2006

TR-2006007: Root-Finding with Eigen-Solving

Victor Y. Pan

Brian Murphy

Rhys Eric Rosholt

Root-finding with Eigen-solving *

Victor Y. Pan, Brian Murphy, Rhys Eric Rosholt,
Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468, USA
victor.pan,brian.murphy,rhys.rosholt@lehman.cuny.edu
Dmitriy Ivolgin, Yuqing Tang,
Xinmao Wang†, and Xiaodong Yan
Ph.D. Programs in Computer Science and Mathematics
Graduate School of CUNY
New York, NY 10036 USA
divolgin,ytang,xyan@gc.cuny.edu, xinmao@ustc.edu.cn†

Abstract

We survey and extend the recent progress in polynomial root-finding based on reducing the problem to eigen-solving for highly structured generalized companion matrices. In particular we cover the selection of the eigen-solvers and the matrices and the resulting benefits based on exploiting matrix structure. No good estimates for global convergence of the basic eigen-solvers have been proved, but according to ample empirical evidence, they require a constant number of iteration steps per eigenvalue. If we assume such a bound, then the resulting root-finders are optimal up to a constant factor because they use linear arithmetic time per step and perform with a constant (double) precision. Some eigen-solvers also supply useful auxiliary results for polynomial root-finding, e.g., new proximity tests for the roots and new technical support for polynomial factorization. The algorithms can be extended to solving secular equations.

Key words: Polynomial root-finding, Eigenvalues, Generalized companion matrices, Secular equation.

1 Introduction

1.1 Background

Polynomial root-finding is a classical and highly developed area but is still an area of active research [McN93], [McN97], [McN99], [McN02], [NAG88], [P97],

*Supported by PSC CUNY Award 66437-0035

[P01/02]. The algorithms in [P95], [P96], [P01/02] (cf. [S82], [G52/58], [CN94], [NR94], and [K98] on some important related works) approximate all roots of a polynomial by using arithmetic and Boolean time which is optimal up to polylogarithmic factors (under both sequential and parallel models of computing), but for practical numerical computing the users prefer other methods, including numerical methods of linear algebra.

In particular Matlab applies the QR algorithm to approximate the eigenvalues of the Frobenius companion matrix whose spectrum is precisely the set of the roots of the polynomial. This property characterizes the more general class, called generalized companion (hereafter *GC*) matrices of the polynomial. Thus root-finding for a polynomial can be reduced to eigen-solving for its GC matrices.

By implementing the effective eigen-solvers numerically, with double precision, one can outperform the known polynomial root-finders applied with extended precision, except that the output approximations to the eigenvalues are relatively crude and need refinement.

Malek and Vaillantcourt in [MV95] and [MV95a] and Fortune in [F01/02], however, propose to apply the QR algorithm with double precision recursively. They update the matrix as soon as the current approximate root values have been refined, and they show experimentally and in [F01/02] partly theoretically that this double-precision computation rapidly improves the approximations to the roots and yields them with a high precision. Instead of the Frobenius companion matrix, they use the diagonal plus rank-one (hereafter *DPR1*) GC matrices. A DPR1 GC matrix is defined for a fixed polynomial and a fixed set of its root approximations, which we call the *companion knots*. In [MV95], [MV95a], and [F01/02] the DPR1 GC matrix is updated as soon as the latter set is updated, and as soon as the approximations approach the roots closely enough, the computations are shifted to the Newton's or Durand–Kerner's (Weierstrass') root-refiners.

In [BGP03/05] and [BGP04] the QR stage of the algorithms in [MV95], [MV95a] and [F01/02] has been accelerated by the order of magnitude to yield linear (rather than quadratic) bounds on the arithmetic time per QR iteration step and also on the memory space. This has been achieved due to exploiting the rank structure of the DPR1 input matrix. Otherwise the behavior of the QR algorithm has not changed. In particular according to theoretical and extensive experimental study in [BGP03/05] and [BGP04], the algorithm remains as robust and converges as rapidly as the classical QR algorithm. The acceleration, however, is achieved only where the companion knots are real or, with the amendment in [BGP04] based on the Moebius transform of the complex plane, where they lie on a line or circle. This restriction causes no problem for computing crude initial approximations but generally prohibits their rapid QR refinement. Thus with the algorithms in [BGP03/05] and [BGP04] we still lacked a rapidly converging eigen-solver which would use linear time per step.

1.2 The QR/IPI DPR1 approach

In this paper we fix this deficiency by employing rather simple means. We examine other polynomial root-finders and matrix eigen-solvers in lieu of or in addition to the algorithms used in [MV95], [MV95a], and [F01/02], and we arrive at an alternating application of various algorithms in a unified recursive process for root-finding. In particular, we employ the earlier algorithm in [BGP02/04], which exploits the inverse power iteration with Rayleigh quotients (hereafter we say the *IPI*), The paper [BGP02/04] elaborates upon the application of the IPI to the Frobenius and DPR1 GC matrices. It is immediately verified that in the case of a DPR1 input, linear memory space and linear arithmetic time are sufficient per an IPI iteration step (as for the QR iteration in [BGP03/05],[BGP04]) and also for deflating a DPR1 matrix. The algorithm in [BGP02/04] is initialized with the companion knots on a large circle, which is a customary recipe for polynomial root-finding. The IPI, however, converges faster near an eigenvalue. This motivates using a hybrid algorithm where the IPI refines the crude approximations computed by the QR algorithm.

For the IPI, QR, and all other popular eigen-solvers no formal upper bounds are known on the number of steps they need for convergence. According to the ample empirical evidence, however, a single QR step as well as a single step of the IPI (initialized close enough to the solution) is typicaly sufficient per an eigenvalue [GL96, pages 359 and 363]. (We refer the reader to our Section 6.1 and [P05] on the nontrivial convergence acceleration of the IPI.) Under the latter semi-empirical model, the hybrid polynomial root-finders based on eigen-solving perform $O(n^2b/d)$ ops with the double precision of d bits to yield the output with the precision of b bits. For comparison, one needs at least n complex numbers to represent the coefficients $c_0, \dots, c_{n-1}$ of a monic input polynomial $c(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1x + c_0$, and needs at least the order of $(n-i)b$ bits in the coefficient c_i to approximate the roots within the error $2^{-b}\max_j |c_j|$. This means the order of n^2b bits in all coefficients, and therefore at least the same order of Boolean operations are required to process these bits. Such a lower bound is within the factor of $\log d \log\log d$ from the cited upper bounds based on eigen-solving, and this factor is a constant if d is constant. Compared to the Boolean complexity bound supported by the algorithm in [P01/02], the eigen-solving approach saves roughly the factor of $(\log n)(\log^2 n + \log b)$ and unlike the algorithm in [P01/02] requires no computations with extended precision.

1.3 Extensions and further study

How much can the progress be pushed further? According to the above argument, at most by a constant factor. This can still be practically important. The natural avenues are by exploting effective eigen-solvers such as Arnoldi's, non-Hermitian Lanczos', and Jacobi–Davidson's (besides the QR and IPI), applying them to the DPR1, Frobenius and other relevant GC matrices, and combining these eigen-solvers with some popular polynomial root-finders. We estimate the computational time for multiplication of these GC matrices and their shifted

inverses by vectors, deflation, and updating a GC matrix when its companion knot changes.

Besides we observe that the eigen-solvers also support some new proximity tests for the roots, that is, provide some bounds for the distances of the roots from a fixed point on the complex plane. Likewise we obtain some effective techniques for computing the basic root-free annuli for polynomial factorization as by-product of root-finding via eigen-solving (see Section 6.5).

We also comment on the potential extension of the algorithms to approximating the eigenvalues of sparse and structured matrices and to the correlation to solving secular equations. The links to many applications of these equations can be found in [G73], [M97], [BP98], and the references therein.

We focus on the main techniques for root-finding with matrix methods. For simplicity we assume dealing with monic input polynomials and skip the important special case of polynomials that have only real roots. We refer the reader to [JV04], [BP98], and the bibliography therein on these omitted subjects.

1.4 Organization of our paper

We organize our paper as follows. In Section 2, we recall some basic definitions. In Section 3, we study some relevant classes of GC matrices. In Section 4, we estimate the arithmetic computational complexity of some basic operations with these matrices. In Section 5, we study their computation, deflation, and updating. In Section 6, we cover various aspects of the application of eigen-solving for these matrices to polynomial root-finding. In Section 7, we comment on the extension of our methods to approximating matrix eigenvalues. In Section 8, we recall the correlation between the polynomial and secular equations. In Appendix A, we describe an approach to implicit removal of the multiple roots and root clusters. All authors share the responsibility for extensive numerical tests that supported the presented exposition and analysis. Otherwise the paper is due to the first author.

2 Basic definitions

$M = (m_{i,j})_{i,j=1}^{n}$ is an $n \times n$ matrix, $\mathbf{v} = (v_i)_{i=1}^{n}$ is a column vector of dimension n, M^T and $\mathbf{v}^T$ are their transposes.

$0_{k,l}$ is the $k \times l$ null matrix, $\mathbf{0}_k = 0_{k,k}$. I_k is the $k \times k$ identity matrix. I is the identity matrix of an appropriate size. $\mathbf{e}_i$ is the i-th column of I_n, $i = 1$, ..., n; $\mathbf{e}_1 = (1, 0, \dots, 0)^T$, $\mathbf{e}_n = (0, \dots, 0, 1)^T$.

$B = (B_1, \dots, B_k)$ is the $1 \times k$ block matrix with blocks B_1, ..., B_k. $\mathrm{diag}(s_i)_{i=1}^{n}$ is the $n \times n$ diagonal matrix with the diagonal entries s_1, ..., s_n. $\mathrm{diag}(B_1, \dots, B_k)$ is the $k \times k$ blocks diagonal matrix with the diagonal blocks B_1, ..., B_k.

$\det M$ and $c_M(\lambda) = \det(\lambda I - M)$ are the determinant and the characteristic polynomial of a matrix M, respectively.

$Z = (z_{i,j})_{i,j=1}^{n} = \begin{pmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}$ is the $n \times n$ shift matrix, $z_{i,i-1} = 1$ for $i = 2, \ldots, n$; $z_{i,j} = 0$ for $i \neq j + 1$, $Z\mathbf{v} = (0, v_1, \ldots, v_{n-1})^T$ for $\mathbf{v} = (v_i)_{i=1}^{n}$. Here and hereafter the blank space in the representation of matrices stands for their zero entries.

$f^* = a - b\sqrt{-1}$ is the complex conjugate of $f = a + b\sqrt{-1}$, for real $a = \Re f$ and $b = \Im c$. $\omega_n = \exp(2\pi\sqrt{-1}/n)$ is a primitive n-th root of 1.

$$V = \frac{1}{\sqrt{n}}(\omega_n^{ij})_{i,j=0}^{n-1} \tag{2.1}$$

is the unitary matrix of the discrete Fourier transform on the n-th roots of 1.

"DPR1", "GC","IPI", "RBDPR1", and "TPR1" stand for "diagonal plus rank-one", "generalized companion", "Inverse Power Iteration", "real block diagonal plus rank-one", and "triangular plus rank-one", respectively. In Sections 3 and 7, "ops" stands for "arithmetic operations". In the Appendix, "gcd" stands for "greatest common divisor".

$C = C_c$ is a GC matrix for a monic polynomial

$$c(x) = c_n x^n + c_{n-1}x^{n-1} + \ldots + c_1 x + c_0,\ c_n = 1, \tag{2.2}$$

if $c_C(x) = c(x)$.

3 Some classes of GC matrices

Root-finding for a polynomial $c(x)$ in equation (2.2) is equivalent to eigen-solving for a GC matrix $C = C_c$. The efficiency of the eigen-solving greatly depends on the choice of the matrix. Next we examine some most relevant classes of GC matrices (compare the studies of GC matrices in [E73], [G73], [B75], [F90], [C91], and [MV95]).

3.1 The Frobenius companion matrix

We first recall the classical Frobenius companion martix.

Theorem 3.1. *The matrix*

$$C = F_c = \begin{pmatrix} 0 & & & & -c_0 \\ 1 & \ddots & & & -c_1 \\ & \ddots & \ddots & & \vdots \\ & & \ddots & 0 & -c_{n-2} \\ & & & 1 & -c_{n-1} \end{pmatrix} \tag{3.1}$$

is a GC matrix F_c for a monic polynomial $c(x)$ in (2.2).

$C = F_c = Z - \mathbf{c}\mathbf{e}_n^T$ for $\mathbf{c} = (c_i)_{i=0}^{n-1}$.

3.2 DPR1 GC matrices

Theorem 3.2. *For a polynomial $c(x)$ in (2.2) and n distinct scalar companion knots $s_1, \dots, s_n$, write*

$$\mathbf{s} = (s_i)_{i=1}^n, q(x) = \prod_{i=1}^n (x - s_i), q_i(x) = \prod_{j=1, j\neq i}^n (x - s_j) = \frac{q(x)}{x - s_i},\ i = 1, \dots, n, \tag{3.2}$$

$$d_i = \frac{c(s_i)}{q'(s_i)},\ i = 1, \dots, n, \tag{3.3}$$

$$\mathbf{u} = (u_i)_{i=1}^n,\ \mathbf{v} = (v_i)_{i=1}^n,\ B = B_{\mathbf{s}} = \mathrm{diag}(s_i)_{i=1}^n,\ C = B - \mathbf{u}\mathbf{v}^T \tag{3.4}$$

where

$$|u_i| + |v_i| \neq 0,\ d_i = u_i v_i,\ i = 1, \dots, n. \tag{3.5}$$

Then C is a DPR1 GC matrix for the polynomial $c(x)$, that is, $c_C(x) = c(x)$.

Proof. $c_C(x) = q(x) + \sum_{i=1}^n d_i q_i(x)$ because the i-th and j-th rows of the matrix $xI - C - \mathrm{diag}(0, x - s_i, 0) - \mathrm{diag}(0, x - s_j, 0)$ for $i \neq j$ are proportional to one another, whereas $c(x) = q(x) + \sum_{i=1}^n d_i q_i(x)$ due to the Lagrange interpolation formula. □

3.3 RBDPR1 GC matrices

The polynomials $c(x)$ in (2.2) with real coefficients may have some pairs of nonreal complex conjugate roots. In this case the DPR1 matrices would have nonreal entries. To avoid this deficiency we introduce the *RBDPR1* GC matrices whose diagonal blocks have size of at most two. We begin with an auxiliary result on block diagonal plus rank-one matrices.

Theorem 3.3. *Let $B = \mathrm{diag}(B_1, \dots, B_k)$ where B_i are $n(i) \times n(i)$ matrices, $m(i) = \sum_{j=1}^i n(j)$, $i = 1, \dots, k$, $m(k) = n$. Write*

$$P_i = \mathrm{diag}(0_{m(i-1)}, I_{n(i)}, 0_{n-m(i)}), \overline{P}_i = (0_{n(i), m(i-1)}, I_{n(i)}, 0_{n(i), n-m(i)}),$$

so that $\overline{P}_i \mathbf{w} = (w_j)_{j=m(i-1)+1}^{m(i)}$ is the projection of a vector $\mathbf{w} = (w_j)_{j=1}^n$ into its subvector made up of the $n(i)$ respective coordinates, whereas by padding the vector $\overline{P}_i \mathbf{w}$ with the $m(i-1)$ leading zero coordinates and the $n - m(i)$ trailing zero coordinates, we arrive at the vector $P_i \mathbf{w}$. Let s_i be an eigenvalue of the matrix B and let $C = B - \mathbf{u}\mathbf{v}^T$. Then $c_C(s_i) = \det(s_i I - B + P_i \mathbf{u}\mathbf{v}^T P_i) = \det(s_i I - B_i + \overline{P}_i \mathbf{u}\mathbf{v}^T \overline{P}_i) \prod_{j \neq i} \det(s_i I - B_j) = c_B(s_i), i = 1, \dots, n$.

Proof. Let $\mathbf{q}_i \neq \mathbf{0}$ be a left eigenvector of the matrix B associated with the eigenvalue s_i such that $B\mathbf{q}_i^T = s_i \mathbf{q}_i$. Write $a_i = \mathbf{q}_i^T \mathbf{u}$ and $\mathbf{u} = (u_j)_{j=1}^n$. If $a_i = 0$, then we have $\mathbf{q}_i^T (s_i I - B) = \mathbf{q}_i^T (s_i I - C) = \mathbf{0}^T$ and therefore $c_B(s_i) = c_C(s_i) = 0$. Otherwise subtract the vector $\frac{u_j}{a_i}\ \mathbf{q}_i^T (s_i I - B + \mathbf{u}\mathbf{v}^T) = \frac{u_j}{a_i} \mathbf{q}_i^T \mathbf{u}\mathbf{v}^T = u_j \mathbf{v}^T$ from the j-th row of the matrix $s_i I - C = s_i I - B + \mathbf{u}\mathbf{v}^T$ for $j = 1, \dots, m(i-1)$, and for

$j = m(i)+1, \dots, n$. This turns the matrix s_iI-C into the matrix $s_iI-B+P_i\mathbf{u}\mathbf{v}^T$ without changing its determinant $c_C(s_i)$. Observe that $\det(s_iI-B+P_i\mathbf{u}\mathbf{v}^T) = \det(s_iI-B+P_i\mathbf{u}\mathbf{v}^TP_i)$ and that $s_iI-B+P_i\mathbf{u}\mathbf{v}^TP_i$ is a block diagonal matrix with the diagonal blocks $s_jI_{n(j)}-B_j$ for $j = 1, \dots, i-1, i+1, \dots, k$ and $s_iI_{n(i)} - B_i + \overline{P}_i\mathbf{u}\mathbf{v}^T\overline{P}_i^T$. This proves Theorems 3.2. □

Theorem 3.3 enables the following alternative proof of Theorem 3.2.

Proof. (*An alternative proof of Theorem 3.2.*) Apply Theorem 3.3 for $B_j = (s_j)$, $j = 1, \dots, n$, $B = \text{diag}(s_j)_{j=1}^n$, $k = n$, $n_i = 1$, $i = 1, \dots, n$. Obtain that $c_C(s_i) = d_iq_i(s_i)$, substitute $q_i(s_i) = q'(s_i)$, $d_iq'(s_i) = c(s_i)$, and obtain that $c(s_i) = c_C(s_i)$, $i = 1, \dots, n$. This proves the theorem because $c(x)$ and $c_C(x)$ are monic polynomials of degree n. □

Theorem 3.4. *For two integers h and n, $0 \le h \le \frac{n}{2}$, a polynomial $c(x)$ in (2.2) with real coefficients, h distinct pairs of real numbers $(f_1, g_1), \dots, (f_h, g_h)$ such that $g_i \neq 0$ for all i, and $n-2h$ distinct real numbers $s_{2h+1}, \dots, s_n$, write $s_{2i-1} = f_i + g_i\sqrt{-1}$, $s_{2i} = f_i - g_i\sqrt{-1}$, $B_i = \begin{pmatrix} f_i & g_i \\ -g_i & f_i \end{pmatrix}$, $i = 1, \dots, h$; $B_{j-h} = (s_j)$, $j = 2h+1, \dots, n$, $B = \text{diag}(B_j)_{j=1}^{n-h}$; $q(x) = \prod_{j=1}^n (x - s_j)$, $d_j = \frac{c(s_j)}{q'(s_j)}$, $j = 1, \dots, n$, so that $d_{2i} = d_{2i-1}^*$, $i = 1, \dots, h$. Let*

$$\mathbf{u} = (u_j)_{j=1}^n,\ \mathbf{v} = (v_j)_{j=1}^n,\ C = B - \mathbf{u}\mathbf{v}^T \tag{3.6}$$

where

$$u_{2i-1}v_{2i-1} + u_{2i}v_{2i} + (u_{2i-1}v_{2i} - u_{2i}v_{2i-1})\sqrt{-1} = 2d_{2i-1},$$

for $i = 1, \dots, h$, $|u_j| + |v_j| \neq 0$, $u_jv_j = d_j$, $j = 2h+1, \dots, n$.

Then the RBDPR1 matrix C is a GC matrix of the polynomial $c(x)$, that is, $c(x) = c_C(x)$.

Proof. Apply Theorem 3.3 for $k = n-h$ and deduce that $(s_{2i-1}-s_{2i})c_C(s_{2i-1}) = q_{2i-1}(s_{2i-1})\det(s_{2i-1}I_2 - W_i)$ for

$$W_i = B_i - \overline{P}_i\mathbf{u}\mathbf{v}^T\overline{P}_i = \begin{pmatrix} f_i - u_{2i-1}v_{2i-1} & g_i - u_{2i-1}v_{2i} \\ -g_i - u_{2i}v_{2i-1} & f_i - u_{2i}v_{2i} \end{pmatrix}, i = 1, \dots, h.$$

Substitute $s_{2i-1} - f_i = g_i\sqrt{-1}$ and deduce that

$$s_{2i-1}I_2 - W_i = \begin{pmatrix} g_i\sqrt{-1} + u_{2i-1}v_{2i-1} & -g_i + u_{2i-1}v_{2i} \\ g_i + u_{2i}v_{2i-1} & g_i\sqrt{-1} + u_{2i}v_{2i} \end{pmatrix},$$

so that $\det(s_{2i-1}I_2 - W_i) = g_i(u_{2i}v_{2i-1} - u_{2i-1}v_{2i} + (u_{2i-1}v_{2i-1} + u_{2i}v_{2i})\sqrt{-1})$, $i = 1, \dots, h$. Substitute the latter expression and the equations $s_{2i-1} - s_{2i} = 2g_i\sqrt{-1}$ and $q_j(s_j) = q'(s_j)$ for $j = 2i-1$ into our expression above for $c_C(s_{2i-1})$ and obtain that

$$2g_ic_C(s_{2i-1})\sqrt{-1} = g_iq'(s_{2i-1})((u_{2i-1}v_{2i-1} + u_{2i}v_{2i})\sqrt{-1} + u_{2i}v_{2i-1} - u_{2i-1}v_{2i}),$$

$$u_{2i-1}v_{2i-1} + u_{2i}v_{2i} + (u_{2i-1}v_{2i} - u_{2i}v_{2i-1})\sqrt{-1} = \frac{2c_C(s_{2i-1})}{q'(s_{2i-1})} = 2d_{2i-1}.$$

Now apply equation (3.3) and deduce that $c_C(s_{2i-1}) = c(s_{2i-1})$ for $i = 1, \ldots, h$.

Since the polynomials $c(x)$ and $c_C(x)$ have real coefficients, obtain that $c_C(s_{2i}) = c_C^*(s_{2i-1}) = c^*(s_{2i-1}) = c(s_{2i})$, $i = 1, \ldots, h$. Deduce that $c_C(s_j) = c(s_j)$ by applying Theorem 3.3 for $B_j = \mathrm{diag}(s_j), j = 2h+1, \ldots, n$. Now Theorem 3.4 follows because $c(x)$ and $c_C(x)$ are monic polynomials of degree n. □

3.4 Arrow-head GC matrices

Theorem 3.5. *For a polynomial $c(x)$ in (2.2) and n distinct nonzero scalars $\overline{s}_1, \ldots, \overline{s}_n$, write*

$$\overline{q}(x) = \prod_{i=2}^{n}(x - \overline{s}_i),\ \overline{q}_i(x) = \prod_{j=2, j\neq i}^{n}(x - \overline{s}_j) = \frac{\overline{q}(x)}{x - \overline{s}_i},\ i = 2, \ldots, n, \tag{3.7}$$

$$\overline{d}_i = \frac{c(\overline{s}_i)}{\overline{q}'(\overline{s}_i)} = \frac{c(\overline{s}_i)}{\overline{q}_i(\overline{s}_i)},\ i = 2, \ldots, n,\ \overline{d}_1 = \frac{c(\overline{s}_1)}{\overline{q}(\overline{s}_1)} + \sum_{i=2}^{n}\frac{\overline{d}_i}{\overline{s}_1 - \overline{s}_i} \tag{3.8}$$

and choose n pairs of scalars $\overline{u}_i$, $\overline{v}_i$, $i = 1, \ldots, n$ such that

$$\overline{u}_1 = \overline{d}_1 - \overline{s}_1,\ \overline{v}_1 = 0,\ \ \overline{u}_i\overline{v}_i = \overline{d}_i,\ i = 2, \ldots, n. \tag{3.9}$$

Write $B = B_{\overline{\mathbf{s}}} = \mathrm{diag}(\overline{s}_i)_{i=1}^n$, $\overline{\mathbf{u}} = (\overline{u}_i)_{i=1}^n, \overline{\mathbf{v}} = (\overline{v}_i)_{i=1}^n$. Then the north-western arrow-head matrix

$$C = B - (\overline{\mathbf{u}}\mathbf{e}_1^T + \mathbf{e}_1\overline{\mathbf{v}}^T) \tag{3.10}$$

is a GC matrix of the polynomial $c(x)$, that is, $c_C(x) = c(x)$.

Proof. Expand the determinant $c_C(x) = \det(xI - C)$ along the first row or the first column of the matrix $xI - C$ and deduce that

$$c_C(x) = (x + \overline{u}_1)\overline{q}(x) - \sum_{i=2}^{n}\overline{u}_i\overline{v}_i\overline{q}_i(x).$$

Therefore,

$$c_C(\overline{s}_i) = \overline{u}_i\overline{v}_i\overline{q}_i(\overline{s}_i),\ i = 2, \ldots, n;$$

$$c_C(\overline{s}_1) = (\overline{s}_1 + \overline{u}_1)\overline{q}(\overline{s}_1) - \sum_{i=2}^{n}\overline{u}_i\overline{v}_i\overline{q}_i(\overline{s}_1).$$

Substitute equations (3.8) and (3.9) and deduce that $c_C(\overline{s}_i) = c(\overline{s}_i), i = 1, \ldots, n$. The theorem follows because the monic polynomials $c_C(x)$ and $c(x)$ of degree n shares their values at n distinct points $\overline{s}_1, \ldots, \overline{s}_n$. □

3.5 Further variations of GC matrices

The Frobenius, DPR1, and arrow-head matrices are the most popular classes of GC matrices. The RBDPR1 GC matrices extend the DPR1 GC matrices in the case of a real input and a nonreal output. Similarly we can extend the class of the arrow-head matrices. Let us point out some further variations and extensions.

1. *Variations of the parameters.*

   For fixed companion knots, each GC matrix in Subsections 3.2–3.4 is defined with n or $n-1$ parameters, which we can vary at will.

   Example 3.1. *Some sample choices of the parameters.*

   - $u_i = 1,\ v_i = d_i,\ i = 1, \dots, n,$ *in Theorem 3.2*
   - $v_{2i-1} = v_{2i} = 1,\ u_{2i-1} = \Re d_i + \Im d_i,\ u_{2i} = \Re d_i - \Im d_i,\ u_j = 1,\ v_j = d_j,\ j = 2i+1, \dots, n,$ *in Theorem 3.4*
   - $\overline{u}_i = 1,\ \overline{v}_i = \overline{d}_i,\ i = 2, \dots, n,$ *in Theorem 3.5*

   Example 3.2. *Scaling for numerical stabilization.*

   In Theorem 3.2 require that $|u_i| = |v_i|$ *(resp.* $|\overline{u}_i| = |\overline{v}_i|$*) for all* i*.*

2. *Variation of the input polynomial.*

   We can fix a root b of a polynomial $(x-b)c(x)$ and seek the remaining n roots based on Theorem 3.5 where we replace $c(x)$ with $(x-b)c(x)$, replace n with $n+1$, and choose $s_1 = b$, so that $\overline{d}_1 = \sum_{i=2}^{n} \frac{\overline{d}_i}{\overline{s}_1 - \overline{s}_i}$.

3. *Modification of the matrices.*

   - We can extend Theorem 3.4 by choosing any set of $2h$ real 2×2 matrices $B_i = \begin{pmatrix} f_i & g_i \\ j_i & k_i \end{pmatrix},\ i = 1, \dots, h$ and any set of $n-2h$ real 1×1 matrices $B_j = (s_j),\ j = 2i+1, \dots, n$, with n distinct eigenvalues overall. Suppose s_{2i-1} and s_{2i} denote the eigenvalues of the matrix $B_i,\ i = 1, \dots, h$. Then for any choice of the values $u_{2i-1}, u_{2i}, v_{2i-1}, v_{2i}$ satisfying

     $$(s_{2i-1} - f_i)u_{2i}v_{2i} + (s_{2i-1} - j_i)u_{2i-1}v_{2i-1} + g_i u_{2i}v_{2i-1} + h_i u_{2i-1}v_{2i}$$
     $$= 2(s_{2i-1} - s_{2i})d_{2i-1}, \quad i = 1, \dots, h,$$

     the matrix C in (3.6) is a GC matrix of the polynomial $c(x)$.
   - We can interchange the roles of the subscripts 1 and n throughout Theorem 3.5 to arrive at the dual south-eastern arrow-head matrix C such that $c_C(x) = c(x)$. Alternatively, we can turn a north-western arrow-head matrix into a south-eastern one by applying the similarity transform $C \longrightarrow JCJ$ where $J = J^{-1}$ is the reflection matrix whose entries equal one on the antidiagonal and equal zero elsewhere.

- More generally, any similarity transform $C \longrightarrow S^{-1}CS$ of a GC matrix $C = C_c$ for a polynomial $c(x)$ maps C into a GC matrix for $c(x)$. If all n roots of $c(x)$ are distinct, then the converse is also true, that is, two GC matrices associated with such a polynomial $c(x)$ are always similar to one another. In the next subsection we specify such transforms among our sample GC matrices. The similarity transforms can be of some help in actual computations, e.g., with appropriate diagonal matrices S we can scale the GC matrices to improve their conditioning. This diagonal scaling of GC matrices is equivalent to choosing n parameters among $u_i, v_i,\ i = 1, \dots, n$ in Sections 3.2 and 3.3 or $n-1$ parameters among $\overline{u}_i, \overline{v}_i,\ i = 1, \dots, n$ in Section 3.4.

3.6 Similarity transforms among GC matrices of four classes

Simple similarity transforms of a 2×2 matrix $B = \begin{pmatrix} f_i & g_i \\ -g_i & f_i \end{pmatrix}$ into the diagonal matrix $\mathrm{diag}(d_{2i-1}, d_{2i})$, $d_{2i-1} = f_i + g_i\sqrt{-1}$, $d_{2i} = f_i - g_i\sqrt{-1}$ can be immediately extended to transforming a block diagonal matrix B in Theorem 3.3 into a diagonal matrix. This relates the matrix classes DPR1 and RBDPR1 in Sections 3.2 and 3.3 and similarly for the arrow-head matrices in Section 3.4 and their counter-parts where the diagonal entries can be replaced by real blocks.

Furthermore, both arrow-head matrix C in (3.10) and the transpose F_c^T of a Frobenius matrix F_c in (3.1) are TPR1 matrices, and the paper [PKMRTYCa] shows non-unitary similarity transforms of TPR1 into DPR1 matrices as well as into arrow-head matrices. For the matrices F_c^T and C in (3.10), these transforms into DPR1 matrices use $O(n^2)$ ops. There are also similarity transforms of our matrices in (3.4), (3.6) and (3.10) into a Frobenius matrix via their reduction to a Hessenberg matrix in [W65, pages 405–408] as well as a unitary similarity transform of a matrix F_c into a DPR1 matrix due to the following result.

Theorem 3.6. *The similarity transform with the matrix V in (2.1) maps the Frobenius matrix F_C in (3.1) into a DPR1 matrix:*

$$VF_CV^H = \mathrm{diag}(w_n^i)_{i=0}^{n-1} + \mathbf{u}\mathbf{v}^T,\ \mathbf{u} = V\mathbf{c},\ \mathbf{v}^T = \mathbf{e}_n^T V^H.$$

4 The complexity of some basic computations

Let us estimate the arithmetic complexity of multiplying the matrices C in (3.1)–(3.10) and their shifted inverses with vectors, which are the basic operations in some popular eigen-solvers.

The estimates are presented in Table 1 and Theorem 4.1. The columns of Table 1 marked by a/s, m, and r show how many times we add/subtract, multiply, and compute reciprocals, respectively, to arrive at the vectors $C\mathbf{w}$, $(xI - C)^{-1}\mathbf{w}$, and $(xI - C - \mathbf{g}\mathbf{h}^T)^{-1}\mathbf{w}$ for a fixed pair of vectors $\mathbf{g}, \mathbf{h}$, any

Matrix C	Vectors		$C\mathbf{w}$		$(xI-C)^{-1}\mathbf{w}$			$(xI-\overline{C})^{-1}\mathbf{w}$		
	$\mathbf{g}$	$\mathbf{h}$	m	a/s	r	m	a/s	r	m	a/s
Frobenius in (3.1)	$\mathbf{c}$	$\mathbf{e}_n$	n	$n-1$	0	$2n-1$	$2n-2$	0	n	$n-1$
					1	$n-1$	n	1	0	1
DPR1 in (3.4)	$\mathbf{u}$	$\mathbf{v}$	$2n-1$	$2n-2$	1	$2n$	$2n-1$	0	n	0
					$n+1$	$n+1$	$2n$	n	0	n
RBDPR1 in (3.6)	$\mathbf{u}$	$\mathbf{v}$	$2n+2h$	$2n$	$n+1$	$n+h$	$2n-1+2h$	n	h	$2h$
					$2n$	$2h$	$n+3h$	h	h	h
Arrow-head in (3.10)	$\mathbf{e}_1$	$-\mathbf{v}$	$2n-1$	$2n-2$	0	$2n-1$	$2n-2$	0	n	$n-1$
					n	$n-1$	$2n-1$	n	0	n

Table 1: The complexity of basic computations with matrices in Section 3

scalar x such that the matrices $xI-C$ and $xI-C-\mathbf{g}\mathbf{h}^T$ are nonsingular, and any vector $\mathbf{w}$. Some entries of the table have two levels. In the upper level the number of ops depending on the vector $\mathbf{w}$ is displayed; in the low level the number of the other ops is displayed. All estimates hold where the parameters u_i, v_i, $\overline{u}_i$, and $\overline{v}_i$ satisfy the equations in Example 3.1. For other choices of the parameters the arithmetic cost can slightly change.

Theorem 4.1. *Let a polynomial $c(x)$ and scalars s_i, d_i, u_i, v_i, $\overline{s}_i$, $\overline{d}_i$, $\overline{u}_i$, and $\overline{v}_i$ for $i=1,\dots,n$, satisfy equations (3.1)–(3.10). Let four matrices, all denoted by C, satisfy equations (3.1), (3.4), (3.6) and (3.10), respectively, and let $\overline{C}=Z$ for C in (3.1), $\overline{C}=B$ for C in (3.4) and (3.6), and $\overline{C}=B+\overline{\mathbf{u}}\mathbf{e}_1^T$ for C in (3.10), so that in all three cases $C-\overline{C}$ denotes rank-one matrices $-\mathbf{c}\mathbf{e}_n^T$, $-\mathbf{u}\mathbf{v}^T$, and $\mathbf{e}_1\overline{\mathbf{v}}^T$, respectively. Let x be a scalar such that the matrices $xI-C$ and $xI-\overline{C}$ are nonsingular. Let $\mathbf{w}$ be a vector. Then Table 1 displays the upper bounds on the numbers of the operations a/s, m, and r involved in computing the vectors $C\mathbf{w}$, $(xI-C)^{-1}\mathbf{w}$, and $(xI-\overline{C})^{-1}\mathbf{w}$. For the two latter vectors, an upper bound on the number of the ops not depending on the vector $\mathbf{w}$ is showed in the lower level of each table's entry. The other ops are counted in its upper level.*

Proof. The straightforward algorithms support the estimates for the complexity of computing the vectors $C\mathbf{w}$ and $(xI-\overline{C})^{-1}\mathbf{w}$. (Apply the forward substitution algorithm under (3.10) for $(xI-\overline{C})^{-1}\mathbf{w}$.)

Compute the vectors $(xI-C)^{-1}\mathbf{w}$ for the matrices C in (3.1) and (3.10) by applying Gaussian elimination. For a Frobenius matrix C in (3.1), first eliminate the subdiagonal entries by using no pivoting and then apply the back substitution. For an arrow-head matrix C in (3.10), first eliminate the first row of the matrix and then apply the forward substitution. Verify the respective estimates in Table 1 by inspection.

The Sherman–Morrison–Woodbury formula [GL96, page 50], [BGP02/04, Section 5] implies that $(xI-C)^{-1}=(I+\frac{1}{1-\tau}(B-xI)^{-1}\mathbf{d}\mathbf{e}^T)(xI-B)^{-1}$, $\tau=\mathbf{e}^T(xI-B)^{-1}\mathbf{d}$, for $\mathbf{e}=(1,\dots,1)^T$ and the DPR1 matrix C in (3.4). Therefore, $(xI-C)^{-1}\mathbf{w}=(xI-B)^{-1}\mathbf{w}+\frac{\sigma}{1-\tau}(B-xI)^{-1}\mathbf{d}$, $\sigma=\mathbf{e}^T(B-xI)^{-1}\mathbf{w}$, and the estimates claimed in Table 1 follow. Similarly proceed with the RBDPR1 matrices. □

5 The computation, updating and deflation of a GC matrix

This section covers the computation of a GC matrix and its updating when the set of companion knots and the input polynomial are modified.

5.1 The computation of a GC matrix

The matrix $C = F_c$ in (3.1) is given with the coefficients of the polynomial $c(x)$. The computation of the GC matrices C of the other three classes can be exemplified with the case of the DPR1 matrices in (3.4) and can be reduced essentially to computing the ratios $d_j = \frac{c(s_j)}{q'(s_j)}$ at the n distinct companion knots $s_j,\ j = 1, \dots, n$.

The computation is simplified for the customary initial choice of the knots equally spaced on a large circle such that $s_j = a\omega_n^{j-1},\ j = 1,\ \dots,\ n$, where a exceeds by a sufficiently large factor the root radius $r = \max_j |z_j|$ of the polynomial $c(x) = \prod_{j=1}^n (x - z_j)$. In this case $q(x) = x^n - a^n$, $q'(x) = nx^{n-1}$. Then application of the generalized discrete Fourier transform [P01, Section 2.4] yields all ratios d_j in (3.3) by using $O(n \log n)$ ops. (Surely if n is a power of two, then one should just apply FFT.)

If, however, some crude initial approximations to the roots are available, they are a natural choice for the companion knots. Then the above complexity bound of $O(n \log n)$ ops generally increases to $O(n \log^2 n)$ based on a numerically unstable algorithm in [P01, Section 3.1] and to $2n^2 - n$ based on a stable version of the Horner's algorithm [BF00]. Even the latter cost bound is still dominated at the subsequent stages of the root approximation.

When the root approximations and the companion knots or the input polynomial are updated, one can recompute the matrix C by applying the algorithms above, but let us next examine some alternative updating means.

5.2 The reversion of a polynomial, the variable shift and the GC matrices

The reversion of the input polynomial $c(x)$ in (2.2) and the shift of the variable x by a scalar s are routinely applied in the preprocessing of the input polynomial and also recursively used in some popular root-finders. To update the associated GC matrices for the shifted polynomial $c_{shift}(x) = c(x - s)$, we can re-use the same values $d_1, \dots, d_n$ at the knots $x = s_j + s$ because $c_{shift}(s_j + s) = c(s_j)$ and $q'_{shift}(s + s_j) = q'(s_j)$, $j = 1, \dots, n$. For the reverse polynomial $c_{rev}(x) = x^n c(1/x)$ we have $c_{rev}(\frac{1}{s_i}) = s_i^{-n} c(s_i)$, $q'_{rev}(\frac{1}{s_i}) = \prod_{j \neq i} (\frac{1}{s_i} - \frac{1}{s_j}) = (-1)^{n-1} s_i^{2-n} q'(s_i) / \prod_{j=1}^n s_j$, $i = 1, \dots, n$, and so we can update $d_1, \dots, d_n$ by computing $s_1^{2-n}, \dots, s_n^{2-n}$ and in addition performing $O(n)$ ops.

Alternatively, we can replace the GC matrix C with C^{-1} or $C - sI$, respectively. We can compute the first column of the matrix $(F_c - sI)^{-1}$ in $O(n)$ ops, due to Theorem 4.1, and we can represent the matrix with this column [C96].

Due to the Sherman–Morrison–Woodbury formula and Theorem 4.1, we obtain the DPR1 representation of the matrix $(C - sI)^{-1}$ by using $O(n)$ ops for a matrix C in (3.4), (3.6), and (3.10). In particular it takes $2n$ divisions, $2n$ multiplications and n additions/subtractions for the DPR1 matrix C in (3.4).

5.3 Deflation of polynomials and GC matrices

Suppose we have approximated a root z of a polynomial $c(x)$ in (2.2). Then we can deflate the associated matrices C in (3.1), (3.4), (3.6) and (3.10) preserving their structure.

For the Frobenius matrix in (3.1), we just compute the quotient polynomial $c^{new}(x) = \frac{c(x)}{x-z}$ by using $n-1$ subtractions and $n-1$ divisions. For the three other matrix classes we also use $O(n)$ ops but unlike the Frobenius case involve no coefficients of $c(x)$.

For the DPR1 matrix in (3.4), we replace the vector $\mathbf{s} = (s_i)_{i=1}^{n}$ with $\mathbf{s}^{new} = (s_i)_{i=1}^{n-1}$ and compute the associated vector $\mathbf{d}^{new} = (d_i^{new})_{i=1}^{n-1}$ according to equations (6.2) in [BGP02/04], that is,

$$d_i^{new} = d_i \frac{s_i - s_n}{s_i - z},\ i = 1, \dots, n-1. \tag{5.1}$$

This requires $2n-2$ additions/subtractions, $n-1$ multiplications, and $n-1$ divisions. If $z \approx s_n$, then $d_i^{new} \approx d_i$ for all $i < n$, and deflation is cost-free.

Similarly we deflate the matrices C in (3.6) and (3.10). Under (3.10) we write $\overline{\mathbf{s}}^{new} = (\overline{s}_i)_{i=1}^{n-1}$, rely on (3.8), and compute the associated vector $\overline{\mathbf{d}}^{new} = (\overline{d}_i^{new})_{i=1}^{n-1}$ according to the following equations, which extend equations (5.1):

$$\overline{d}_1^{new} = \overline{d}_1 \frac{\overline{s}_n}{z},\ \overline{d}_i^{new} = \overline{d}_i \frac{\overline{s}_i - \overline{s}_n}{\overline{s}_i - z},\ i = 2, \dots, n-1. \tag{5.2}$$

The computations involve $2n-3$ additions/subtractions, $n-1$ multiplications, and $n-1$ divisions. We can keep the deflation processes (5.1), (5.2) in the field of real numbers for polynomials with real coefficients. We just need to deflate the pair of the complex conjugate roots as soon as one of them is approximated.

And again if $z \approx \overline{s}_n$, then $\overline{d}_i^{new} \approx \overline{d}_i$ for all $i < n$, and the deflation is cost-free.

5.4 Updating the companion knots and matrices

If we have updated a single companion knot s_i, we can update the DPR1 matrix in (3.4) by using $O(n)$ ops. Indeed the values $c(s_j)$ remain invariant for $j \neq i$, whereas we can compute the values $c(s_i)$ and $q_i(s_i)$ by using $4n-3$ ops with Horner's algorithm, and we can compute $q_j^{new}(s_j) = q_j^{old}(s_j)\ \frac{s_j - s_i^{new}}{s_j - s_i^{old}}$ for every $j \neq i$ by using four ops per value.

Similar observations apply to the RBDPR1 and the arrow-head matrices.

6 Root-finding via eigen-solving

6.1 Approximating the extremal eigenvalues

In Table 2 we display the numbers of basic operations required at the kth iteration step in four popular eigen-solvers. They approximate the extremal eigenvalues, that is, the eigenvalues which are the farthest from and the closest to the selected shift value s and which for $s = 0$ are the absolutely largest and the absolutely smallest eigenvalues, respectively. Tables 1 and 2 together furnish us with the respective ops estimates for these eigen-solvers.

The inverse power iteration, IPI, approximates the single eigenvalue closest to the shift value s. We refer the reader to [GL96, Sections 8.2.2 and 8.2.3], [S98, Section 2.1.2]), and [BDDRvV00], and the bibliography therein on this iteration and its Rayleigh–Ritz block version for approximating some blocks of the extremal eigenvalues. The recent papers [BGP02/04] and [P05] specialize the IPI to the DPR1 and Frobenius input matrices.

The Jacobi–Davidson algorithms also approximate the single extremal eigenvalue or a block of such eigenvalues [S98, Section 6.2], [BDDRvV00], whereas the Arnoldi and the non-Hermitian Lanczos algorithms [GL96, Section 9.4], [S98, Chapter 5], [BDDRvV00] approximate simultaneously a small number of eigenvalues consisting of both eigenvalues closest to and farthest from a fixed shift value. The algorithms only compute crude approximations to the closest eigenvalues, but in the root-finding application this deficiency can be easily fixed because the absolutely smallest roots of a polynomial $c(x)$ are the absolutely largest for its reverse polynomial $c_{rev}(x)$. Actually all these algorithms approximate the Ritz eigenpairs, that is, the pairs of the eigenvalues and the associated eigenvector (or more generally, associated eigenspaces).

Table 2 does not cover the ops required for approximating a Ritz pair for an $k \times k$ auxiliary Hessenberg (resp. tridiagonal) matrix in the Arnoldi (resp. non-Hermitian Lanczos) algorithm and for computing the Euclidean vector norms (at most two norms are required per step). Actually, to make the Arnoldi and the Jacobi–Davidson algorithms competitive, one must keep k smaller, although such a policy is in conflict with the task of approximating the eigenvalues closely. This seems to give the upper hand to the Lanczos algorithm and the IPI, although such a theoretical conclusion is yet to be confirmed experimentally.

Another crucial factor is the number of iteration steps required for convergence, but all the cited eigen-solvers have good local and global convergence according to the theory and extensive empirical evidence [GL96], [S98], [BDDRvV00]. The local convergence of the Arnoldi and Lanczos algorithms can be substantially speeded up with the shift-and-invert techniques [S98, pages 334–336].

The convergence of the IPI can be additionally accelerated in the case of the Frobenius input matrix $C = F_c$ [P05]. Formally, let $\theta = \max_{\mu \neq \lambda} |\frac{\mu - s}{\lambda - s}|$ where s is the selected shift value approximating an eigenvalue λ, and the maximum is over all other eigenvalues μ. Then the eigenvalue λ is approximated within the error in $O(\theta^k)$ in k IPI steps, whereas the much smaller error bound in $O(\theta^{2^k})$ can be

reached in k steps of the algorithm in [P05]. The latter algorithm, however, uses almost as many ops per step as six FFT's at 2^h points for $h = \lceil \log_2(2n-1) \rceil$, that is, the order of $n \log n$ ops per step, versus $O(n)$ ops per an IPI step.

Finally, since all of the above algorithms approximate the eigenvalues which are the closest to the shift value s, a by-product of their application is the *proximity test* at the complex point s for the roots of the polynomial $c(x)$. We exploit this observation at the very end of the section.

6.2 Approximating all eigenvalues

To extend the algorithms in the previous subsection to computing all eigenvalues, we can recursively combine them with deflating the polynomial $c(x)$ and/or updating its GC matrix (see Section 5.3) provided the eigenvalues have been approximated closely enough to counter the error propagation. We discuss how to improve the initial approximations to the eigenvalues in the next subsections.

We can dispense with deflation and apply the selected eigen-solvers to the same matrix but vary the shift values s trying to direct the eigen-solver to a new eigenvalue. The iteration can occasionally converge to the same eigenvalue already approximated, but according to the empirical evidence and some theory available for Newton's iteration, running it for the order of n to $n \log n$ initial shift values equally spaced on a large circle is usually sufficient to approximate all eigenvalues.

Furthermore, the algorithm in [P05] always enforces convergence to a new eigenvalue of the Frobenius matrix F_c, so that in n applications it outputs approximations to all n eigenvalues.

Finally we recall that the QR algorithm approximates all eigenvalues of a matrix in roughly $10n^3$ ops according to extensive empirical evidence [GL96, Section 7.5.6]. The bound relies on using $10n^2$ ops per QR iteration step for an $n \times n$ Hessenberg input matrix. For a DPR1 input and the initial companion knots equally spaced on a circle, as well as for any set of companion knots on a circle or a line, the QR algorithms in [BGP03/05], [BGP04] uses $120n$ ops per step, so that we can extrapolate the cited empirical cost bound to $120n^2$ for all eigenvalues.

6.3 Eigen-solvers and root-finders as root-refiners

Based on our study in the previous sections, we should approximate the roots of a polynomial $c(x)$ in (2.2) by applying selected eigen-solvers to appropriate GC matrices, performing the computations numerically, with double precision, updating the matrices when the approximations to the eigenvalues improve, and possibly changing the eigen-solvers during the iteration process. According to the extensive experiments in [MV95], [MV95a], and [F01/02] and partly to some formal study in [F01/02], a variant of this approach with the QR algorithm and the DPR1 GC matrices rapidly achieves high precision approximations to the eigenvalues.

Eigen-solver	$C * v$	$C^H * v$	$(C - \mu I)^{-1} * v$	Additional Ops
Arnoldi	1			$(4k + 4) + O(1)$
non-Hermitian Lanczos	1	1		$15n + O(1)$
Jacobi–Davidson	1		1	$(9 + k^2)n + O(1)$
IPI	1		1	$5n - 1$

Table 2: The numbers of multiplications of the matrix C, C^H and $(C - \mu I)^{-1}$ by vectors and additional ops at the kth iteration step

Our study in Section 3 indicates that the effect should be the same if we replace the DPR1 matrices with the RBDPR1 (in the real case) or arrow-head matrices. Furthermore, all other eigen-solvers in the previous subsections can be used instead of the QR algorithm except for the algorithm in [P05]. This algorithm, devised for the Frobenius matrix F_c, is only relevant for computing the initial approximations, but not for their refinement, based on updating the GC matrices. Let us briefly compare the remaining eigen-refiners.

The QR algorithm in [BGP03/05] and [BGP04] requires quadratic time per step and quadratic memory space for DPR1 matrices with general complex companion knots and thus becomes inferior as an eigen-refiner.

The IPI and the non-Hermitian Lanczos algorithms seem to be better candidate to be the GC eigen-refiner of choice because they require fewer ops per an iteration step than the Jacobi–Davidson and the Arnoldi algorithms (see Section 6.1). There is a potential competition from the popular root-finders applied as root-refiners. They have superlinear local convergence, like the IPI, but require extended precision of computing. Note another practical advantage of the IPI over the popular polynomial root-finders: for a real input matrix C the IPI can be easily extended to confine the computations to the real field; namely, we should just apply the power iteration step to the real matrix $(sI - C)^{-1}(s^*I - C)^{-1}$ where s and s^* denote two complex conjugate approximations to two complex conjugate eigenvalues of the matrix C.

Tables 3 and 4 display some relevant data on some most popular root-finders that approximate one root at a time and simultaneously k roots, respectively. Note the respective increase of the arithmetic cost per step in Table 4.

Table 3 does not cover the Jenkins–Traub algorithm in [JT70], [JT72]. The statistics of its application show that its performance is similar to the other root-finders in Table 3 (in fact they tend to be inferior in accuracy to the QR based root-finders), but the formal data on its ops count are hard to specify because this algorithm combines various other methods.

Among the modifications of the listed root-finders, we note the application of Muller's algorithm to the ratio $\frac{c(x)}{c'(x)}$ rather than to the polynimial $c(x)$. This increases the ops count per step to $4n + O(1)$ but substantially improves convergence according to our extensive tests.

Root-finder	References	ops/step	Order of convergence
Muller's	[T64, pages 210-213], [W68]	$2n+20$	1.84
Newton's	[M73], [MR75], [NAG88]	$4n$	2
Halley's	[OR00], [ST95]	$6n$	3
Laguerre's	[HPR77], [P64]	$6n+6$	3

Table 3: Four root-finders for a polynomial of degree n approximating one root at a time (One op in each of Muller's and Laguerre's algorithms stands for computing a square root.)

Root-finder	References	ops/step	Order of convergence
Durand–Kerner's	[W03], [D60] [K66]	$(4n-1)k$	2
Aberth's	[B-S63], [E67] [A73], [BF00]	$(7n-3)k$	3

Table 4: Two root–finders approximating simultaneously k roots of a polynomial of degree n

6.4 Flowcharts for root-finding with eigen-solving

To summarize, here is a flowchart of our root-finding for a polynomial $c(x)$ in (2.2).

- *Initial approximation.*

  Select and compute a GC matrix for $c(x)$ (cf. Section 5.1); select and apply an eigen-solver for this matrix to compute n distinct approximations to the roots of $c(x)$ (see Sections 6.1 and 6.2).

- *Updating the GC matrix and the approximations to the roots.*

  Choose the companion knots equal to the computed approximations to the roots and update the GC matrix (cf. Section 5.1).

  Apply the IPI n times with the shifts into the n current companion knots to improve the approximations to all eigenvalues.

  Repeat recursively until convergence.

In a modified version of this flowchart, we select a root-finder in Tables 3 or 4 and (unless this is again the IPI) substitute it for the IPI at the initial and/or updating stage.

The computation in both original and modified versions can include deflation (see Section 5.3).

Practical implementation of the flowchart should include numerical stabilization of the eigen-solvers by means of diagonal scaling (see the end of Section 3.5) and the root-finders by means of shifting the variable x to yield the equation $c_{n-1} = 0$ followed by scaling both the variable x and the polynomial $c(x)$, that is, by shifting to the polynomial $d^n c(x/d)$ where a scalar d is chosen to decrease the disparity in the magnitudes of the coefficients of the latter polynomial.

6.5 Divide-and-conquer root-refining and bounding the output errors

We can accelerate root-finding and eigen-solving if we can split a polynomial $c(x)$ into the product $\prod_{i=1}^{k} c_i(x)$ of $k > 1$ nonscalar polynomials $c_i(x)$ and repeat this step recursively (see [P01/02], [BP98], and the bibliography therein). Effective splitting algorithms in [S82], [K98], [P01/02], and [BGM02] compute the factors $c_i(x)$ in nearly optimal arithmetic and Boolean time provided we know some sufficiently wide root-free annuli on the complex plane that isolate the root sets of the factors $c_i(x)$ from each other (see also [C96] and [BP96] on some alternative splitting algorithms and [W69], [BJ76], [B83], [DM89], [DM90], and [VD94] on various applications to signal and image processing). The algorithms in [P01/02] compute the desired annuli also in nearly optimal time but are quite involved, which diminishes their practical value. For a large input class, however, the annuli are readily available as by-product of approximating the roots even with a low precision.

With the GC representations in Sections 3.2 and 3.4 we can bound the approximation errors and detect the basic root-free annuli for splitting based on the following result for the DPR1 and arrow-head matrices.

Theorem 6.1. *The union $\sum_{i=1}^{n} D_i$ (resp. $\sum_{i=1}^{n} \overline{D}_i$) contains all eigenvalues of the matrix C in equation (3.4) (resp. the matrix $\overline{C}$ in (3.10)) provided D_i (resp. $\overline{D}_i$) denote the discs $\{x:\ |x - s_i + d_i| \le \sum_{j\ne i} |u_j v_i|\}$ or $\{x:\ |x - s_i + d_i| \le \sum_{j\ne i} |u_i v_j|\}$, $i = 1, \dots, n$ (resp. the discs $\{x:\ |x - \overline{s}_1 + \overline{u}_1| \le \sum_{i=2}^{n} |\overline{u}_i|\}$, $\{x:\ |x - \overline{s}_j| \le |\overline{v}_j|\}$, $j = 2, \dots, n$, or the discs $\{x:\ |x - \overline{s}_1 + \overline{u}_1| \le \sum_{j=2}^{n} |\overline{v}_j|\}$, $\{x:\ |x - \overline{s}_i| \le |\overline{u}_i|\}$, $i = 2, \dots, n$). Moreover, if the union of any set of k discs D_i (resp. $\overline{D}_i$) is isolated from all remaining $n - k$ discs, then this union contains exactly k eigenvalues of the matrix C (resp. $\overline{C}$).*

Proof. The theorem (due to [E73] for DPR1 matrices) immediately follows from the Gerschgörin theorem [GL96, Theorem 7.2.1] applied to the matrices C and $\overline{C}$. □

We need $3n - 1$ ops to compute the radii of the discs $D_1, \dots, D_n$ (or just $2n - 1$ ops under the choice of parameters in Example 3.1), and we only need $n - 1$ ops to compute the radii of the discs $\overline{D}_1, \dots, \overline{D}_n$.

Similarity transforms into a DPR1 matrix (see Section 3.6) enable us to extend the estimates in Theorem 6.1 to the RBDPR1 matrices C in (3.6), and we can yield a similar extension from the arrow-head matrices.

All discs D_i (resp. $\overline{D}_i$) are isolated from each other for all i if the matrix C (resp. $\overline{C}$) has n distinct eigenvalues and if the values $|u_i|$ and $|v_i|$ (resp. $|\overline{u}_i|$ and $|\overline{v}_i|$) are small enough. In this case the disc radii serve as upper bounds on the errors of the computed approximations s_i (resp. $\overline{s}_i$) to the eigenvalues.

Finally recall that a proximity test at a point s for the roots of a polynomial $c(x) = \prod_{j=1}^{n}(x - z_j)$ defines a root-free disc $\{x : |x - s| < \min_j |z_j - z|\}$ and that such a proximity test is a by-product of the application of either of the IPI, Arnoldi, non-Hermitian Lanczos and Jacobi–Davidson algorithms to the matrix $sI - C_c$. Now if the latter disc covers the intersection of two discs D_h and D_i (resp. $\overline{D}_h$ and $\overline{D}_i$), then they are isolated from one another. This observation combined with Theorem 6.1 suggests a promising heuristic method for isolating the eigenvalues.

7 Extension to eigen-solving

We can extend our eigen-solvers for GC matrices to the matrices A for which we can compute the following scalars and vectors.

- the scalar $c_A(x) = \det(xI - A)$ for a scalar x
- the scalars $c'_A(x) = -\operatorname{trace}(xI - A)^{-1}c_A(x)$ and $c''_A(x)$ for a scalar x
- the vector $(xI - A)^{-1}\mathbf{v}$ for a vector $\mathbf{v}$
- the vector $A\mathbf{v}$ for a vector $\mathbf{v}$.

Furthermore, as soon as we have n values $c_A(x)$ at n distinct points $s_1, \ldots, s_n$ computed, we can compute GC matrices $C = C_c$ in Sections 3.2–3.4 for $c(x) = c_A(x)$ and apply our algorithms to compute some crude approximations $\tilde{z}_1, \ldots, \tilde{z}_n$ to the roots. The approximations are crude due to the rounding errors in computing the GC matrix, but we can apply the IPI or the algorithm in [P05] to the matrix A and the shift values $\tilde{z}_1, \ldots, \tilde{z}_n$ as refiners.

Moreover, we can compute some crude initial approximations to the roots without computing a GC matrix. We can apply the eigen-solvers in Table 2 as long as we can compute the listed vectors and we can apply the root-finders in Tables 3 and 4 as long as we can compute the listed scalars. In fact the Durand–Kerner's and Muller's algorithms only require the computation of the scalars $c_A(x)$, whereas the algorithm in [P05] requires no preliminaries.

For many important classes of matrices all or most of the listed scalars and vectors can be readily computed at a low cost. This is the case, e.g., for various structured (e.g., Toeplitz) matrices [P01, Chapter 5], for banded matrices B having a small bandwidth (e.g., tridiagonal matrices) or more generally, for matrices associated with graphs that have small separator famillies [LRT79], [GH90], [GS92], and [PR93].

8 Polynomial and secular equations

The polynomial equations $c(x) = 0$ are closely related to the secular equations, encountered in updating the singular value decomposition of a matrix, the solution of the least-squares constrained eigenproblem, invariant subspace computation, divide-and-conquer algorithms for the tridiagonal Hermitian eigenproblem, and the "escalator method" for matrix eigenvalues (see [G73], [M97], and the bibliography therein).

For a matrix C in (3.10), recall the characteristic equation $c_C(x) = 0$, rewrite it as $c_C(x) = (x + a)\overline{q}(x) - \sum_{i=2}^{n} \overline{d}_i \overline{q}_i(x) = 0$, for the scalar $a = \overline{u}_1 - \overline{s}_1$ and then divide it by $\overline{q}(x)$ to arrive at the secular equation

$$x + a - \sum_{i=2}^{n} \frac{\overline{d}_i}{x - \overline{s}_i} = 0,$$

whose roots are given by the eigenvalues of the matrix C in (3.10). Likewise, recall the Lagrange interpolation formula (used in our first proof of Theorem 3.1) and divide its both sides by $q(x)$ to arrive at the secular equation

$$1 + \sum_{i=1}^{n} \frac{d_i}{x - s_i} = 0,$$

whose roots are equal to the eigenvalues of the matrix C in (3.4). By allowing to scale the equation, we reduce the root-finding for any secular equation of the form

$$\alpha x + \beta + \sum_{i=1}^{k} \frac{d_i}{x - s_i} = 0$$

to solving the eigenproblem for the arrow-head or DPR1 matrices. This also enables simple reduction of the polynomial and secular equations to one another.

Appendix

A Simplification of root-finding

The efficiency of the known polynomial root-finders applied as root-refiners typically decreases where the roots are multiple. Since $\frac{c'(x)}{c(x)} = \sum_{i=1}^{k} \frac{m_i}{x - z_i}$ for $c(x) = \prod_{i=1}^{k} (x - z_i)^{m_i}$ where $z_1, \ldots, z_k$ are distinct, this suggests the application of root-finders to the rational function $\frac{c(x)}{c'(x)}$ or to the polynomial $\frac{c(x)}{g(x)}$ where $g(x) = \gcd(c', c)$ is the gcd of $c'(x)$ and $c(x)$. With approximate division by approximate gcds, we can also replace root clusters by their single simple representatives. (On computing approximate gcds, see [CGTW95], [P98/01], [GKMYZ04], [Za], [LYZ05], and the bibliography therein).

We can avoid computation of the gcd by applying the root-finders to the rational function $f(x) = \frac{c(x)}{c'(x)}$. The Börsch-Supan's root-finder [B-S63] (widely

Method	References	The highest order i of $r^{(i)}(\lambda)$ used	Order of local convergence
Muller's	[T64, pages 210-213], [W68]	0	1.84
Newton's modified	[M73], [MR75], [NAG88]	1	2
Halley's	[OR00], [ST95]	2	3
Laguerre's	[HPR77], [P64]	2	3
Laguerre's discrete	[DJLZ96], [DJLZ97], [Z99]	0	3

Table 5: Five root-finders for a function $r(\lambda)$

known as Aberth's or Ehrlich's [B96]) proceeds by recursively computing the values of this function. The iterative processes in Tables 3 and 4 can be reduced essentially to the recursive evaluation of $c^{(i)}(x)$ at the approximation points x for $i = 0, 1, \ldots, k$ for a small fixed integer k. Due to the affect of the poles of the function $\frac{c(x)}{c'(x)}$, the chances for divergence grow, but overall the convergence tends to be faster and more reliable when we apply Muller's method to $f(x)$ according to our tests with Muller's and Newton's root-finders, each applied to both $c(x)$ and $f(x)$.

References

[A73] O. Aberth, Iteration Methods For Finding All Zeros of a Polynomial Simultaneously, *Math. Comp.*, **27**, **122**, 339–344, 1973.

[B75] S. Barnett, A Companion Matrix Analogue for Orthogonal Polynomials, *Linear Algebra and Its Appl.*, **12**, **3**, 97–208, 1975.

[B96] D. A. Bini, Numerical Computation of Polynomial Zeros by Means of Aberth's Method, *Numerical Algorithms*, **13**, **3–4**, 179–200, 1996.

[B83] S. Barnett, *Polynomials and Linear Control Systems*, Marcel Dekker, New York, 1983.

[BDDRvV00] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[BBCD93] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. Van Der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1993.

[BF00] D. A. Bini, G. Fiorentino, Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder, *Numerical Algorithms*, **23**, 127–173, 2000.

[BGM02] D. A. Bini, L. Gemignani, B. Meini, Computations with Infinite Toeplitz Matrices and Polynomials, *Linear Algebra Appl.*, **343–344**, 2002.

[BGP02/04] D. A. Bini, L. Gemignani, V. Y. Pan, Inverse Power and Durand/Kerner Iteration for Univariate Polynomial Root-finding, *Computers and Mathematics (with Applications)*, **47**, **2/3**, 447–459, 2004. (Also Technical Report TR 2002 020, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2002.)

[BGP04] D. A. Bini, L. Gemignani, V. Y. Pan, Improved Initialization of the Accelerated and Robust QR-like Polynomial Root-finding, *Electronic Transactions on Numerical Analysis*, **17**, 195–205, 2004. Proc. version in *Proceedings of the Seventh International Workshop on Computer Algebra in Scientific Computing (CASC 04)*, St. Petersburg, Russia (July 2004), (edited by E. W. Mayr, V. G. Ganzha, E. V. Vorozhtzov), 39-50, Technische Univ. Mnchen, Germany, 2004.

[BGP03/05] D. A. Bini, L. Gemignani, V. Y. Pan, Fast and Stable QR Eigenvalue Algorithms for Generalized Companion Matrices and Secular Equation, *Numerische Math.*, **3**, 373–408, 2005. (Also Technical Report 1470, *Department of Math, University of Pisa*, Pisa, Italy, July 2003.)

[BJ76] G. E. P. Box, G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, Holden-Day, San Francisco, California, 1976.

[BP94] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[BP96] D. Bini, V. Y. Pan, Graeffe's, Chebyshev, and Cardinal's processes for splitting a polynomial into factors, *J. Complexity*, **12**, 492-511, 1996.

[BP98] D. Bini, V. Y. Pan, Computing Matrix Eigenvalues and Polynomial Zeros Where the Output Is Real, *SIAM Journal on Computing*, **27**, **4**, 1099–1115, 1998.

[B-S63] W. Börsch-Supan, A-posteriori Error Bounds for the Zeros of Polynomials, *Numerische Math.*, **5**, 380–398, 1963.

[C91] C. Carstensen, Linear Construction of Companion Matrices, *Linear Algebra Appl.*, **149**, 191–214, 1991.

[C96] J. P. Cardinal, On Two Iterative Methods for Approximating the Roots of a Polynomial, *Proceedings of AMS-SIAM Summer Seminar: Mathematics of Numerical Analysis: Real Number Algorithms* (J. Renegar, M. Shub, and S. Smale, editors), Park City, Utah, 1995. *Lectures in Applied Mathematics*, **32**, 165–188, American Mathematical Society, Providence, Rhode Island, 1996.

[CGTW95] R. M. Corless, P. M. Gianni, B. M. Trager, S. M. Watt, The Singular Value Decomposition for Polynomail Systems, *Proc. Intern. Symposium on Symbolic and Algebriac Computation* (ISSAC'95), 195–207, ACM Press, New York, 1995.

[CN94] D. Coppersmith, C. A. Neff, Roots of a Polynomial and Its Derivatives. *In Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'94), 271–279, ACM Press, New York, and SIAM Publications, Philadelphia, 1994.

[D60] E. Durand, Solutions numériques des équations algébriques, *Tome 1: Equations du type F(X)=0; Racines d'un polynôme*, Masson, Paris, 1960.

[DJLZ96] Q. Du, M. Jin, T. Y. Li, Z. Zeng, Quasi-Laguerre Iteration in Solving Symmetric Tridiagonal Eigenvalue Problems. *SIAM J. Sci. Comput.* **17, 6**, 1347–1368, 1996.

[DJLZ97] Q. Du, M. Jin, T. Y. Li, Z. Zeng, The Quasi-Laguerre Iteration. *Math. of Computation*. **66, 217**, 345–361, 1997.

[DM89] C. J. Demeure, C. T. Mullis, The Euclid Algorithm and Fast Computation of Cross-covariance and Autocovariance Sequences, *IEEE Trans. Acoust., Speech and Signal Processing*, **37**, 545–552, 1989.

[DM90] C. J. Demeure, C. T. Mullis, A Newton-Raphson Method for Moving-Average Spectral Factorization Using the Euclid Algorithm, *IEEE Trans. Acoust., Speech and Signal Processing*, **38**, 1697–1709, 1990.

[E67] L. W. Ehrlich, A Modified Newton Method for Polynomials, *Comm. ACM* **10**, 107–108, 1967.

[E73] L. Elsner, A Remark on Simultaneous Inclusions of the Zeros of a Polynomial by Gershgörin's Theorem, *Numerische Math.*, **21**, 425–427, 1973.

[EMP04] E. Z. Emiris, B. Mourrain, V. Y. Pan, Guest Editors, Algebraic and Numerical Algorithms, *Special Issue of Theoretical Computer Science*, **315, 2–3**, 307–672, 2004.

[F90] M. Fiedler, Expressing a Polynomial As the Characteristic Polynomial of a Symmetric Matrix, *Linear Algebra and Its Appl.*, **141**, 265–270, 1990.

[F01/02] S. Fortune, An Iterated Eigenvalue Algorithm for Approximating Roots of Univariate Polynomials, *J. of Symbolic Computation*, **33, 5**, 627–646, 2002. Proc. version in *Proc. Intern. Symp. on Symbolic and Algebraic Computation (ISSAC'01)*, 121–128, ACM Press, New York, 2001.

[G52/58] A. Gel'fond, *Differenzenrechnung*, Deutsher Verlag Der Wissenschaften, Berlin, 1958. (Russian edition: Moscow, 1952.)

[G73] G. H. Golub, Some Modified Matrix Eigenvalue Problems, *SIAM Rev.*, **15**, 318–334, 1973.

[GG03] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*, 2nd edition, Cambridge University Press, Cambridge, UK, 2003.

[GKMYZ04] S. Gao, E. Kaltofen, J. May, Z. Yang, S. Zhi, Approximate Factorization of Multivariate Polynomial via Differential Equations, *Proc. International Symposium on Symbolic and Algebraic Computaion (ISSAC'04)*, 167–174, ACM Press, New York, 2004.

[GH90] J. R. Gilbert, H. Hafsteinsson, Parallel Symbolic Factorization of Sparse Linear Systems, *Parallel Computing*, **14**, 151–162, 1990.

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

[GS92] J. R. Gilbert, R. Schreiber, Highly Parallel Sparse Cholesky Factorization, *SIAM J. on Scientific Computing*, **13**, 1151–1172, 1992.

[H71] A. S. Householder, Generalization of an Algorithm by Sebastiao e Silva, *Numerische Math.*, **16**, 375–382, 1971.

[H04] Z. Huang, The Convergence Ball of Newton's Method and the Uniqueness Ball of Equations under Hölder-type Continuous Derivatives, *Computers and Math. with Applications*, **47**, 247–251, 2004.

[HPR77] E. Hansen, M. Patrick, A Family of Root Finding Methods, *Numerische Math.*, **27**, 257–269, 1977.

[JT70] M. A. Jenkins, J. F. Traub, A Three-Stage Variable-Shift Iteration for Polynomial Zeros and Its Relation to Generalized Rayleigh Iteration, *Numerische Math.*, **14**, 252–263, 1969/1970.

[JT72] M. A. Jenkins, J. F. Traub, A Three-Stage Algorithm for Real Polynomials Using Quadratic Iteration, *SIAM J. Numer. Anal.*, **7**, 545–566, 1970.

[JV04] J. F. Jónsson, S. Vavasis, Solving Polynomials with Small Leading Coefficients, *SIAM J. on Matrix Analysis and Applications*, **26, 2**, 400–412, 2004.

[K66] I. O. Kerner, Ein Gesamtschrittverfahren zur Berechung der Nullstellen von Polynomen, *Numer. Math.* **8**, 290–294, 1966.

[K98] P. Kirrinnis, Polynomial Factorization and Partial Fraction Decomposition by Simultaneous Newton's Iteration. *J. of Complexity*, **14**, 378–444, 1998.

[LYZ05] B. Li, Z. Yang, L. Zhi, Fast Low Rank Approximation of a Sylvester Matrix by Structured Total Least Norm, *Journal JSSAC*, **11**, 165–174, 2005.

[LF94] M. Lang, B. C. Frenzel, Polynomial Root-Finding, *IEEE Signal Processing Letters*, **1**, **10**, 141–143, 1994.

[LRT79] R. J. Lipton, D. Rose, R. E. Tarjan, Generalized Nested Dissection, *SIAM J. on Numerical Analysis*, **16**, **2**, 346–358, 1979.

[M73] K. Madsen, A Root-finding Algorithm Based on Newton's Method, *BIT*, **13**, 71–75, 1973.

[M97] A. Melman, A Unifying Convergence Analysis of Second-Order Methods for Secular Equations, *Math. Comp.*, **66**, 333–344, 1997.

[McN93] J. M. McNamee, Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics*, **47**, 391–394, 1993.

[McN97] J. M. McNamee, A Supplementary Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics*, **78**, **1**, 1997.

[McN99] J. M. McNamee, An Updated Supplementary Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics*, **110**, 305–306, 1999.

[McN02] J. M. McNamee, A 2002 Updated Supplementary Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics*, **142**, 433–434, 2002.

[MP00] B. Mourrain, V. Y. Pan, Multivariate Polynomials, Duality and Structured Matrices, *J. of Complexity*, **16**, **1**, 110–180, 2000.

[MR75] K. Madsen, J. Reid, Fortran Subroutines for Finding Polynomial Zeros, Report HL75/1172 (C.13), *Computer Science and Systems Division*, A. E. R. E. Harwell, Oxford, 1975.

[MV95] F. Malek, R. Vaillancourt, Polynomial Zerofinding Iterative Matrix Algorithms, *Computers and Math. with Applications*, **29**,**1**, 1–13, 1995.

[MV95a] F. Malek, R. Vaillancourt, A Composite Polynomial Zerofinding Matrix Algorithm, *Computers and Math. with Applications*, **30**,**2**, 37–47, 1995

[NAG88] *NAG Fortran Library Manual*, Mark 13, Vol. **1**, 1988.

[NR94] Neff, C. A., Reif, J. H. An $O(n^{l+\epsilon})$ Algorithm for the Complex Root Problem, *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Scinece*, 540–547, IEEE Computer Society Press, Los Alamitos, California, 1994.

[OR00] J. M. Ortega, W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, SIAM, Philadelphia, 2000.

[P64] B. Parlett, Laguerre's Method Applied to the Matrix Eigenvalue Problem, *Math. of Computation*, **18**, 464–485, 1964.

[P92] V. Y. Pan, Complexity of Computations with Matrices and Polynomials, *SIAM Review*, **34**, **2**, 225–262, 1992.

[P95] V. Y. Pan, Optimal (up to Polylog Factors) Sequential and Parallel Algorithms for Approximating Complex Polynomial Zeros, *Proc. 27th Ann. ACM Symp. on Theory of Computing*, 741–750, ACM Press, New York, May, 1995.

[P96] V. Y. Pan, Optimal and Nearly Optimal Algorithms for Approximating Polynomial Zeros, *Computers and Math. (with Applications)*, **31**, **12**, 97–138, 1996.

[P98] V. Y. Pan, Some Recent Algebraic/Numerical Algorithms, *Electronic Proceedings of IMACS/ACA'98*, 1998.

http:www-troja.fjfi.cvut.cz/aca98/sessions/approximate/pan/

[P98/01] V. Y. Pan, Numerical Computation of a Polynomial GCD and Extensions, *Information and Computation*, **167**, **2**, 71–85, 2001. Proc. version in *Proc. of 9th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'98)*, 68–77, ACM Press, New York, and SIAM Publications, Philadelphia, 1998.

[P97] V. Y. Pan, Solving a Polynomial Equation: Some History and Recent Progress, *SIAM Review*, **39, 2**, 187–220, 1997.

[P00] V. Y. Pan, Approximating Complex Polynomial Zeros: Modified Quadtree (Weyl's) Construction and Improved Newton's Iteration, *J. of Complexity*, **16**, **1**, 213–264, 2000.

[P01] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

[P01/02] V. Y. Pan, Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *Journal of Symbolic Computations*, **33, 5**, 701–733, 2002. Proc. version in *Proc. International Symp. on Symbolic and Algebraic Computation (ISSAC 01),* 253-267, ACM Press, New York, 2001.

[P05] V. Y. Pan, Amended DSeSC Power Method for Polynomial Rootfinding, *Computers and Math. with Applications*, **49, 9-10**, 1515-1524, 2005.

[PHI98] M. S. Petković, D. Herceg, S. Ilić, Safe Simultaneous Methods for Polynomial Zeros, *Numerical Algorithms*, **17**, 313–331, 1998.

[PKMRTYCa] V. Y. Pan, M. Kunin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, W. Cao, Linking Arrow-head, DPR1, and TPR1 Matrix Structures, preprint, 2005. Proc. version (by V. Y. Pan) in Proc. of *Annual Symposium on Discrete Algorithms (SODA'05)*, 1069-1078, ACM Press, New York, and SIAM Publications, Philadelphia, 2005.

[PR93] V. Y. Pan, J. Reif, Fast and Efficient Parallel Solution of Sparse Linear Systems, *SIAM J. on Computing*, **22**, **6**, 1227–1250, 1993.

[S82] A. Schönhage, The Fundamental Theorem of Algebra in Terms of Computational Complexity, *Mathematics Department, University of Tübingen*, Germany, 1982.

[S98] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, 1998.

[SeS41] J. Sebastiao e Silva, Sur une Méthode d'Approximation Semblable a Celle de Graeffe, *Portugal Math.*, **2**, 271–279, 1941.

[ST95] T. R. Scavo, J. B. Thoo, On the Geometry of Halley's Method., *Amer. Math. Monthly*, **102**, 417–426, 1995.

[T64] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1963.

[VD94] P. M. Van Dooren, Some Numerical Challenges in Control Theory. In *Linear Algebra for Control Theory*, volume **62** of IMA Vol. Math. Appl., Springer, 1994.

[W03] K. Weierstrass, Neuer Beweis des Fundamentalsatzes der Algebra, *Mathematische Werker*, Tome III, Mayer und Mueller, Berlin, 251–269, 1903.

[W65] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[W68] V. Whitley, Certification of Algorithm 196: Muller's Method for Finding Roots of Arbitrary Function, *Comm. ACM*, **11**, 12–14, 1968.

[W69] G.T. Wilson, Factorization of the Covariance Generating Function of a Pure Moving-average Process, *SIAM J. Num. Anal.*, **6**, 1–7, 1969.

[WZ95] D. Wang, F. Zhao, The Theory of Smale's Point Estimation and Its Application, *J. of Comput. and Applied Math.*, **60**, 253–269, 1995.

[Z99] X. Zou, Analysis of the Quasi-Laguerre Method, *Numerische Math.*, **82**, 491–519, 1999.

[Za] Z. Zeng, The Approximate GCD of Inexact Polynomials, Part I: a Univariate Algorithm, preprint, 2004.