p(z_i^{(k)})/p'(z_i^{(k)})$ (cf. Remark 2.1), so that $l_i^{(k+1)} = l_i^{(k)} + p(z_i^{(k)})/p'(z_i^{(k)})$, $z_i^{(k+1)} = z_i^{(k)} - p(z_i^{(k)})/p'(z_i^{(k)})$. This defines Newton's classical iteration having local quadratic convergence. Kirrinnis (1998) generalizes it to splitting $p(x)$ into m factors, extends to this case the classical results on local quadratic convergence of Newton's iteration, and specifies the Boolean (that is bitwise) operation complexity provided that the factors $L_1, \dots, L_m$ as well as their initial approximations $l_1^{(0)} \approx l_m^{(0)}$ have pairwise isolated zero sets, all lying in the unit disc $D(0, 1) = \{x :\ |x| \le 1\}$. (We can move all zeros into this disc by scaling the variable x.) In the case of such factors he proposes to replace the above recipes for updating $l_i^{(k)}$ for $i = 1, \dots, m-1$ by the expressions

$$q_i^{(k)} = l^{(k)}/l_i^{(k)},\ \ l_i^{(k+1)} = l_i^{(k)} + ((2 - v_i^{(k)} q_i^{(k)}) v_i^{(k)} p \bmod l_i^{(k)}),\ i = 1, \dots, m,$$

with the goal of improving numerical stability of the computations.

4 Newton's Iteration for Convolution Equation

In view of its close link to the classical Newton's univariate root-finder, the PFD factorization method above by Schönhage and Kirrinnis preserves the benefits and limitations of this root-finder, and so by following the PAC we shall try to enhance the convergence power by applying Newton's multivariate iteration to refine the initial solution $l_1^{(0)} \approx L_1$ and $l_2^{(0)} \approx L_2$ to the convolution equation $p = L_1 L_2$.

The kth iteration step is essentially the solution of a Sylvester linear system with the Jacobian coefficient matrix $-S(l_2^{(k)}, l_1^{(k)})$ (see Zeng (2005), Bini and Boito (2010), Pan and Zheng (2011)). If $\deg L_i = \deg l_i^{(k)} = O(1)$ for $i = 1$ or $i = 2$, we can solve this linear system in $O(n)$ arithmetic operations by applying Algorithm 2.1. We can yield further simplifications where $\deg L_i = \deg l_i^{(k)} = 1$ for $i = 1$ or $i = 2$ (see Remark 2.1).

The iteration has local quadratic convergence, and so it can ensure fast refinement of a sufficiently close initial approximate factorization precomputed by another algorithm. Our acceleration of the solution of Sylvester linear systems is translated into acceleration of every iteration step.

Based on our conjecture about the PAC, we are motivated to try this iteration for randomized heuristic initial approximations where a factor $l_i^{(k)}$ is defined by a single complex parameter, e.g., where we seek a zero of a polynomial p or a pair of its complex conjugate zeros where its coefficients are real. This leads us to some standard initialization recipes known for polynomial root-finding (as well as for matrix eigen-solving) in the case where initial information about the location of the output values is limited or absent.

According to these recipes, the random initial values can be chosen near the origin, near the center of gravity $-p_{n-1}/(np_n)$ of the n zeros of p, on a large circle $\{x :\ |x| = R\}$ for $R \geq 2 \max_{i>1} |p_{n-i}/p_n|$ (cf. Habbard, Schleicher and Sutherland (2001)), or on the Bini's circles in Bini (1996) and Bini and Fiorentino (2000). One can try to apply the iteration successively or concurrently at a number of such initial points to increase the chances for its fast convergence.

For each approximate zero $z_1^{(0)}$ or a pair of complex conjugate zeros $z_1^{(0)} = r_1^{(0)} + i_1^{(0)}\sqrt{-1}$ and $z_2^{(0)} = \bar{z}_1^{(0)} = r_1^{(0)} - i_1^{(0)}\sqrt{-1}$ one can immediately define the initial linear or quadratic factor $l_1^{(0)} = x - z_1^{(0)}$ or $l_1^{(0)} = (x - z_1^{(0)})(x - \bar{z}_1^{(0)}) = x^2 - 2r_1^{(0)}x + (r_1^{(0)})^2 + (i_1^{(0)})^2$ and then initialize the coefficient vector of the second factor $l_2^{(0)}$ by setting this vector equal to least-squares solution of the overdetermined linear system $C_{n-1}(l_1^{(0)})\mathbf{l}_1^{(0)} = \mathbf{p}$ defined by the convolution equation $l_1^{(0)}l_2^{(0)} \approx p$ (cf. Corless et al. (1995)). Now one can refine this initial factorization by applying Newton's iteration and employing Theorem 2.1 and Remark 2.1.

5 Discussion

1. The convolution equation $p = L_1 L_2$ has two equivalent vector representations $C(L_1)\mathbf{L_2} = \mathbf{p}$ and $C(L_2)\mathbf{L_1} = \mathbf{p}$. Assuming a fixed approximation to L_1 (resp. L_2) one can approximate L_2 (resp. L_1) by computing the least squares solution of the former (resp. latter) vector equation. One can complement Newton's iteration for the convolution equation by occasionally updating approximations to both factors in this way.

2. In the spirit of PAC we can apply Newton's iteration to the convolution equation complemented with equations (2.3) (or (2.2)) because this increases the number of constraints in the system that we solve.

3. We plan to apply the PAC extensively in devising univariate and multivariate polynomial and possibly nonpolynomial root-finders and to study their power both theoretically and experimentally. Of course, various implementation "details" must be taken into account. E.g., seeking the zeros z_j of p that lie outside the unit disc $D(0,1)$ one should seek the factors $x/z_j - 1$ of p or the factors $x - 1/z_j$ of the reverse polynomial $p_{\rm rev} = \sum_{i=0}^{n} p_i x^{n-i}$ (rather than the monic linear factors $x - z_j$ of p) to improve numerical stability of the computations (cf. Schönhage (1982)). Alternatively one can scale the variable $x \rightarrow y = ax$ for a fixed scalar a to bring the zero set of the polynomial p in (2.1) into the unit disc $D(0,1) = \{y : \ |y| \leq 1\}$, and then one would seek only the factors $y - az_j$ of of the resulting polynomial $q(y) = p(x)$ for $x = y/a$.

References

[1] Barnett, S. (1983), *Polynomial and Linear Control Systems*, Marcel Dekker, New York.

[2] Bini, D. (1996), Numerical Computation of Polynomial Zeros by Means of Aberth's Method, *Numerical Algorithms* **13**, 179–200

[3] Bini, D. A. and Boito, P. (2010), A Fast Algorithm for Approximate Polynomial GCD Based on Structured Matrix Computations, *Operator Theory: Advances and Applications* **199**, 155–173, Birkhäuser Verlag

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

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

[6] Börsch-Supan, W. (1963), A-posteriori Error Bounds for the Zeros of Polynomials, *Numerische Math.* **5**, 380–398

[7] Box, G. E. P. and Jenkins, G. M. (1976), *Time Series Analysis: Forecasting and Control*, Holden-Day, San Francisco, CA

[8] Corless, R. M., Gianni, P. M., Trager, B. M. and Watt, S. M. (1995), The Singular Value Decomposition for Polynomail Systems, *Proc. Intern. Symposium on Symbolic and Algebriac Computation (ISSAC'95)*, 195–207, ACM Press, New York

[9] Demeure, C. J. and Mullis, C. T. (1989), The Euclid algorithm and the fast computation of cross–covariance and autocovariance sequences, *IEEE Trans. Acoust., Speech, Signal Processing* **37**, 545–552

[10] Demeure, C. J. and Mullis, C. T. (1990), A Newton–Raphson method for moving-average spectral factorization using the Euclid algorithm, *IEEE Trans. Acoust., Speech, Signal Processing* **38**, 1697–1709

[11] Gohberg, I. and Semencul, A. (1972), On the Inversion of Finite Toeplitz Matrices and Their Continuous Analogs, *Matematicheskiie Issledovaniia* (in Russian) **7**, **2**, 187–224

[12] Habbard, J., Schleicher, D. and Sutherland, S. (2001), How to Find All Roots of Complex Polynomials by Newton's Method, *Invent. Math.* **146**, 1–33

[13] Kirrinnis, P. (1998), Polynomial factorization and partial fraction decomposition by simultaneous Newton's iteration, *J. of Complexity* **14**, 378–444

[14] McNamee, J. M. (1993), Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics* **47**, 391–394

[15] McNamee, J. M. (1997), A Supplementary Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics* **78**, 1

[16] McNamee, J. M. (2002), A 2002 update of the supplementary bibliography on roots of polynomials, *J. Computational and Applied Mathematics* **142**, 433–434

[17] McNamee, J. M. (2007), *Numerical Methods for Roots of Polynomials*, Elsevier

[18] Pan, V. Y. (1995), 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

[19] Pan, V. Y. (1996), Optimal and nearly optimal algorithms for approximating polynomial zeros, *Computers and Math. (with Applications)* **31, 12**, 97–138

[20] Pan, V. Y. (2001), Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *Proc. International Symp. on Symbolic and Algebraic Computation (ISSAC '01)*, 253–267, ACM Press, New York

[21] Pan, V. Y. (2001a), *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York

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

[23] Pan, V. Y. and Zheng, A. (2011), New Progress in Real and Complex Polynomial Root-Finding, *Computers and Math. (with Applications)* **61**, 1305–1334

[24] Schönhage, A. (1982), The fundamental theorem of algebra in terms of computational complexity, *Dept. of Math., University of Tübingen*, Tübingen, Germany

[25] Van Dooren, P. M. (1994), Some numerical challenges in control theory, *Linear Algebra for Control Theory, IMA Vol. Math. Appl.* **62**

[26] Weierstrass, K. (1903), Neuer Beweis des Fundamentalsatzes der Algebra, *Mathematische Werker* **Tome III**, 251–269, Mayer und Mueller, Berlin

[27] Wilson, G. T. (1969), Factorization of the Covariance Generating Function of a Pure Moving-average Process, *SIAM J. on Numerical Analysis* **6**, 1–7

[28] Z. Zeng (2005), Computing multiple roots of inexact polynomials, *Mathematics of Computation* **74**, 869–903