Root-finding with Implicit Deflation

Rémi Imbach[1],[a], Victor Y. Pan [2,3],[b], Chee Yap[1],[c],
Ilias S. Kotsireas[4],[d] and Vitaly Zaderman[3],[e]

[1] Courant Institute of Mathematical Sciences
New York University,USA
[2] Department of Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA and
[3] Ph.D. Programs in Mathematics and Computer Science
The Graduate Center of the City University of New York
New York, NY 10036 USA
[4] Wilfrid Laurier University
Department of Physics and Computer Science
75 University Avenue West
Waterloo, Ontario N2L 3C5 CANADA
[a] remi.imbach@nyu.edu
[b] victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/
[c] Email: yap@cs.nyu.edu www.cs.nyu.edu/yap/
[d] ikotsire@wlu.ca
http://web.wlu.ca/science/physcomp/ikotsireas/
[e] vza52@aol.com

Abstract

Functional iterations such as Newton's are a popular tool for polynomial root-finding. We consider a realistic situation where some roots have already been approximated (we say tamed), and one would like to restrict further root-finding to the approximation of the remaining (wild) roots. A natural approach of applying explicit deflation has been much studied and recently advanced by one of the authors of this paper, but presently we consider the alternative of implicit deflation combined with mapping of the variable and reversion of an input polynomial. The hope is that the union of the sets of tame roots approximated in a number of such transformations can cover all roots of a polynomial.

We also show another direction to substantial further progress in this

long and extensively studied area. Namely we dramatically increase the local efficiency of root-finding by means of the incorporation of fast algorithms for multipoint polynomial evaluation and the Fast Multipole Method.

Key Words: Polynomial roots; Functional iterations; Newton's iterations; Weierstrass's iterations; Ehrlich's iterations; Deflation; Taming wild roots; Maps of the variable; Efficiency; Multipoint evaluation; Fast Multipole Method

2000 Math. Subject Classification: 26C10, 30C15, 65H05

1 Introduction

Univariate polynomial root-finding, that is, approximation of the roots $x_1, \ldots, x_d$ of a polynomial equation

$$p(x) = 0 \text{ for } p(x) = \sum_{j=0}^{d} p_j x^j = p_d \prod_{i=1}^{d} (x - x_i), \quad p_d \neq 0, \tag{1}$$

has been the central problem of Mathematics for four millennia, since the Sumerian times. It is still involved in various areas of modern computation and is the subject of intensive research worldwide. The user's choice since 2000 has been the package MPSolve (cf. [7], [12]), which implements Ehrlich's functional iterations, but other functional iterations such as Newton's and Weierstrass's are also highly popular. Ehrlich's and Weierstrass's iterations converge simultaneously to all complex roots of a polynomial. Newton's iterations converge to a single root but can be extended to approximation of all roots or the roots in a fixed domain.

Usually root-finding iterations approximate (we say *tame*) most of the roots, and then one can deflate an input polynomial and keep updating only the approximations to the remaining, *wild* roots.

Efficient methods for explicit deflation can be found in [53] and references therein, but here we study alternative techniques of implicit deflation, which enable us to exploit the sparseness of an input and to avoid numerical stability problems caused by the coefficient growth in factorization of a polynomial.

The partition of the root set into tame and wild roots for a fixed root-finder depends on the roots disposition relatively to the initial root approximations. So the partition should change if we change the initial approximations or if we map the variable, apply the same root-finder to the new resulting polynomial, and then recover the roots of the original polynomial by applying the converse map. By using implicit deflation we avoid computing the new coefficients in these mappings (see Sections 5 and 6). We conjecture that already the union of a small number of the resulting variations of the set of tame roots would include all roots of p, and we can strengthen our chances for success of this heuristic approach by applying it concurrently using various functional iterations.

In Section 8 we point out another promising direction to enhancing the power of root-finding iterations, namely by means of incorporation of superfast multipoint polynomial evaluation and the Fast Multipole Method. We demonstrate the high promise of this approach by showing that it yields a dramatic increase of local efficiency of root-finding iterations.

Otherwise we organize our paper as follows. In the next section we recall some popular functional iterations for polynomial root-finding. In Section 3 we comment on partitioning polynomial roots into tame ones (already approximated) and wild ones. In Section 4 we compare explicit and implicit deflation and specify implicit deflation for Newton's iterations. We combine implicit deflation with linear maps of the variable and reversion of a polynomial in Section 5 and with squaring the variable in Section 6, followed by our comments on potential benefits of concurrent root-finding in Section 7.

2 Functional Iterations for Root-Finding

Among hundreds if not thousands known polynomial root-finders (see up to date coverage in [38], [41], [53], and the bibliography therein) consider the class of functional iterations. For a fixed set of functions

$$f_1(z), \dots, f_m(x),\ 1 \le m \le d,$$

these iterations recursively refine current approximations $z_1^{(k)}, \dots, z_m^{(k)}$ to m roots $x_1, \dots x_m$ of $p(x)$ according to the expressions

$$z_i \leftarrow f_i(z_i),\ i = 1, \dots, m. \tag{2}$$

In the case where $m = 1$ write $f(z) = f_1(z)$ and

$$z \leftarrow f(z). \tag{3}$$

These iterations include various interpolation methods, which use no derivatives of $p(x)$ and are recalled in [41, Section 7], for example, Muller's method (see [41, Section 7.4]); methods involving derivative such as Newton's iterations [38, Section 5]; and methods involving higher order derivatives [41, Section 7]. We exemplify our study with Newton's iterations (where $m = 1$):

$$z \leftarrow z - N_p(z), \tag{4}$$

$$N_p(x) = p(x)/p'(x), \tag{5}$$

which have efficient extensions to the solution of polynomial systems of equations [6] and to root-finding for various smooth functional equations and systems of equations [23]; Weierstrass's iterations of [67] (rediscovered by Durand in [17] and Kerner in [33]), in which case $m = d$:

$$z_i \leftarrow z_i - W_{p,l}(z_i),\ i = 1, \dots, d, \tag{6}$$

$$W_{p,l}(x) = \frac{p(x)}{p_n l'(x)}, \tag{7}$$

$$l(x) = \prod_{i=1}^{d}(x - z_i), \tag{8}$$

and Ehrlich's iterations of [22] (rediscovered by Aberth in [1]), where again $m = d$:

$$z_i \leftarrow z_i - E_{p,i}(z_i), \tag{9}$$

$$E_{p,i}(x) = 0 \text{ if } p(x) = 0;\ \frac{1}{E_{p,i}(x)} = \frac{1}{N_p(x)} - \sum_{j=1, j\neq i}^{d} \frac{1}{x - z_j} \text{ otherwise;} \tag{10}$$

$i = 1, \ldots, d$, and $N_p(x)$ is defined by (5).

Remark 1. The above root-finders are readily extended to any function $s(x)$ sharing its root set with the polynomial $p(x)$. For example, deduce from the Lagrange interpolation formula that

$$p(x) = l(x)s(x),$$

$$s(x) = p_n + \sum_{i=1}^{d} \frac{W_{p,l}(z_i)}{x - z_i}$$

for any set of d distinct nodes $z_1, \ldots, z_d$. Apply selected iterations to the above *secular rational function* $s(x)$ or the polynomial $l(x)s(x)$. Bini and Robol in [12] show substantial benefits of that application of Ehrlich's iterations to $l(x)s(x)$ rather than $p(x)$, both for convergence acceleration and error estimation.

3 Tame and Wild Roots

Now suppose that we have applied a fixed functional iteration (2) and have approximated m roots of a polynomial $p(x)$ for $m < d$ (we call them *tame*); next we discuss efficient approximation of the remaining roots; we call them *wild* and call their approximation *taming*.

For example, we face a taming problem where functional iterations (3) have approximated a single root of a polynomial $p(x)$ and we seek the other roots.

Newton's and other iterations (3), devised for approximation of a single root, can be also applied at a number of initial points in order to approximate all roots. This can succeed for most of the roots, while some roots can escape and stay wild. In particular in the paper [64] Newton's iterations initialized at a universal set of $O(d)$ points[1] approximate $t = d - w$ roots of $p(x)$ but leave

[1]This set is *universal* for all polynomials $p(x)$ that have all roots lying in the unit disc $D(0, 1) = \{z :\ |z| = 1\}$. Given any polynomial $p(x)$ one can move all its roots into this disc by means of first readily computing a reasonably close upper bound on the absolute values of all roots and then properly shifting and scaling the variable x.

out a narrow set of w wild roots where $w < 0.001\ d$ for $d < 2^{17}$ and $w < 0.01\ d$ for $d < 2^{20}$. The paper [64] continued a long study traced back to [35] and [30].

Likewise some roots remain wild while most of the roots are tamed in Weierstrass's, Ehrlich's, and various other iterations that recursively update approximations of all roots.

Finally the subdivision root-finding iterations of [14] extend the earlier study in [68], [29], [28], [59], and [47], where such iterations are called the Quad-tree construction. This root-finder has recently been implemented in [31]. It first approximates some sets of tame roots of $p(x)$ in certain domains on the complex plane well-isolated from the other roots and then approximates the remaining wild roots, in particular by combining the subdivision process with complex extension of Abbott's real QIR iterations.

4 Taming Wild Roots by Means of Deflation

Seeking wild roots one can deflate an input polynomial, that is, apply a selected root-finder to the polynomial

$$q(x) = \sum_{i=0}^{w} q_i x^i = p_d \prod_{j=1}^{w} (x - x_j), \quad p_d \neq 0. \tag{11}$$

In *explicit deflation* we first compute the coefficients of $q(x)$. If the roots of the quotient $q(x)$ are well isolated from the other roots of $p(x)$, we can apply the efficient method of Delves and Lyness [19]. The root-finders of [60] and [34] incorporate its advanced versions; [53] presents them in a concise form.

Bini and Fiorentino argue in [7] that explicit deflation of a polynomial $p(x)$ does not preserve its sparseness and in some cases can be numerically unstable, for instance, in the case of a polynomial $p(x) = x^d \pm 1$ of a large degree d. These potential problems somewhat limit the value of explicit deflation where a polynomial $q(x)$ has large degree w. Moreover we can completely avoid these problems by applying *implicit deflation*, that is, applying functional iterations that evaluate $q(x)$ at a point x as the ratio $\frac{p(x)}{t(x)}$ for $t(x) = p_d \prod_{j=1+w}^{d} (x - x_j)$.

We can readily implement this recipe in the case of functional interpolation iterations of [41, Section 7].

Let us specify implicit deflation when we apply Newton's iterations and the following well-known identity (cf. [37]),

$$\frac{1}{N_p(x)} = \sum_{j=1}^{n} \frac{1}{x - x_j}. \tag{12}$$

Algorithm 2. *Implicit Deflation with Newton's iterations.*

INPUT: A polynomial $p(x)$ of (1), a set of sufficiently close approximations[2] to its tame roots $x_{w+1}, \dots, x_d$, an initial approximation z to a wild root of

[2]We assume that we can very quickly refine approximations to tame roots.

$p(x)$, a Stopping Criterion (see, e.g., [7], [12]), and a black-box program EVAL_p that evaluates the ratio $\frac{1}{N_p(z)} = \frac{p'(z)}{p(z)}$ for a polynomial $p(x)$ of (1) and a complex point z.

OUTPUT: The updated approximation $z \leftarrow z - N_p(z)$ to a root of $p(x)$ (see (4)).

COMPUTATIONS: Apply Newton's iteration (4) to the polynomial $q(x)$ defined implicitly, that is, successively compute the values:

1. $r = p'(z)/p(z) \leftarrow 1/N_p(z)$,
2. $s \leftarrow \sum_{j=w+1}^{d} \frac{1}{z-x_j}$,
3. $N_q(z) = \frac{q(z)}{q'(z)} \leftarrow \frac{1}{r-s}$.
4. $z \leftarrow z - N_p(z)$.
5. If the fixed Stopping Criterion is met, output z and stop. Otherwise go to stage 1.

Dario A. Bini (private communication) proposed to improve numerical stability of this algorithm by means of scaling as follows:

$$N_q(z) = \frac{1/r}{1 - s/r}.$$

Complexity of a single iteration of Algorithm 2.
Stage 1 amounts to a single invocation of the program EVAL_p.
Stage 2 involves $d-w$ divisions, $d-w$, subtractions and $d-w-1$ additions.
Stages 3 and 4 together involve 2 subtractions and a single division.

We can readily extend implicit deflation to various other root-finders involving Newton's ratio $N_p(x)$, in particular, to Ehrlich's iterations (9) because we can assume that $z_j = x_j$ for $j > w$, and then equation (12) implies that $E_{p,j}(x) = E_{q,j}(x)$ for $q(x)$ of (11), $E_{p,j}(x)$ of (10), and $j \le w$.

5 Combining Newton's Iterations, Linear Maps of the Variable and Reversion

The set of tame roots output by fixed functional iterations varies when an input polynomial $p(x)$ varies. This suggests that we can approximate many or all wild roots if we reapply the same iterations to the polynomials

$$v(z) = v_{a,b,c}(z) = (z+c)^d p\Big(a + \frac{b}{z+c}\Big) \tag{13}$$

for various triples of complex scalars a, $b \neq 0$, and c. We must limit the overall number of the triples in order to control the overall computational cost.

The following equations map the roots x_j of $p(x)$ to the roots z_j of $v(x)$ and vice versa,

$$x_j = a + \frac{b}{z_j + c},\ z_j = \frac{b}{x_j - a} - c. \tag{14}$$

Let us specify this recipe for the algorithm of [64], cited in Section 3.

Algorithm 3.

INITIALIZATION: Define a polynomial $v(z) = v_{a,b,c}(z)$ by choosing the parameters a, b, and c such that all roots of the polynomial $v(z)$ lie in the unit disc $D(0,1) = \{z :\ |z| = 1\}$; do not actually compute the coefficients of that polynomial.

COMPUTATIONS: 1. Apply Newton's iteration (4) to the polynomial $v(z)$ by using initialization at the universal set of [64] and by expressing the Newton's ratios $N_v(z) = v(z)/v'(z)$ (cf. (4)) via the following equations:

$$\frac{1}{N_v(z)} = \frac{d}{z+c} - \frac{b}{(z+c)^2 N(x)} \text{ for } v(z) \text{ of (13) and } x \text{ of (14).} \tag{15}$$

2. Having approximated a root z_j of $v(z)$ for any j, readily recover the root x_j of $p(x)$ from equation (14).

In the particular case where $a = c = 0$ and $b = 1$, the above expressions are simplified: $z = 1/x$; $v(z)$ turns into the reverse polynomial of $p(x)$,

$$v(z) = p_{\rm rev}(z) = \sum_{i=0}^{d} p_{d-i} z^i = z^d p(1/z),$$

$$\frac{1}{N_v(z)} = \frac{v'(z)}{v(z)} = \frac{d}{z} - \frac{1}{z^2 N_p(1/z)},$$

and $p_{\rm rev}(x) = p_0 \prod_{j=1}^{d}(x - 1/x_j)$ if $p_0 \neq 0$.

6 Combining Newton's Iterations and Squaring of the Variable

One can hope to obtain all roots of $p(x)$ by applying Newton's iterations to the polynomials $v(z) = v_{a,b,c}(z)$ for a reasonable number of triples of a, b, and c, but one can also extend this approach by using more general rational maps $y = r(x)$ (cf., e.g., [40]).

For a simple example, consider the Dandelin's root-squaring map of 1826, rediscovered by Lobachevsky in 1834 and then by Gräffe in 1837 (see [27]):

$$u(y) = (-1)^d p(\sqrt{y}) p(-\sqrt{y}) = \prod_{j=1}^{d} (y - x_j^2). \tag{16}$$

In this case one should make a polynomial $p(x)$ of (1) monic by scaling the variable x and then express the Newton's ratio $N_u(y) = u(y)/u'(y)$ as follows:

$$\frac{1}{N_u(y)} = 0.5\Big(\frac{1}{N_p(\sqrt{y})} - \frac{1}{N_p(\sqrt{-y})}\Big)\ y^{-1/2}.$$

Notice that under map (16) the roots lying in the unit disc $D(0,1)$ stay in it.

Having approximated the n roots $y_1, \dots, y_n$ of the polynomial $u(y)$, we readily recover the n roots $x_1, \dots, x_n$ of the polynomial $p(x)$ by selecting them from the $2n$ values $\pm\sqrt{y_j}$, $j = 1, \dots, n$.

We can apply the above maps recursively (a limited number of times, in order to control the overall computational cost); then we can recover the roots from their images in these rational maps by extending the lifting/descending techniques of [44], [49].

7 Concurrent Root-finding

Remark 4. We recalled that Newton's iterations can compute most of the roots of a fixed polynomial but not all of them. Seeking the remaining, wild roots, we applied the iterations to a number of related polynomial. This recipe can be immediately extended to application of Ehrlich's, Weierstrass's or another fixed iterative root-finder to a variety of polynomials linked to an input polynomial. Furthermore we can extend this idea to concurrent application of a number of iterative root-finders to such a variety of polynomials, and one can perform computations on a number of processors with minimal need for their communication and synchronization.

Remark 5. Weierstrass's and Ehrlich's functional iterations, as well as their Gauss-Seidel's and Werner's accelerated variations (cf. [12] and [70]) converge very fast empirically, but formal support of this empirical observation is a well-known challenge. Can we facilitate obtaining such a support if we allow random maps of the variable x, e.g., if we apply these iterations to the polynomials $v_{a,b,c}(z)$ of (13) for random choice of the parameters a, b, and c? For example, initialization of Newton's iterations at a set of points $\{c+r\exp(\phi_j\mathbf{i}),\ j = 1, \dots, s,$ of a circle $\{x :\ |x - c| = r\}$ on the complex plane can be equivalently interpreted as the application of these iterations at a single point $y = c$ to a set of polynomials $p_j(y)$ obtained from $p(x)$ via the linear maps $y \leftarrow x - r\exp(\phi_j\mathbf{i})$, $j = 1, \dots, s$.

8 Efficiency of Root-finding Iterations

Since Ostrowski's paper [43], it is customary to measure local efficiency of functional root-finding iterations by the quantity eff$= q^{1/\alpha}$ or sometimes $\log_{10}(\text{eff}) = (1/\alpha)\log_{10} q$ where q denotes the convergence order (rate) and α is the number of function evaluations per iteration and per root. In particular $q = 2$, $\alpha = 2$, and eff$=\sqrt{2} \approx 1.414$ for Newton's and Weierstrass's iterations while $q = 3$,

$\alpha = 3$, and eff=$3^{1/3} \approx 1.442$ for Ehrlich's iterations where we assign the same cost to the evaluation of the functions $\sum_{j=1,j\neq i}^{d} \frac{1}{x-z_j}$, $p(x)$, $p'(x)$, and $l'(x)$ at $x = z_i$, noting that $l'(z_i) = \prod_{j=1,j\neq i}^{d}(z_i - z_j)$.

Actually the cost of function evaluation requires further elaboration. Exact evaluation of the values $\sum_{i=1,i\neq j}^{d} \frac{1}{z_j^{(k)}-z_i^{(k)}}$ for $j = 1, \dots, d$ is Trummer's celebrated problem, whose solution, like exact evaluation of a polynomial $p(x)$ of (1) at d points, involves $O(d\log^2(d))$ arithmetic operations [48, Section 3.1], [25], [39].

Both of these superfast algorithms – for polynomial evaluation and the Trummer's problem – are numerically unstable for $d > 50$, but one can use numerically stable superfast alternatives based on the Fast Multipole Method [16]. Its application to Trummer's problem is well-known [26], but in the case of multipoint polynomial evaluation is more recent and more involved [50] and [52].

Using superfast algorithms for both problems decreases α to the order of $O(\log^2(d)/d)$. Hence local efficiency of Weierstrass's and Ehrlich's iterations grows to infinity as $d \to \infty$, and similarly for Newton's iterations initialized and applied simultaneously at the order of d points.

The above formal analysis applies locally, where the convergence to the roots becomes superlinear, while the overall computational cost is usually dominant at the previous initial stage, for which only limited formal results are available (see also Remark 5). These limited results favor Ehrlich's iterations, which empirically have milder sufficient conditions for superlinear convergence than both Newton's and Weierstrass's iterations [65].

Acknowledgements: The research of R. Imbach, V. Y. Pan, V. Zaderman and C. Yap was supported by NSF Grant CCF-1563942. The research of R. Imbach was also supported by NSF Grants CCF-1564132 and CCF-1708884. The research of V. Y. Pan and V. Zaderman was also supported by NSF Grant CCF 1116736 and PSC CUNY Award 69813 00 48. The research of Ilias Kotsireas was supported by an NSERC grant.

References

[1] Aberth, O.: Iteration Methods for Finding All Zeros of a Polynomial Simultaneously, Mathematics of Computation **27, 122**, 339–344 (1973) doi: 10.1090/S0025-5718-1973-0329236-7

[2] Bell, E.T.: The Development of Mathematics. McGraw-Hill, New York (1940) doi: 10.2307/2268176

[3] Boyer, C.A.: A History of Mathematics. Wiley, New York (1991) doi: 10.1177/027046769201200316

[4] Barnett, S.: Polynomial and Linear Control Systems. Marcel Dekker, New York (1983) doi: 10.1112/blms/16.3.331

[5] Bini, D.A.: Parallel Solution of Certain Toeplitz Linear Systems. SIAM J. Comput. **13, 2**, 268–279 (1984) doi: 10.1137/0213019

[6] Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer (1998). doi: 10.1007/978-1-4612-0701-6

[7] Bini, D.A., Fiorentino, G.: Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder. Numer. Algorithms **23**, 127–173 (2000) doi: 10.1023/A:1019199917103

[8] Bini, D.A., Gemignani, L., Pan, V.Y.: Inverse Power and Durand/Kerner Iteration for Univariate Polynomial Root-finding. Comput. Math. Appl. **47**, **2/3**, 447–459 (2004) doi: 10.1016/S0898-1221(04)90037-5

[9] Box, G.E.P., Jenkins, G.M.: Time Series Analysis: Forecasting and Control. Holden-Day, San Francisco, California (2015, fifth edition) doi: 10.1111/jtsa.12194

[10] Bini, D., Pan, V.Y.: Computing Matrix Eigenvalues and Polynomial Zeros Where the Output Is Real. SIAM J. Comput. **27**, **4**, 1099–1115 (1998). Proc. version in SODA'91, 384–393. ACM Press, NY, and SIAM Publ., Philadelphia (1991) doi: 10.1137/S0097539790182482

[11] Bini, D., Pan, V.Y.: Graeffe's, Chebyshev, and Cardinal's Processes for Splitting a Polynomial into Factors. J. Complex. **12**, 492–511 (1996) doi: 10.1006/jcom.1996.0030

[12] Bini, D.A., Robol, L.: Solving Secular and Polynomial Equations: a Multiprecision Algorithm. J. Comput Appl Math **272**, 276–292 (2014) doi: 10.1016/j.cam.2013.04.037

[13] Becker, R., Sagraloff, M., Sharma, V., Xu, J., Yap, C.: Complexity Analysis of Root Clustering for a Complex Polynomial. In: International Symposium on Symbolic and Algebraic Computation (ISSAC 2016), 71–78 (2016) doi: 10.1145/2930889.2930939

[14] Becker, R., Sagraloff, M., Sharma, V., Yap, C.: A Near-Optimal Subdivision Algorithm for Complex Root Isolation based on the Pellet Test and Newton Iteration. J. Symb Comput **86**, 51–96 (2018) doi: 10.1016/j.jsc.2017.03.009

[15] Ben-Or, M., Tiwari, P.: Simple algorithms for approximating all roots of a polynomial with real roots. J. Complexity **6(4)**, 417–442 (1990) doi: 10.1016/0885-064X(90)90032-9

[16] Barba, L. A., Yokota, R.: How Will the Fast Multipole Method Fare in Exascale Era? SIAM News, **46, 6**, 1–3, July/August (2013)

[17] Durand, E.: Solutions numériques des équations algébriques, *Tome 1: Equations du type F(X)=0; Racines d'un polynôme*. Masson, Paris (1960)

[18] Du, Q., Jin, M., Li, T.Y., Zeng, Z.: The quasi-Laguerre iteration. Math. Comput. **66(217)**, 345–361 (1997) doi: 10.1090/S0025-5718-97-00786-2

[19] Delves, L.M., Lyness, J.N.: A Numerical Method for Locating the Zeros of an Analytic Function. Math. Comput. **21**, 543–560 (1967). doi: 10.1090/S0025-5718-1967-0228165-4

[20] Demeure, C.J., Mullis, C.T.: The Euclid Algorithm and the Fast Computation of Cross–covariance and Autocovariance Sequences. IEEE Trans. Acoust, Speech, Signal Process. **37**, 545–552 (1989) doi: 10.1109/29.17535

[21] Demeure, C.J., Mullis, C.T.: A Newton–Raphson Method for Moving-average Spectral Factorization Using the Euclid Algorithm. IEEE Trans. Acoust, Speech, Signal Processing **38**, 1697–1709 (1990) doi: 10.1109/29.60101

[22] Ehrlich, L.W.: A Modified Newton Method for Polynomials. Commun ACM **10**, 107–108 (1967) doi: 10.1145/363067.363115

[23] Encyclopedia of Mathematics. Newton method (Hazewinkel, Michiel, ed.). Springer Science+Business Media B.V. Kluwer Academic Publishers (1994, first edition) doi: 10.1007/978-94-009-5991-0 , (2000, second edition) doi: 10.1007/978-94-015-1279-4

[24] Emiris, I.Z., Pan, V.Y., Tsigaridas, E.: Algebraic Algorithms. In: Tucker, A.B., Gonzales, T., Diaz-Herrera, J.L. (eds) Computing Handbook (3rd edition), Vol. I: Computer Science and Software Engineering, Ch. 10, pp. 10-1 - 10-40. Taylor and Francis Group (2014)

[25] Gerasoulis, A., Grigoriadis, M. D., Sun, L.: A Fast Algorithm for Trummer's Problem. SIAM Journal on Scientific and Statistical Computing **8, 1**, 135–138 (1987) doi: 10.1137/0908017

[26] Greengard, L., Rokhlin, V.: A Fast Algorithm for Particle Simulation. Journal of Computational Physics **73**, 325–348 (1987) doi: 10.1016/0021-9991(87)90140-9

[27] Householder, A.S.: Dandelin, Lobachevskii, or Graeffe? Amer. Math. Monthly **66**, 464–466. (1959) doi: 10.2307/2310626

[28] Henrici, P.: Applied and Computational Complex Analysis. Vol. 1: Power Series, Integration, Conformal Mapping, Location of Zeros. Wiley (1974)

[29] Henrici, P., Gargantini, I.: Uniformly Convergent Algorithms for the Simultaneous Approximation of All Zeros of a Polynomial. In: Dejon, B., Henrici, P. (eds) Constructive Aspects of the Fundamental Theorem of Algebra. Wiley (1969)

[30] Habbard, J., Schleicher, D., Sutherland, S.: How to Find All Roots of Complex Polynomials by Newton's Method. Invent. Math. **146**, 1–33 (2001) doi: 10.1007/s002220100149

[31] Imbach, R., Pan, V.Y., Yap, C.: Implementation of a Near-Optimal Complex Root Clustering Algorithm. In: Proc. of International Congress on Math Software (ICMS 2018), 235–244 (2018) doi: 10.1007/978-3-319-96418-8 28

[32] Imbach, R., Pan, V.Y., Yap, C., Kotsireas, I.S., Zaderman, V.: Root-finding with Implicit Deflation. In: Proc. of CASC 2019 (to appear). arXiv:1606.01396, submitted on 21 May (2019)

[33] Kerner, I. O.: Ein Gesamtschrittverfahren zur Berechung der Nullstellen von Polynomen. Numerische Math. **8**, 290–294 (1966) doi: 10.1007/BF0216256

[34] Kirrinnis, P.: Polynomial Factorization and Partial Fraction Decomposition by Simultaneous Newton's Iteration. In: J. Complex. **14**, 378–444 (1998) doi: 10.1006/jcom.1998.0481

[35] Kim, M.-H., Sutherland, S.: Polynomial Root-Finding Algorithms and Branched Covers. SIAM Journal on Computing **23, 2**, 415–436 (1994) doi: 10.1137/S0097539791201587

[36] Kobel, A., Rouillier, F., Sagraloff, M.: Computing Real Roots of Real Polynomials ... and Now for Real! In: Intern. Symp. Symb. Algebraic Computation (ISSAC 2016), 301 - 310. ACM Press, New York (2016) doi: 10.1145/2930889.2930937

[37] Mahley, H.: Zur Auflösung Algebraisher Gleichngen, Z. Andew. Math. Physik. **5**, 260 – 263 (1954)

[38] McNamee, J.M.: Numerical Methods for Roots of Polynomials, Part I, XIX+354 pages. Elsevier (2007)

[39] Moenck, R., Borodin, A.: Fast Modular Transforms via Division., In: Proc. 13th Annual Symposium on Switching and Automata Theory (SWAT 1972), 90–96, IEEE Computer Society Press (1972) doi: 10.1109/SWAT.1972.5.

[40] Mourrain, B., Pan, V.Y.: Lifting/Descending Processes for Polynomial Zeros and Applications. J. of Complexity **16, 1**, 265 – 273 (2000) doi: 10.1006/jcom.1999.0533

[41] McNamee, J.M., Pan, V.Y.: Numerical Methods for Roots of Polynomials, Part II, XXI+728 pages. Elsevier (2013)

[42] Neff, C.A., Reif, J.H.: An $o(n^{1+\epsilon})$ Algorithm for the Complex Root Problem. In: Proc. of 35th Ann. IEEE Symp. on Foundations of Computer Science (FOCS '94), 540–547. IEEE Computer Society Press (1994) doi: 10.1109/SFCS.1994.365737

[43] Ostrowski, A. M.: Solution of Equations and Systems of Equations. Academic Press, New York (1966) doi: 10.1017/S0008439500029805

[44] Pan, V.Y.: Optimal (up to Polylog Factors) Sequential and Parallel Algorithms for Approximating Complex Polynomial Zeros. In: Proc. 27th Ann. ACM Symp. on Theory of Computing (STOC'95), 741–750. ACM Press, New York (1995) doi: 10.1145/225058.225292

[45] Pan, V.Y.: Solving a Polynomial Equation: Some History and Recent Progress. SIAM Rev **39, 2**, 187–220 (1997) doi: 10.1137/S0036144595288554

[46] Pan, V.Y.: Solving Polynomials with Computers. Am Sci **86**, 62–69. January–February (1998) doi: 10.1511/1998.1.62

[47] Pan, V.Y.: Approximation of Complex Polynomial Zeros: Modified Quadtree (Weyl's) Construction and Improved Newton's Iteration. J.Complex. **16, 1**, 213–264 (2000) doi: 10.1006/jcom.1999.0532

[48] Pan, V.Y.: Structured Matrices and Polynomials: Unified Superfast Algorithms. Birkhäuser/Springer, Boston/New York, (2001) doi: 10.1007/978-1-4612-0129-8

[49] Pan, V.Y.: Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding. J. Symb Comput **33, 5**, 701–733 (2002) doi: 10.1006/jsco.2002.0531

[50] Pan, V.Y.: Transformations of Matrix Structures Work Again. Linear Algebra and Its Applications **465**, 1 – 32 (2015) doi: 10.1016/j.laa.2014.09.004

[51] Pan, V.Y.: Root-finding with Implicit Deflation. arXiv:1606.01396, submitted on 4 June (2016)

[52] Pan, V.Y.: Simple and Nearly Optimal Polynomial Root-Finding by Means of Root Radii Approximation. In: Kotsireas I.S., Martinez-Moro, E. (eds) Springer Proceedings in Mathematics and Statistics, Ch. 23: Applications of Computer Algebra **198** AG. Springer International Publishing (2017).
Chapter DOI.10.1007/978-3-319-56932-1 23

[53] Pan, V.Y.: Old and New Nearly Optimal Polynomial Root-finders. In: CASC (2019) (to appear) Also arxiv: 1805.12042 [cs.NA] May (2019)

[54] Pan, V.Y., Sadikou, A., Landowne, E.: Univariate Polynomial Division with a Remainder by Means of Evaluation and Interpolation. In: Proc. of 3rd IEEE Symposium on Parallel and Distributed Processing, 212–217. IEEE Computer Society Press, Los Alamitos, California (1991) doi: 10.1109/SPDP.1991.218277

[55] Pan, V.Y., Sadikou, A., Landowne, E.: Polynomial Division with a Remainder by Means of Evaluation and Interpolation. Inform Process Lett **44**, 149–153 (1992) doi: 10.1016/0020-0190(92)90055-Z

[56] Pan, V.Y., Tsigaridas, E.P.: Nearly Optimal Refinement of Real Roots of a Univariate Polynomial. J. Symb Comput **74**, 181–204 (2016) doi: 10.1016/j.jsc.2015.06.009. Also in: Kauers, M. (ed) Proc. of ISSAC 2013, pp. 299–306. ACM Press, New York (2013)

[57] Pan, V.Y., Tsigaridas, E.P.: Nearly Optimal Computations with Structured Matrices. In: Theor. Comput. Sci., Watt, S., Verschelde, J., Zhi, L. (eds) Special Issue on Symbolic–Numerical Algorithms **681**, 117–137 (2017).
doi: 10.1016/j.tcs.2017.03.030.

[58] Pan, V.Y., Zhao, L.: Real Polynomial Root-finding by Means of Matrix and Polynomial Iterations. In: Theor. Comput. Sci., Watt, S., Verschelde, J., Zhi, L. (eds) Special Issue on Symbolic–Numerical Algorithms **681**, 101–116 (2017).
doi: 10.1016/j.tcs.2017.03.032.

[59] Renegar, J.: On the Worst-case Arithmetic Complexity of Approximating Zeros of Polynomials. J. Complex. **3, 2**, 90–113 (1987) doi: 10.1016/0885-064X(87)90022-7

[60] A. Schönhage. The Fundamental Theorem of Algebra in Terms of Computational Complexity. Tech. Report, Math. Dept., University of Tübingen, Tübingen, Germany, (1982)

[61] Schönhage, A.: Asymptotically Fast Algorithms for the Numerical Muitiplication and Division of Polynomials with Complex Coefficients. In: Proc. of European Computer Algebra Conference (EUROCAM 1982), 3–15. Computer Algebra, (1982) doi: 10.1007/3-540-11607-9 1

[62] Schönhage, A.: Quasi GCD Computations. J. Complex. **1**, 118–137 (1985) doi: 10.1016/0885-064X(85)90024-X

[63] Schleicher, D.: private communication.

[64] Schleicher, D., Stoll, R.: Newton's Method in Practice: Finding All Roots of Polynomials of Degree One Million Efficiently. Theor. Comput. Sci. **681**, 146–166 (2017) doi: 10.1016/j.tcs.2017.03.025

[65] Tilli, P.: Convergence Conditions of Some Methods for the Simultaneous Computation of Polynomial Zeros. Calcolo **35**, 3–15 (1998) doi: 10.1007/s100920050005

[66] Van Dooren, P..M.: Some Numerical Challenges in Control Theory. Linear Algebra for Control Theory, IMA Vol. Math. Appl. **62**, 177 - 189 (1994) doi: 10.1007/978-1-4613-8419-9 12

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

[68] Weyl, H.: Randbemerkungen zu Hauptproblemen der Mathematik. II. Fundamentalsatz der Algebra and Grundlagen der Mathematik. Mathematische Zeitschrift, **20**, 131–151 (1924)

[69] Wilson, G.T.: Factorization of the Covariance Generating Function of a Pure Moving-average. SIAM J. Numer Anal **6**, 1–7 (1969) doi: 10.1137/0706001

[70] Werner, W.: Some Improvements of Classical Iterative Methods for the Solution of Nonlinear Equations. In: Allgower, E.L. et al (eds) Numerical Solution of Nonlinear Equations, Proc. Bremen 1980 (L.N.M. 878), 427 - 440. Springer, Berlin (1982) doi: 10.1007/BFb0090691