2007

TR-2007004: Additive Preconditioning, Eigenspaces, and the Inverse Iteration

Victor Y. Pan

Xiaodong Yan

Additive Preconditioning, Eigenspaces, and the Inverse Iteration *

Victor Y. Pan[1,3] and Xiaodong Yan[2]

[1] Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
victor.pan@lehman.cuny.edu

[2] Ph.D. Program in Computer Science
The City University of New York
New York, NY 10036 USA
xyan@gc.cuny.edu

[3] http://comet.lehman.cuny.edu/vpan/

Abstract

Previously we have showed that the computation of vectors in and bases for the null space of a singular matrix can be accelerated based on additive preconditioning and aggregation. Now we incorporate these techniques into the inverse iteration for computing the eigenvectors and eigenspaces of a matrix, which are the null vectors and null spaces of the same matrix shifted by its eigenvalues. According to our analysis and extensive experiments, our acceleration of every iteration step does not slow down the convergence.

Key words: Additive preconditioning, Eigenspaces, Inverse Iteration

*Supported by PSC CUNY Awards 66437-0035 and 67297-0036

1 Introduction

1.1 Computing in the null spaces, additive preconditioning, and extension to eigen-solving

Given an $n \times n$ matrix A of a rank $\rho < n$, suppose we seek its null vector or null basis, that is, a vector in or a basis for the (right) null space $RN(A)$. One can obtain them via computing the SVD, QRP or PLU factorizations [1]–[5], or the inverse of a nonsingular $\rho \times \rho$ submatrix of matrices A or MAN for some nonsingular multipliers M and N.

In [6] we study an alternative approach based on *addititive preprocessing* of the input matrix A. Hereafter M^H denotes the Hermitian (that is, complex conjugate) transpose of a matrix M, which is just its transpose M^T for a real matrix M, and "A-" and "APP" abbreviate "additive" and "additive preprocessor", respectively.

Define two *generators* U and V of size $n \times r$, suppose an APP UV^H has rank $r = n - \rho$ equal to the nullity of the matrix A, and let the A-modification $C = A + UV^H$ have full rank. Then the columns of the *null aggregate* $C^{-1}U$ form a null basis for the matrix A, and so we call the matrix $C^{-1}U$ a *null matrix basis* for the matrix A and call the computation of this basis the *Null Aggregation* (cf. Theorem 2.1).

According to the analysis and extensive experiments in [7], A-preprocessing of an $n \times n$ ill conditioned input matrix A with a random well conditioned and properly scaled APPs UV^H of a rank r (such that the ratio $||UV^H||_2/||A||_2$ is neither large nor small) is likely to yield an A-modification $C = A + UV^H$ with the condition number of the order of $\sigma_{r+1}(A)/\sigma_n(A)$ where $\sigma_j(A)$ denotes the jth largest singular value of the matrix A. If $\sigma_1(A) \gg \sigma_{r+1}(A)$, then our A-preprocessing is likely to be *A-preconditioning*, that is, likely to decrease the condition number substantially.

Furthermore, very weak randomization, which we call *pseudo randomization*, is actually sufficient, allowing us to choose structured and/or sparse APPs [7, Examples 4.1–4.6]. This is an important advantage where the ranks r are large.

Since our techniques preserve matrix structure and improve conditioning, they enable effective application of the GMRES and Conjugate Gradient algorithms [1, Section 10.2], [5], [8]–[10] in these computations.

1.2 The cases of rank-one and rank-two modifications

Let us examine more closely the simplest case where $r = 1$. Given a normalized $n \times n$ matrix A of rank $n - 1$ (with $||A||_2 = 1$), suppose we seek its normalized null vector $\mathbf{y}$. Let a normalized rank-one APP $\mathbf{u}\mathbf{v}^H$ define a nonsingular A-modification $C = A + \mathbf{u}\mathbf{v}^H$. Then $\mathbf{y} = \tilde{\mathbf{y}}/||\tilde{\mathbf{y}}||_2$, $\tilde{\mathbf{y}} = C^{-1}\mathbf{u}$, so that the problem is essentially reduced to solving a nonsingular linear system of equations $C\tilde{\mathbf{y}} = \mathbf{u}$.

This reduction little affects the conditioning of the problem. Indeed, according to our study in [7], for a pair of (pseudo) random normalized vectors $\mathbf{u}$ and $\mathbf{v}$, we can expect that the ratios $\sigma_n(C)/\sigma_{n-1}(A)$ and therefore $\mathrm{cond}_2 C/\mathrm{cond}_2 A$

are neither large nor small, so that the A-modification C is well conditioned if and only if so is the matrix A.

Now suppose the ratio $\sigma_1(A)/\sigma_{n-2}(A)$ is not large but $\sigma_{n-2}(A) >> \sigma_{n-1}(A)$. Then for (pseudo) random normalized APPs $\mathbf{u}\mathbf{v}^H$ and $\mathbf{u}_1\mathbf{v}_1^H$ and A-modifications

$$C_1 = A + \mathbf{u}_1\mathbf{v}_1^H, \quad C = C_1 + \mathbf{u}\mathbf{v}^H = A + UV^H, \quad U = (\mathbf{u},\ \mathbf{u}_1), \quad V = (\mathbf{v}, \mathbf{v}_1),$$

we can expect that the ratios $\sigma_1(C_1)/\sigma_{n-1}(C_1)$ and $\sigma_1(C)/\sigma_n(C)$ are not large, even though $\sigma_{n-1}(C_1) >> \sigma_n(C_1)$.

In this case, rank-one modification $A \leftarrow C_1 = A + \mathbf{u}_1\mathbf{v}_1^H$ avoids singularity, but reduces the null vector computation to the solution of an ill conditioned linear system $C_1\mathbf{y} = \mathbf{u_1}$.

We can fix this defect by applying rank-two modification. The range (that is, the column span) of the $n \times 2$ matrix $C^{-1}U$ contains the vector $\mathbf{y}$, so that $\mathbf{y} = C^{-1}U\mathbf{x}$ where $AC^{-1}U\mathbf{x} = \mathbf{0}$. This reduces the search for a null vector of an $n \times n$ matrix A to the similar problem for its $n \times 2$ *null aggregate* $AC^{-1}U$ and, if the $n \times 2$ matrix U has full rank two, then to the null vector problem for the 2×2 *Schur aggregate* $G = I_r - V^HC^{-1}U$ because $AC^{-1}U = UG$. We refer the reader to [11] on the computations with the Schur aggregates.

We call this technique the Null Aggregation. It is a natural descendant of the *aggregation methods* in [12], which in the 1980s evolved into the *Algebraic Multigrid*. To obtain a better insight into the nature of aggregation, one can also compare our present approach with *trilinear aggregating* in [13]. The latter technique has been an indispensible ingredient in the design of the currently fastest algorithms for $n \times n$ matrix multiplication, both the ones in [14], which are the fastest known algorithms for immense n, and the ones in [13], [15], which are the fastest known algorithms for dimensions n from 20 to, say, 2^{20} and which have efficient numerical implementations in [16], [17].

1.3 Extension to eigen-solving and the related works

The eigenspace of a matrix M associated with its eigenvalue λ is just the null space of the matrix $A = \lambda I_n - M$, and so the above approach can be incorporated into the known eigen-solvers, in particular the inverse iteration. In the present paper we elaborate upon this incorporation. Our study can be readily extended to shift-and-invert enhancement of the Lanczos, Arnoldi, Jacobi–Davidson, and other effective eigen-solvers. Our analysis and extensive experiments show that our modification does not affect the convergence rate, even though it improves conditioning of every iteration step. We demonstrate the power of our approach in its initial version. We suspect that this power can be substantially enhanced.

Small-rank modification is a known tool for decreasing the rank of a matrix [18], [19], fixing its small-rank deviations from the Hermitian, positive definite, and displacement structures, and supporting the divide-and-conquer algorithms for approximating the eigenvalues and eigenvectors of Hermitian tridiagonal matrices [1, Section 8.5.4], [3, Section 3.2], [20], [21]. (We refer the reader to [22] on some serious difficulties with the extension of this approach to the non-Hermitian eigenproblem.) These techniques, however, have not been directed

to preconditioning the input matrix, which is the main issue of our application of such modifications to eigen-solving.

We first introduced our techniques of A-preconditioning to accelerate the inverse power iteration, which we applied to polynomial root-finding (see [23] and Appendix B).

1.4 Organization of the paper

We organize our presentation as follows. In the next section we recall the basic theorem from [6]. In Section 3 and also in the next section, we state some definitions. (We present and analyze our algorithms for the general case of matrix polynomials, but at least on first reading one can assume just the classical algebraic eigenproblem [1], [3], [24].) In Section 4 we observe some affects of the A-preprocessing on the eigenvectors. In Section 5 we briefly review the inverse iteration for eigenproblem and sketch our modification. In Sections 6 and 7 we describe our rank-one and varaible-rank modifications of this iteration, respectively. In Section 8 we show their local quadratic convergence. In Section 9 we present the results of our numerical experiments. In Section 10 we list some natural extensions of our algorithms. In Appendix A we theoretically study the affects of A-preconditioning on the eigensystem. In Appendix B we comment on applications to polynomial root-finding.

The tests have been designed by both coauthors, and the second coauthor has actually run the tests. Otherwise the paper is due to the first author.

2 The basic theorem

Designing our algorithms for null bases and null vectors and extending them to eigenvectors, we rely on the following theorem from [6]. Hereafter $L(A)$ and $N(A) = RN(A)$ denote the left and (right) null spaces of a matrix A, respectively; $\text{range}(M)$ is the range of a matrix M, that is its column span; $\text{rnul}\, A = \text{nul}\, A = n - \text{rank}\, A$ is the (right) nullity of an $m \times n$ matrix A; $\text{lnul}\, A = m - \text{rank}\, A$ is its left nullity, and C^+ denotes the Moore–Penrose generalized inverse of a matrix C (cf. definitions in [1, Section 5.5.4]).

Theorem 2.1. *([6, Theorem 3.1].) Suppose $m \geq n$ and for an $m \times n$ matrix A of a rank ρ and a pair of two matrices U of size $m \times r$ and V of size $n \times r$, the matrix $C = A + UV^H$ has full rank n. Then*

$$r \geq \text{rank}\, U \geq n - \rho = \text{rnul}\, A, \tag{2.1}$$

$$N(A) \subseteq \text{range}(C^+U). \tag{2.2}$$

Furthermore if

$$r = \text{rank}\, U = n - \rho = \text{rnul}\, A, \tag{2.3}$$

then we have

$$N(A) = \text{range}(C^+U), \tag{2.4}$$

$$V^H C^+ U = I_r, \tag{2.5}$$

and if $\mathbf{y} \in N(A)$*, then*

$$\mathbf{y} = C^+ U(V^H \mathbf{y}). \tag{2.6}$$

Corollary 2.1. *Under the assumptions of Theorem 2.1 let equations (2.1) and (2.2) hold. Then* $N(A) = \text{range}(C^+UX)$ *if* X *is a matrix bases for the null space* $N(AC^+U)$*.*

Remark 2.1. *For* $C = \lambda I_r - M$ *the matrix equation (2.5) can be rewritten as* $I_r - V^H(\lambda I - M)^{-1}U = 0$*. Its left-hand side is a special case of the expressions* $B - V^H(\lambda I - M)^{-1}U$*, called the* realizations of rational matrix functions*, naturally interpreted as the Schur complements of the blocks* $\lambda I - M$ *in the* 2×2 *block matrix* $\begin{pmatrix} \lambda I - M & U \\ V^H & B \end{pmatrix}$ *and extensively used in linear systems and control [28].*

Our eigen-solvers rely on equations (2.3)–(2.6) for $m = n$. Further refined study could rely on equations (2.1) and (2.2) for $m = n$ complemented with the equation $N(A) = \text{range}(C^+UX)$ for X such that $\text{range}\, X = N(G)$, $G = I_r - V^H C^+ U$.

3 Matrix polynomials and the algebraic eigenproblem

We rely on the customary definitions for matrix computations in [1]–[3], [25] and on their extension to matrix polynomials $A(\lambda) = \sum_{i=0}^{k} A_i \lambda^i$ where $A_0, \dots, A_k$ are matrices of the same size and where the norm $||A(\lambda)||$ is the sum of the norms $||A_0||, \dots, ||A_k||$ or their another fixed positive function.

The eigenvalues of a matrix polynomial $A(\lambda)$ of a positive degree m are the roots of the characteristic polynomial $c_A(\lambda) = \det A(\lambda)$. The eigenvalues of a scalar matrix A are the eigenvalues of the linear matrix polynomial $A(\lambda) = \lambda I - A$.

The (algebraic) multiplicity $m(\mu)$ of an eigenvalue μ of $A(\lambda)$ is the multiplicity of the root μ of the polynomial $c_A(\lambda)$.

An eigenvalue μ of $A(\lambda)$ is associated with the left and right eigenspaces $LN(A(\mu))$ and $N(A(\mu))$ made up of its associated left and right eigenvectors, respectively. It has the left and right *geometric multiplicities* $l.g.m._A(\mu) = \text{lnul}\, A(\mu)$ and $r.g.m._A(\mu) = g.m._A(\mu) = \text{rnul}\, A(\mu)$, respectively.

To a fixed set $\{\lambda_1, \dots, \lambda_h\}$ of the eigenvalues of $A(\lambda)$ we associate the left and right invariant eigenspaces $LN(A)$ and $N(A)$ of the matrix $A = \prod_{i=1}^{h} A(\lambda_i)$.

4 APPs and the eigenvectors: preliminary observations

The eigenspaces of a matrix polynomial $A(\lambda)$ associated with its eigenvalue $\lambda = \mu$ are precisely the null spaces $LN(A(\mu))$ and $RN(A(\mu))$. Therefore, the algorithms in [6] can be applied to the respective eigen-computations. Theorem 2.1 enables us to express the eigenvectors associated with the eigenvalue μ via linear systems of equations with the matrices $C(\mu) = A(\mu) + U(\mu)V(\mu)^H$. Let us specify these expressions in the simple case of a rank-one modification of a diagonalizable matrix polynomial $A(\lambda)$.

Theorem 4.1. *Suppose G_A is a nonsingular $n \times n$ matrix, whereas*

$$A(\lambda) = G_A D_A(\lambda) G_A^{-1} \ , \quad D_A(\lambda) = \mathrm{diag}(p_{A,i}(\lambda)(\lambda - \lambda_{A,i}))_{i=1}^n \ , \quad \text{and} \quad p_{A,i}(\lambda)$$

for $i = 1, \dots, n$ are scalar polynomials in λ (so that $\lambda = \lambda_{A,i}$ are the eigenvalues of the matrix polynomial $A(\lambda)$). Furthermore, let $\mathbf{u}(\lambda)$ and $\mathbf{v}(\lambda)$ be a pair of n-dimensional vectors or vector polynomials in λ and let $C(\lambda) = A(\lambda) + \mathbf{u}(\lambda)\mathbf{v}^H(\lambda)$ such that the matrices $C = C(\lambda_{A,i})$ are nonsingular, for $i = 1, \dots, n$. Write $\mathbf{u} = \mathbf{u}(\lambda_{A,i})$ and $\mathbf{v} = \mathbf{v}(\lambda_{A,i})$, for $i = 1, \dots, n$. Then the vectors $G_A\mathbf{e}_i = (\mathbf{v}^H G_A \mathbf{e}_i) C^{-1}\mathbf{u}$ and $\mathbf{e}_i^H G_A^{-1} = (\mathbf{e}_i^H G_A^{-1}\mathbf{u})\mathbf{v}^H C^{-1}$ are the eigenvectors associated with the eigenvalues $\lambda_{A,i}$.

Proof. The first claim of the theorem (about the vectors $G_A\mathbf{e}_i$) is supported by equations (2.3), (2.4), and (2.6) for $\mathbf{y} = G_A\mathbf{e}_i$, $U = \mathbf{u}$ and $V = \mathbf{v}$. The second claim is proved similarly. □

The eigenvectors associated with a fixed eigenvalue of a matrix or a matrix polynomial are the solutions of some homogeneous singular linear systems of equations. Theorem 4.1 and the algorithms in [6] enable us to compute these vectors by solving nonsingular and sufficiently well conditioned linear systems of equations. In the next sections we incorporate similar techniques to refine some popular eigen-solvers.

5 Inverse iteration and our modifications: an overview

The solution of an ill conditioned linear system of equations is the basic operation in some popular eigen-solvers such as the inverse power iteration, the Jacobi–Davidson algorithm, and the Arnoldi and Lanczos algorithms with the shift-and-invert enhancements. The same operation is encountered at the deflation stage of the QR algorithm. As we have already recalled, random or pseudo random, well conditioned, and consistently scaled APCs are likely to improve the conditioning of such linear systems. Next we analyze this approach for the inverse power iteration, which is a classical tool for the refinement of a crude solution to the algebraic eigenproblem [1], [3], [24]–[27] and has block versions,

called the inverse orthogonal iteration [1, page 339] and the inverse Rayleigh–Ritz subspace iteration [3, Section 6.1]. We use the respective abbreviations *IPI* and *IR–RI*.

Somewhat counter-intuitively, the IPI produces an accurate eigenvector as the solution of an ill conditioned linear systems of equations. This is not completely painless, however. In [25] the exposition of the inverse power iteration is concluded with the following sentence: "... inverse iteration does require a factorization of the matrix $A-\delta I$, making it less attractive when this factorization is expensive." Furthermore, since the matrix $A(\lambda)$ is ill conditioned near its eigenvalues λ, we loose a chance to combine the IPI and IR-RI processes with the iterations of the CG/GMRES type, which would enable us to approximate the eigenvalues of large sparse matrices lying in the middle of their spectra. We overcome both of these deficiencies by applying A-preconditioning (and involving neither M-preconditioning nor the Schur Aggregation).

Recall that for a matrix polynomial $A(\lambda)=\sum_{i=0}^{m} A_i\lambda^i$, we seek its eigenpairs (λ, Y) such that λ is a scalar, $\det A(\lambda)=0$, and Y is a unitary matrix basis for the null space $N(A(\lambda))$.

Given a close approximation $\tilde{\lambda}$ to a geometrically simple eigenvalue λ of $A(\lambda)$ and a generally crude normalized approximation $\tilde{\mathbf{y}}$ to an associated eigenvector $\mathbf{y}$, the IPI recursively alternates updatings of the scalar $\tilde{\lambda}$ and the vector $\tilde{\mathbf{y}}$ according to the mappings $\{\tilde{\lambda} \leftarrow$ a root of the equation $\mathrm{trace}(\mathbf{y}^H A(\tilde{\lambda})\mathbf{y})=0\}$ and $\{\tilde{\mathbf{y}} \leftarrow A^{-1}(\tilde{\lambda})\tilde{\mathbf{y}}/||A^{-1}(\tilde{\lambda})\tilde{\mathbf{y}}||_2\}$. (The root above turns into the Rayleigh quotient $\tilde{\mathbf{y}}M\tilde{\mathbf{y}}$ in the classical case where $A(\lambda)=\lambda I-M$ and $||\mathbf{y}||_2=1$.) The process stops where a fixed tolerance value exceeds the residual norm $||A(\lambda)\mathbf{y}||_2$.

For $\tilde{\lambda}\approx\lambda$ the matrix $A(\tilde{\lambda})$ is ill conditioned, but we can reduce updating the vector $\tilde{\mathbf{y}}$ to solving a linear system $C(\tilde{\lambda})\tilde{\mathbf{y}}=\mathbf{u}$ with a preconditioned coefficient matrix $C(\tilde{\lambda})=A(\tilde{\lambda})+\mathbf{u}\mathbf{v}^H$. Here the APP is generated by a pair of (pseudo) random normalized vectors $\mathbf{u}$ and $\mathbf{v}$ (cf. [7, Examples 4.1–4.6]). If the matrix polynomial $A(\lambda)$ has no small positive singular values, then according to our study in [7] we can expect that the matrix polynomial $C(\tilde{\lambda})$ is well conditioned. In this case we stabilize the IPI numerically by incorporating the computation of the approximation vectors $\tilde{\mathbf{y}}=C^{-1}(\tilde{\lambda})\mathbf{u}$. Apart from this, we stay with essentially the same iterative process and extend the customary study of the convergence and arithmetic cost. Indeed, if $\tilde{\lambda}\approx\lambda$, then the vector $C^{-1}(\tilde{\lambda})\mathbf{u}$ is close to a vector $\mathbf{y}\in N(A)$. We can immediately deduce this from equation (2.4) or from the equations $C(\lambda)\mathbf{y}=b\mathbf{u}$ for $\mathbf{y}\in N(A)$ and $b=\mathbf{v}^H\mathbf{y}$.

Studying the resulting algorithms, we write $||\cdot||_q$ for $q=2$ or $q=F$ to denote the 2-norm or the Frobenius norm of a matrix, respectively, write $\mathbf{y}$ instead of $\mathbf{u}$, and recursively update the vector $\mathbf{y}$ by over-writing it with the vector $C^{-1}(\tilde{\lambda})\mathbf{y}$ where $C(\tilde{\lambda})=A(\tilde{\lambda})$ if the matrix $A(\tilde{\lambda})$ is well conditioned and $C(\tilde{\lambda})=A(\tilde{\lambda})+\mathbf{y}\mathbf{v}^H$ otherwise.

6 Inverse iteration with APPs of rank one

Algorithm 6.1. Inverse iteration with APPs of rank one.

INPUT: *a matrix polynomial $A(\lambda)$, an approximation $\tilde{\lambda}$ to its eigenvalue λ, two positive values τ and k, $q = 2$ or $q = F$, and a Subroutine LIN·SOLVE for solving a nonsingular and well conditioned linear system of equations.*

OUTPUT: *either FAILURE, or PROBABLY G·MULTIPLE/CLUSTERED, or an approximation $(\lambda_{final}, \mathbf{y}_{final})$ to an eigenpair of $A(\lambda)$ such that $||A(\lambda_{final})\mathbf{y}_{final}||_2 \le \tau ||A(\lambda)||_q$.*

INITIALIZATION: *Set $COUNTER \longleftarrow 0$, $\sigma \longleftarrow ||A(\lambda)||_F$, $\sigma(\tilde{\lambda}) \longleftarrow ||A(\tilde{\lambda})||_q$, and $(\mathbf{v}, \mathbf{y}) \longleftarrow$ a pair of normalized random or pseudo random vectors.*

COMPUTATIONS:

1. *If $COUNTER > k$, output FAILURE and stop.*
   Otherwise $C(\tilde{\lambda}) \longleftarrow A(\tilde{\lambda})$ and apply Subroutine LIN·SOLVE to compute the vector $C^{-1}(\tilde{\lambda})\mathbf{y}$.
2. *If this application fails (that is, if the matrix $C(\tilde{\lambda})$ is singular and/or ill conditioned), then $C(\tilde{\lambda}) \longleftarrow (1/\sigma(\tilde{\lambda}))C(\tilde{\lambda}) + \mathbf{y}\mathbf{v}^H$ and apply Subroutine LIN·SOLVE to compute the vector $C^{-1}(\tilde{\lambda})\mathbf{y}$. If this application fails (that is, if the matrix $C(\tilde{\lambda})$ is still singular and/or ill conditioned), then output PROBABLY G·MULTIPLE/CLUSTERED and stop.*
3. *Otherwise $\mathbf{y} \longleftarrow \frac{C^{-1}\mathbf{y}}{||C^{-1}\mathbf{y}||_2}$ (approximate a normalized eigenvector).*
4. *$\tilde{\lambda} \longleftarrow$ a root $\tilde{\lambda}$ that satisfies the equation $\mathbf{y}^H A(\tilde{\lambda})\mathbf{y} = 0$ and minimizes the residual norm $||A(\tilde{\lambda})\mathbf{y}||_2$. ($\tilde{\lambda} = \frac{\mathbf{y}^H M\mathbf{y}}{\mathbf{y}^H N\mathbf{y}}$ if $A = \lambda N - M$.) Update the matrix $A(\tilde{\lambda})$ for the updated value of $\tilde{\lambda}$.*
5. *$\sigma(\tilde{\lambda}) \longleftarrow ||A(\tilde{\lambda})||_q$. ($\sigma(\tilde{\lambda}) \longleftarrow \sigma(\tilde{\lambda}) + \tilde{\lambda}_{new} - \tilde{\lambda}_{old}$ if $q = 2$ and $A = \lambda I - M$.) If $||A(\tilde{\lambda})\mathbf{y}||_2 \le \sigma\tau$ (that is, if the residual norm is small enough), output $\lambda_{final} = \tilde{\lambda}$, $\mathbf{y}_{final} = \mathbf{y}$ and stop. Otherwise set $COUNTER \longleftarrow COUNTER + 1$ and go to Stage 1.*

7 Inverse iteration with APPs of adjusted ranks

Algorithm 6.1 outputs PROBABLY G·MULTIPLE/CLUSTERED if the Subroutine LIN·SOLVE fails for both coefficient matrices $A(\tilde{\lambda})$ and $C(\tilde{\lambda})$. This can occur either because the vectors $\mathbf{v}$ and/or $\mathbf{y}$ lie in or near the ranges of the matrices $A(\lambda)^H$ and/or $A(\lambda)$, respectively (although such a case is unlikely for (pseudo) random vectors $\mathbf{v}$ and $\mathbf{y}$ and singular matrices $A(\lambda)$), or because λ is a geometrically multiple eigenvalue of the matrix polynomial $A(\lambda)$ or lies near another eigenvalue. Below we modify Algorithm 6.1 to approximate such eigenvalues λ and the associated eigenspaces. We just keep adding the outer products $\mathbf{y}\mathbf{v}^H$ of pairs of random or pseudo random vectors $\mathbf{y}$ and $\mathbf{v}^H$ to the matrix $C(\tilde{\lambda})$ until it becomes nonsingular and well conditioned. The resulting

algorithm (specified below) can be viewed as a modification of the IPI/IR–RI that for $\tilde{\lambda} \approx \lambda_{A,i}$ employs APPs.

Alternatively, we could have added a random or pseudo random, well conditioned, and suitably scaled APC UV^H of a larger rank r and then recursively repeated this stage decreasing the integer r until we obtain a nonsingular and well conditioned matrix $C(\mu) = A(\mu) + UV^H$ such that $||A(\mu)C^{-1}(\mu)U||_2 \leq \tau||A(\mu)||_q$.

As in the classical IPI/IR-RI iteration [3, Section 4.4], if λ represents a cluster of eigenvalues separated from all other eigenvalues, then one can expect convergence to this cluster.

Algorithm 7.1. Inverse iteration with APPs of adjusted ranks.

INPUT: *as in Algorithm 6.1.*

OUTPUT: *either FAILURE or an approximation* $(\lambda_{final}, Y_{final})$ *to an eigenpair of* $A(\lambda)$ *such that* $||A(\lambda_{final})Y_{final}||_2 \leq \tau||A(\lambda)||_q$.

INITIALIZATION: $COUNTER \longleftarrow 0$, $\sigma \longleftarrow ||A(\lambda)||_F$, $i \longleftarrow 0$, $\sigma(\tilde{\lambda}) \longleftarrow ||A(\tilde{\lambda})||_q$, $V_0 \longleftarrow ()$, $Y_0 \longleftarrow ()$ *where* $()$ *denotes the* $n \times 0$ *empty matrix,* $Y_1 \longleftarrow$ *an* $n \times 1$ *(pseudo) random unitary matrix.*

COMPUTATIONS:

1. *If* $COUNTER > k$, *output FAILURE and stop. Otherwise* $C(\tilde{\lambda}) \longleftarrow (1/\sigma(\tilde{\lambda}))A(\tilde{\lambda}) + Y_iV_i^H$, *apply the Subroutine LIN·SOLVE to compute the matrix* $C^{-1}(\tilde{\lambda})Y_{\mu(i)}$ *where* $\mu(i) = 1$ *if* $i = 0$, $\mu(i) = i$ *otherwise.*
2. *If this application fails (that is, if the matrix* $C(\tilde{\lambda})$ *is singular and/or ill conditioned), then set* $(\mathbf{v}, \mathbf{y}) \longleftarrow$ *a pair of normalized random or pseudo random vectors such that* $\mathbf{v}^HV_i = \mathbf{0}$ *if* $i > 0$, $V_{i+1} \longleftarrow (V_i, \mathbf{v})$, $Y_{i+1} \longleftarrow (Y_i, \mathbf{y})$ *(so that* $Y_{i+1}V_{i+1}^H = Y_iV_i^H + \mathbf{y}\mathbf{v}^H$*),* $i \longleftarrow i + 1$, $COUNTER \longleftarrow COUNTER + 1$, *and go to Stage 1.*
3. $Y \longleftarrow$ *the Q-factor in the QR factorization of the matrix* $C^{-1}(\tilde{\lambda})Y_{\mu(i)}$ *or a properly truncated Q-factor in a rank revealing QRP factorization of this matrix [1, Sections 5.2–5.4], [2, Chapters 4 and 5].*
4. $\tilde{\lambda} \longleftarrow$ *a root* $\tilde{\lambda}$ *that satisfies the equation* $\mathrm{trace}(Y^HA(\tilde{\lambda})Y) = 0$ *and minimizes the residual norm* $||A(\tilde{\lambda})Y||_2$. *(*$\tilde{\lambda} = \mathrm{trace}\,\frac{Y^HMY}{Y^HNY}$ *if* $A = \lambda N - M$*.) Update the matrix* $A(\tilde{\lambda})$ *for the updated value of* $\tilde{\lambda}$.
5. $\sigma(\tilde{\lambda}) \longleftarrow ||A(\tilde{\lambda})||_q$. *(*$\sigma(\tilde{\lambda}) \longleftarrow \sigma(\tilde{\lambda}) + \tilde{\lambda}_{new} - \tilde{\lambda}_{old}$ *if* $q = 2$ *and* $A = \lambda I - M$*.) If* $||A(\tilde{\lambda})Y||_2 \leq \sigma\tau$ *(that is, if the residual norm is small enough), output* $\lambda_{final} = \tilde{\lambda}$, $Y_{final} = Y$ *and stop. Otherwise set* $COUNTER \longleftarrow COUNTER + 1$, $Y_{\mu(i)} \longleftarrow Y$, *and go to Stage 1.*

Remark 7.1. *By applying Algorithms 6.1 and/or 7.1 to the matrix polynomial* $A^H(\lambda)$, *we approximate its right eigenvectors, which are the left eigenvectors of the matrix polynomial* $A(\lambda)$ *associated with the same eigenvalues.*

Remark 7.2. *In Algorithms 6.1 and 7.1 one can choose, e.g., random or pseudo random vectors* $\mathbf{v}$ *or write* $\mathbf{v} = \mathbf{y}$.

Remark 7.3. *For bad choices of the vectors* $\mathbf{v}$ *and* $\mathbf{y}$ *in Algorithm 7.1, the parameter* i *can exceed the rank of the matrix* $Y_i V_i^H$. *For (pseudo) random vectors* $\mathbf{v}$ *and* $\mathbf{y}$, *this degeneration occurs rarely; moreover, the QR factorizations at Stage 3 can fix it.*

8 Perturbations and errors in the modified inverse iteration

Although iteration steps of Algorithms 6.1 and 7.1 are better conditioned and thus computationally simpler, they yield about the same approximations to the eigenspaces of a matrix polynomial $A(\lambda)$ as the IPI and the IR–RI do. So we can extend the extensive analysis of the latter iterations from [1], [3], [24]–[27], and the bibliography therein. Moreover, we can slightly simplify this analysis because we can involve the matrix $C^{-1}(\mu)$ even where $\lambda = \mu$ is an eigenvalue of the matrix polynomial $A(\lambda)$. Let us estimate the errors to show local quadratic convergence of Algorithms 6.1 and 7.1 for the classical algebraic eigenproblem, where

$$A = A(\lambda) = \lambda I - M, \ \tilde{A} = A(\tilde{\lambda}) = \tilde{\lambda} I - M, \tag{8.1}$$

and the algorithms recursively refine approximations $\tilde{\lambda}$ to an eigenvalue λ and $\tilde{Y}$ to a matrix basis Y for the associated eigenspace.

We first express the errors in the Rayleigh quotients via the eigenvectors errors (without assuming equations (8.1)).

Theorem 8.1. *Let* $\tilde{Y}$ *and* Y *be* $n \times k$ *matrices and write* $\Delta = \tilde{Y} - Y$. *Then for an* $n \times n$ *matrix* A *we have* $\tilde{Y}^H A \tilde{Y} - Y^H A Y = \Delta^H A Y + Y^H A \Delta + \Delta^H A \Delta$.

Next we express the residual $\tilde{C}^{-1}\tilde{Y}$ via the input errors.

Theorem 8.2. *Let* Y *be a unitary* $n \times k$ *matrix basis for the null space* $N(A)$ *of an* $n \times n$ *matrix* A. *Let a pair of matrices* $\tilde{A}$, $\tilde{Y}$ *approximate the pair of* A *and* Y. *Write* $C = A + \tilde{Y} V^H$, $\tilde{C} = \tilde{A} + \tilde{Y} V^H$, $E = \tilde{C} - C = \tilde{A} - A$, $\Delta = \tilde{Y} - Y$ *for an* $n \times k$ *matrix* V *such that the matrices* $B = V^H Y$ *and* $\tilde{C}$ *are nonsingular. (Observe that* $B = I_k$ *if* $V = Y$.*) Then we have*

a) $\tilde{C}^{-1}\tilde{Y} = Y B^{-1} - \tilde{C}^{-1} E Y B^{-1}$.

b) Furthermore, suppose that

$$\text{range}(EY) \subseteq \text{range}\, Y = N(A) \tag{8.2}$$

and define a matrix F *such that* $EYB^{-1} = YF$. *Then* $\tilde{C}^{-1}\tilde{Y} = YB^{-1}(I - F) + \tilde{C}^{-1} Y F^2 + \tilde{C}^{-1}\Delta F$.

Proof. First assume that the matrix C is nonsingular.

Observe that $\tilde{C}^{-1} = (I - \tilde{C}^{-1}E)C^{-1}$. Recall that $AY = 0$, and so $CY = (A + \tilde{Y}V^H)Y = \tilde{Y}(V^HY) = \tilde{Y}B$, $C^{-1}\tilde{Y} = YB^{-1}$. Therefore,

$$\tilde{C}^{-1}\tilde{Y} = (I - \tilde{C}^{-1}E)C^{-1}\tilde{Y} = YB^{-1} - \tilde{C}^{-1}EYB^{-1}.$$

This proves part a).

Substitute the equation $EYB^{-1} = YF$ into the equation of part a) and obtain that $\tilde{C}^{-1}\tilde{Y} = YB^{-1} - \tilde{C}^{-1}YF$.

Substitute

$$\tilde{C}^{-1}Y = \tilde{C}^{-1}\tilde{Y} - \tilde{C}^{-1}\Delta = YB^{-1} - \tilde{C}^{-1}YF - \tilde{C}^{-1}\Delta$$

on the right-hand side and obtain that

$$\tilde{C}^{-1}\tilde{Y} = YB^{-1}(I - F) + \tilde{C}^{-1}YF^2 + \tilde{C}^{-1}\Delta F.$$

This proves part b).

Relax the assumption that the matrix C is nonsingular by applying infinitesimal perturbations of the matrix A. □

The following lemma support the assumptions in part b).

Lemma 8.1. *Under (8.1), we have*

$$E = (\tilde{\lambda} - \lambda)I, \quad F = (\tilde{\lambda} - \lambda)B^{-1}, \tag{8.3}$$

and assumption (8.2) in part b) holds.

Theorem 8.2 implies the following estimates for the residual norm.

Corollary 8.1. *Let $||\cdot||$ denote any operator matrix norm. Then the norm $||C^{-1}\tilde{Y} - YB^{-1}||$ is in $O((||E||)$ under the assumptions of Theorem 8.2 a) whereas the norm $||\tilde{C}^{-1}\tilde{Y} - YB^{-1}(I - F)||$ is in $O((||\Delta|| + ||F||)||F||)$ under the assumptions of Theorem 8.2 b).*

Combining Theorem 8.1, Lemma 8.1, and Corollary 8.1 immediately implies quadratic convergence of Algorithms 6.1 and 7.1 to the eigenvalue/eigenspace pair assuming (8.1), the choice of $V = \tilde{Y}$, and a close initial approximation to the eigenvalue λ (but not necessarily to the associated eigenspace).

Remark 8.1. *In Theorem 8.2 b) we require that the matrix B be nonsingular. This property is expected to hold under random variation of the matrices $\tilde{Y}$ and V. The above estimate for the residual norm does not depends on the norm $||B^{-1}||_2$, which we estimate below only for the sake of completeness.*

Lemma 8.2. *Let $V = \tilde{Y}$ be a unitary matrix and let $||\Delta||_2 < 1$. Then the matrix B is nonsingular and $||B^{-1}||_2 \le \frac{1}{1-||\Delta||_2}$.*

Proof. Under the assumptions of the lemma, we have $B = I_k - \tilde{Y}^H\Delta$ and $B^{-1} = I_k + \sum_{i=1}^{\infty}(\tilde{Y}^H\Delta)^i$, and the lemma follows. □

9 Experimental iteration count for the IPI and Algorithm 6.1

In Tables 9.1 and 9.2 we show the numbers of iterations required for the convergence of the IPI and Algorithm 6.1. We display the average (mean) values and standard deviations in 200 tests with $n \times n$ matrices $A = \lambda I - M$ for $M = G^{-1}TG$, $n = 64$ and $n = 100$, G being either a random matrix or the Q-factor in the QR factorization of a random matrix, and T from one of the four following matrix classes.

1. $T = D_r$ is a real diagonal matrix with random entries in the closed line interval $[0, 10]$.
2. $T = D_c$ is a complex diagonal matrix whose entries have random absolute values in the line interval $[0, 10]$ and random arguments in the semi-open line interval $[0, 2\pi)$.
3. $T = D_r + \mathbf{e}_1\mathbf{v}^T + \mathbf{u}\mathbf{e}_n^T$ is an arrow-head matrix, D_r is a matrix of class 1, and the vectors $\mathbf{u}$ and $\mathbf{v}$ have random entries in the closed line interval $[0, 10]$.
4. $T = D_r + \mathbf{u}\mathbf{v}^T$, D_r and $\mathbf{v}$ are as in matrix class 3, and the vector $\mathbf{u}$ has random coordinates in the closed line interval $[0, 1]$.

The results of our extensive tests reported in Tables 9.1 and 9.2 show that the IPI and Algorithm 6.1 converge with about the same rate, even though Algorithm 6.1 deals with better conditioned matrices.

Table 9.1: Iteration count for IPI and Algorithm 6.1 with unitary matrix G

Matrix		Algorithm 6.1		IPI	
Classes	n	iter	std dev	iter	std dev
$T = D_r$	64	4.74	1.145	4.93	1.242
	100	4.71	1.277	4.88	1.299
$T = D_c$	64	5.67	1.415	5.61	1.396
	100	5.67	1.461	5.62	1.321
$T = D_r + \mathbf{e}_1\mathbf{v}^T + \mathbf{u}\mathbf{e}_n^T$	64	4.94	1.230	5.01	1.341
	100	4.75	1.176	4.75	1.260
$T = D_r + \mathbf{u}\mathbf{v}^T$	64	5.77	1.668	5.95	1.808
	100	5.54	1.445	5.67	1.553

10 Further extensions

We can extend Algorithm 7.1 to simultaneous approximation of more than one eigenvalue of a matrix polynomial $A(\lambda)$ (e.g., a cluster of h eigenvalues or a

Table 9.2: Iteration count for IPI and Algorithm 6.1 with random matrices G

Matrix		Algorithm 6.1		IPI	
Classes	n	iter	std dev	iter	std dev
$T = D_r$	64	5.36	2.532	5.36	2.520
	100	4.88	2.509	4.86	2.452
$T = D_c$	64	5.76	1.716	5.71	1.516
	100	5.59	1.401	5.64	1.497
$T = D_r + \mathbf{e}_1\mathbf{v}^T + \mathbf{u}\mathbf{e}_n^T$	64	5.09	1.621	5.03	1.605
	100	4.72	1.473	4.67	1.467
$T = D_r + \mathbf{u}\mathbf{v}^T$	64	5.550	1.907	5.550	1.872
	100	5.660	2.118	5.555	1.992

pair of complex conjugate eigenvalues of a real matrix polynomial M). In this case the variable λ is replaced by a set of h variables $\lambda_1, \ldots, \lambda_h$ and the matrix polynomial is modified respectively. For example, in the classical case the matrix polynomial $\lambda I - M$ is replaced by the product $\prod_{i=1}^{h}(\lambda_i I - M)$, which is a real matrix polynomial where $h = 2$ and λ_1 and λ_2 are the complex conjugate pair of eigenvalues of the matrix M.

We can extend Algorithm 7.1 by replacing the scalars λ, $\tilde{\lambda}$, and $\sigma(\tilde{\lambda})$ with the h-tuples $(\lambda_1, \ldots, \lambda_h)$, $(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_h)$, and $(||A(\tilde{\lambda}_i)||_q)_{i=1}^{h}$, respectively, replacing the vectors $\mathbf{v}$ and $\mathbf{y}$ with $n \times h$ matrices, and modifying its Stage 4 as follows:

> *4.* $(\tilde{\lambda}_i, \tilde{Y}_i) \longleftarrow$ *the eigenpairs of the* $k \times k$ *matrix polynomial* $B(\lambda) = Y^H A(\tilde{\lambda}) Y$*;* $Y \longleftarrow Y(\tilde{Y}_1, \ldots, \tilde{Y}_h)$*.*

Finally, one can extend A-preconditioning to any eigen-solver involving ill conditioned linear system of equations. This includes the IR-RI iteration, the Jacobi–Davidson algorithm, the shift-and-invert enhancements of the Arnoldi and Lanczos algorithms [3], and the deflation stage of the QR algorithm.

Appendix

A The affects of A-preprocessing on the eigensystem

Theorem 2.1 implies some rational characteristic equations for the eigenvalues of a matrix polynomial $A = A(\lambda)$. Suppose $g.m._A(\lambda) = r$ for a fixed value of λ, U and V are $n \times r$ matrix polynomials in λ, and $C = A + UV^H$. Then matrix equation (2.5) turns into the system of r^2 rational equations

$$F(\lambda) = I_r - V^H C^{-1} U = 0_r \tag{A.1}$$

satisfied by the eigenvalues λ with $g.m._A(\lambda) = r$. By pre- and post-multiplying matrix equation (A.1) by vectors $\mathbf{s}^H$ and $\mathbf{t}$ of dimension r, respectively, we obtain a single scalar equation in λ,

$$f(\lambda) = \mathbf{s}^H F(\lambda)\mathbf{t} = \mathbf{s}^H \mathbf{t} - \mathbf{s}^H V^H C^{-1} U \mathbf{t} = 0.$$

Let us estimate the impact of random A-preprocessing on the geometric multiplicity of the eigenvalues. We recall some basic definitions and a basic result for randomized algebraic computations.

Random sampling of elements from a finite set Σ is their selection from the set Σ at random, independently of each other, and under the uniform probability distribution on Σ. A matrix is *random* if its entries are randomly sampled (from a fixed finite set Σ).

An $k \times l$ *random unitary* matrix is the $k \times l$ Q-factor $Q(M)$ in the QR factorization of random $k \times l$ matrix M of the full rank.

Lemma A.1. *[29] (cf. also [30], [31]). For a finite set Σ of cardinality $|\Sigma|$, let a polynomial in m variables have total degree d, let it not vanish identically on the set Σ^m, and let the values of its variables be randomly sampled from the set Σ. Then the polynomial vanishes with a probability of at most $d/|\Sigma|$.*

Theorem A.1. *Let $A = A(\lambda)$, $U = U(\lambda)$, and $V = V(\lambda)$ denote three matrix polynomials of sizes $n \times n$, $n \times r$, and $n \times r$, respectively. Write $C = A + UV^H$. Fix a scalar λ and suppose that $r \le h = g.m._A(\lambda)$. Then*

a) $g.m._C(\lambda) \ge h - r$ and

b) $g.m._C(\lambda) = h - r$ with a probability of at least $1 - 2r/|\Sigma|$ if the $(m+n)r$ entries of the matrices U and V have been randomly sampled from a set Σ of cardinality $|\Sigma|$.

Proof. Part a) is immediate. Now suppose λ is fixed, $\rho = \operatorname{rank} A$, $q = \rho + r < n$, and A_q is a $q \times q$ submatrix of the matrix A such that $\operatorname{rank} A_q = \operatorname{rank} A = \rho$. Clearly, we can readily choose the matrices U and V such that the respective $q \times q$ submatrix C_q of the matrix $C = A + UV^H$ is nonsingular. Part b) follows from Lemma A.1 because $\det C_q$ is a nonzero polynomial of a degree of at most $2r$ in the entries of the matrices U and V. □

It follows that randomized A-preprocessing of a rank r is likely to decrease the geometric multiplicity of a multiple eigenvalue λ by $\min\{r,\ g.m._A(\lambda)\}$, and we should expect similar impact on the clusters of the eigenvalues. For Hermitian matrices the eigenvalues are also the singular values, and so random APPs are likely to decompress a compressed singular spectrum.

It is also likely, however, that the approximation of an eigenvalue λ of multiplicity $h > 1$ for a nonderogatory matrix A can be simplified if we apply a random APP UV^H of rank $r = h - 1$ to obtain the matrix $C = A + UV^H$. Indeed, in virtue of Theorem A.1, we can expect that $g.m._C(\lambda) = 1$.

B Application to polynomial root-finding

Matrix methods are effective and increasingly popular for the classical task of polynomial root-finding (see [23], [32]–[38], and the bibliography therein). The paper [23] exploits the structure of the input companion or generalized companion matrix to yield linear time per iteration versus quadratic time in the preceeding papers [33]–[35]. The root-finder relies on the IPI and, according to the test results in [23], is already slightly superior to the Durand–Kerner's (Weierstrass') celebrated root-finder. Application of A-preconditioning and aggregation should further enhance the power of this approach.

References

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

[2] G. W. Stewart, *Matrix Algorithms, Vol I: Basic Decompositions*, SIAM, Philadelphia, 1998.

[3] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, 1998 (first edition), 2001 (second edition).

[4] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[5] L. N. Trefethen, D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[6] V. Y. Pan, Computations in the Null Spaces with Additive Preconditioning, Technical Report TR 2007, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2007.

[7] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, Technical Report TR 2007, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2007.

[8] 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.

[9] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Co., Boston, 1996 (first edition) and SIAM Publications, Philadelphia, 2003 (second edition).

[10] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems,* Cambridge University Press, Cambridge, England, 2003.

[11] V. Y. Pan, The Schur Aggregation and Extended Iterative Refinement, Technical Report TR 2007, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2007.

[12] W. L. Miranker, V. Y. Pan, Methods of Aggregations, *Linear Algebra and Its Applications*, **29**, 231–257, 1980.

[13] Pan, V.Y., How Can We Speed up Matrix Multiplication? *SIAM Rev.,* **26**, **3**, 393–415, 1984.

[14] D. Coppersmith, S. Winograd, Matrix Multiplicaton via Arithmetic Progressions, *J. of Symbolic Computation*, **9**, **3**, 251–280, 1990.

[15] J. Laderman, V. Y. Pan, H. X. Sha, On Practical Algorithms for Accelerated Matrix Multiplication, *Linear Algebra and Its Applications*, **162–164**, 557–588, 1992.

[16] I. Kaporin, A Practical Algorithm for Faster Matrix Multiplication, *Numerical Linear Algebra with Applications*, **6, 8**, 687-700, 1999.

[17] I. Kaporin, The Aggregation and Cancellation Techniques As a Practical Tool for Faster Matrix Multiplication, *Theoretical Computer Science*, **315**, **2–3**, 469–510, 2004.

[18] M. T. Chu, R. E. Funderlic, G. H. Golub, A Rank-One Reduction Formula and Its Applications to Matrix Factorizations, *SIAM Review*, **37**, **4**, 512–530, 1995.

[19] L. Hubert, J. Meulman, W. Heiser, Two Purposes for Matrix Factorization: A Historical Appraisal, *SIAM Review*, **42**, **1**, 68–82, 2000.

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

[21] D. Bini, V. Y. Pan, Computing Matrix Eigenvalues and Polynomial Zeros Where the Output Is Real, *SIAM J. on Computing*, **27, 4**, 1099–1115, 1998. (Proceedings version in *Proc. 2nd Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'91)*, 384–393, ACM Press, New York, and SIAM Publications, Philadelphia, 1991.)

[22] E. R. Jessup, A Case Against a Divide and Conquer Approach to the Nonsymmetric Eigenvalue Problem, *Appl. Numer. Math.,* **12**, 403–420, 1993.

[23] 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, January 2004. (Also Technical Reports TR 2002 003 and 2002 020, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2002.)

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

[25] 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.

[26] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, New Jersey, 1980, and SIAM, Philadelphia, 1998.

[27] I. C. F. Ipsen, Computing an Eigenvector with Inverse Iteration, *SIAM Review*, **39**, 354–391, 1998.

[28] T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

[29] R. A. Demillo, R. J. Lipton, A Probabilistic Remark on Algebraic Program Testing, *Information Processing Letters*, **7**, **4**, 193–195, 1978.

[30] R. E. Zippel, Probabilistic Algorithms for Sparse Polynomials, *Proceedings of EUROSAM'79, Lecture Notes in Computer Science*, **72**, 216–226, Springer, Berlin, 1979.

[31] J. T. Schwartz, Fast Probabilistic Algorithms for Verification of Polynomial Identities, *Journal of ACM*, **27**, **4**, 701–717, 1980.

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

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

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

[35] 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.)

[36] 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.)

[37] 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.

[38] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Wang, X. Yan, Root-finding with Eigen-solving, pages 219–245 in *Symbolic-Numerical Computation*, (Dongming Wang and Lihong Zhi editors), Birkhäuser, Basel/Boston, 2007.