TR-2008006: 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

We incorporate our recent preconditioning techniques into the classical inverse power (Rayleigh quotient) iteration for computing matrix eigenvectors. Every loop of this iteration essentially amounts to solving an ill conditioned linear system of equations. Due to our modification we solve a well conditioned linear system instead. We prove that this modification preserves local quadratic convergence, show experimentally that fast global convergence is preserved as well, and yield similar results for higher order inverse iteration, covering the cases of multiple and clustered eigenvalues.

Key words: Additive preconditioning, Eigenspaces, Inverse Iteration

[*]Supported by PSC CUNY Awards 68291–0037 and 69330–0038. Some results of this paper have been presented at the International Conferences on the Matrix Methods and Operator Equations in Moscow, Russia, in June of 2005 and July 2007, on the Foundations of Computational Mathematics (FoCM'2005) in Santander, Spain, in July 2005, and on Industrial and Applied Mathematics, in Zürich, Switzerland, in July 2007, as well as at the SIAM Annual Meeting, in Boston, in July 2006, and at the Third International Workshop on Symbolic-Numeric Computation (SNC'07) in London, Ontario, Canada, in July 2007.

1 Introduction

We begin with preconditioning general linear systems of equations and then focus on the ill conditioned linear systems of equations that arise in eigen-solving iterations.

1.1 Preconditioning linear systems of equations

Preconditioning is a classical subject of numerical solution of linear systems of equations $A\mathbf{x} = \mathbf{b}$. One modifies the input matrix A to improve its conditioning. Better conditioned linear systems can be solved more accurately and faster (cf. [1]–[3] and the bibliography therein). Traditional preconditioning is the transition to better conditioned linear systems $MAN\mathbf{x} = M\mathbf{b}$ such that $\mathbf{y} = N\mathbf{x}$.

The critical problem for preconditioning is the choice of the multipliers M and N above (one of them can be the identity matrix) that would decrease the large condition number $\operatorname{cond} A$ to a much smaller value $\operatorname{cond}(MAN)$ or would compress the spectrum of the singular values of the matrix A into a small number of clusters. Computing such multipliers involves approximate factorization or inversion of the matrix A, which is generally as expensive as the solution of a linear system, can be unstable numerically [4, page 535], and can destroy the sparseness and structure of the matrix A.

To counter this problem we apply randomized *addititive preprocessing* of an 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; I_r and I denote the $r \times r$ identity matrix; "A-" and "APP" abbreviate "additive" and "additive preprocessor", respectively.

For an $n \times n$ matrix A and a positive integer r, define two *generators* U and V of the size $n \times r$, the APP UV^H, and the A-modification $C = A + UV^H$. According to the analysis and experiments in [5]–[7], A-preprocessing $A \rightarrow C$ for a random and properly scaled APPs UV^H of a rank r (such that the ratio $\frac{||UV^H||_2}{||A||_2}$ is neither large nor small) is expected to decrease the condition number $\operatorname{cond} A = \frac{\sigma_1(A)}{\sigma_n(A)}$ to the order $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$. Here and hereafter $\sigma_j(M)$ denotes the jth largest singular value of a matrix M, so that $\sigma_1(M) = ||M||_2$. If $\sigma_{n-r}(A) \gg \sigma_n(A)$, then our randomized A-preprocessing is expected to be *A-preconditioning*, that is, to decrease the condition number substantially.

Furthermore, we achieve effective preconditioning even with *weak randomization*, restricted to fewer random parameters and fixed patterns of structure and sparseness (see [6, Sections 4 and 6], [7, Sections 4.5 and 5]). With these patterns we can support popular iterative algorithms (such as the Conjugate Gradient algorithm) [1]–[3], [4, Section 10.2], [8]–[11], which essentually amount to recursive multiplication of the input matrix and its transpose by vectors and which rapidly converge provided the input matrix is well conditioned. Such algorithms are indispensible in large scale computations, where the input matrices are too large to permit any other operations.

1.2 Extension to eigen-solving

We have originally introduced A-preconditioning to accelerate the inverse power iteration, which we applied to polynomial root-finding (see [12], [13] and Appendix B). For an $n \times n$ input matrix M every iteration step essentially amounts to the solution of a linear system of equations with the matrix $A(\tilde{\lambda}) = \tilde{\lambda} I_n - M$, whose conditioning rapidly deteriorates as the approximation $\tilde{\lambda}$ converges to an eigenvalue λ. Solving such linear systems is a hurdle, even though the scaled solutions rapidly converge to an eigenvector in spite of the rounding errors [14], [15].

In our modification we yield the same convergence rate, but solve well conditioned linear systems with the coefficient matrices $C(\tilde{\lambda}) = A(\tilde{\lambda}) + UV^H$. We propose to do this in two ways.

In Approach 1 we apply the Sherman–Morrison–Woodbury inversion formula

$$A^{-1} = (C - UV^H)^{-1} = C^{-1} + C^{-1}UG^{-1}V^HC^{-1}, \quad G = I_r - V^HC^{-1}U \quad (1.1)$$

[4, page 50]. In Approach 2 we approximate the eigenvectors associated with an eigenvalue λ of the matrix M by the solutions of linear systems $C\mathbf{y} = \mathbf{u}$ where $\mathbf{u} = U\mathbf{x}$ for some vectors $\mathbf{x}$.

In both approaches the computed approximations to the eigenvectors are distinct from each other and from the approximations obtained in the classical inverse iteration, but we still prove local quadratic convergence and experimentally observe rapid global convergence, that is rapid convergence right from the start, the same as in the classical iteration.

We specify both new approaches to cover the cases of simple, multiple, and clustered eigenvalues and in Section 9 point out some natural modifications and extensions. In particular one can similarly incorporate weakly randomized additive preconditioning into other effective eigen-solvers such as Jacobi–Davidson algorithm, the shift-and-invert enhancements of the Lanczos and Arnoldi algorithms, and the deflation stage of the QR algorithm.

1.3 Related works

Small-rank modification is a known tool for decreasing the rank of a matrix [16], [17], 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 [4, Section 8.5.4], [18], [19], [20, Section 3.2]. (We refer the reader to [21] on some serious difficulties with the extension of the approach to the non-Hermitian eigenproblem.) We, however, know of no works on weakly randomized additive preconditioning of the input matrix, which is the main feature of our approach to eigen-solving.

1.4 Organization of the paper

We organize our presentation as follows. In the next section we state some definitions and recall some basic results from [22], [23]. In Section 3 we briefly review the inverse iteration for the eigenproblem and sketch our modifications. In Sections 4–6 we describe our rank-one, fixed-rank, and variable-rank modifications of this iteration. This includes its multilinear variants. In Section 7 we prove local quadratic convergence. The results of our numerical experiments in Section 8 show rapid global convergence. In Section 9 we list some natural extensions of our work. In Appendix A we estimate the impact of A-preconditioning on the eigensystem. In Appendix B we comment on applications of the inverse iteration to polynomial root-finding, in which case one can always ensure quadratic convergence right from the start. In Appendix C we briefly examine a modification of our approach and point out its potential problems.

The paper is due to the first author, except for the tests in Section 8, performed by the second author and Dr. Xinmao Wang in the University of Sciences and Technology of China (hereafter USTC) at Hefei, China. Some of our results were included into the proceedings paper [24].

2 Definitions and preliminaries

We use the customary definitions for matrix computations in [4], [20], [25]–[28].

2.1 Some basic results on null spaces

Hereafter $LN(A)$ and $N(A) = RN(A)$ denote the left and (right) null spaces of a matrix A, respectively; $\operatorname{nul} A = n - \operatorname{rank} A$ is the nullity of an $n \times n$ matrix A; $\operatorname{range}(M)$ is the range of a matrix M, that is its column span.

Our Approach 2 handles the eigenspaces of a matrix M associated with its eigenvalues λ as the null spaces of the matrices $\lambda I - M$ and is supported with the following theorem.

Theorem 2.1. *Suppose for an $n \times n$ matrix A of a rank ρ and a pair of $n \times r$ matrices U and V, the matrix $C = A + UV^H$ is nonsingular. Then*

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

$$N(A) \subseteq \operatorname{range}(C^{-1}U). \tag{2.2}$$

Furthermore if

$$r = \operatorname{rank} U = n - \rho = \operatorname{nul} A, \tag{2.3}$$

then we have

$$N(A) = \operatorname{range}(C^{-1}U), \tag{2.4}$$

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

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

$$\mathbf{y} = C^{-1}U(V^H \mathbf{y}). \tag{2.6}$$

Furthermore, $N(AC^{-1}U) = N(U(I_r - V^H V^{-1} U))$.

Proof. See [22, Theorem 3.1] or [23, Theorem 3.1] for $m = n$. □

Corollary 2.1. *Under the assumptions of Theorem 2.1 let equations (2.1) and (2.2) hold. Then* $N(A) = \mathrm{range}(C^{-1}UX)$ *if* X *is a matrix bases for the null space* $N(AC^{-1}U)$.

2.2 Matrix polynomials and the algebraic eigenproblem

For k matrices $A_0, \ldots, A_k$ of the same size we define matrix polynomials $A(\lambda) = \sum_{i=0}^{k} A_i\lambda^i$ whose norm $||A(\lambda)||$ is the sum of the norms $||A_0||, \ldots, ||A_k||$ or their another fixed positive function.

The eigenvalues of a matrix polynomial $A(\lambda)$ of a positive degree are the roots of the characteristic polynomial $c_A(\lambda) = \det A(\lambda)$. The eigenvalues of a scalar matrix M are the eigenvalues of the linear matrix polynomial $A(\lambda) = \lambda I - M$. One can assume just this simplest classical case until Section 5.

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* $l.g.m._A(\mu) = \mathrm{lnul}\, A(\mu)$ and $r.g.m._A(\mu) = g.m._A(\mu) = \mathrm{rnul}\, A(\mu)$, respectively.

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

3 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 task must be solved also at the deflation stage of the QR algorithm. As we recall in Section 1.1, by applying properly scaled weakly random APPs of suitable ranks we are likely to improve the conditioning of such linear systems. We elaborate upon this approach for the inverse power (Rayleigh quotient) iteration, which is a classical tool for the refinement of a crude solution to the algebraic eigenproblem [4], [14], [15], [20], [29], and for its block versions, called the inverse orthogonal iteration [4, page 339] and the inverse Rayleigh–Ritz subspace iteration [20, Section 6.1], for which we use the abbreviations *IPI* and *IR–RI*.

Somewhat counter-intuitively, the IPI produces a quite accurate eigenvector via the solution of ill conditioned linear systems of equations. This is not completely painless, however. In [28] 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 $A - \lambda I$ is ill conditioned near its eigenvalues λ, one cannot apply effective iterative algorithms for solving linear systems involved into the IPI and IR–RI processes, and this is a critical problem for large scale computations. We counter such a deficiency by applying A-preconditioning.

To explain our modifications, we first briefly recall the IPI. Given a close approximation $\tilde{\lambda}$ to a simple eigenvalue λ of a matrix polynomial $A(\lambda) = \sum_{i=0}^{m} A_i\lambda^i$ and a generally crude normalized approximation $\tilde{\mathbf{y}}$ to an associated eigenvector $\mathbf{y}$, the IPI recursively alternates the updatings of the scalar $\tilde{\lambda}$ and the vector $\tilde{\mathbf{y}}$ according to the maps $\{\tilde{\lambda} \leftarrow$ a root of the equation $\mathbf{y}^H A(\tilde{\lambda})\mathbf{y} = 0\}$ and $\{\tilde{\mathbf{y}} \leftarrow \frac{A^{-1}(\tilde{\lambda})\tilde{\mathbf{y}}}{||A^{-1}(\tilde{\lambda})\tilde{\mathbf{y}}||_2\}}$. The root above turns into the Rayleigh quotient $\tilde{\mathbf{y}}^H M\tilde{\mathbf{y}}$ 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 $||A(\lambda)\mathbf{y}||_2$.

If $\tilde{\lambda}$ approximates an eigenvalue, then the matrix $A(\tilde{\lambda})$ is ill conditioned, but we reduce the updating of the vector $\tilde{\mathbf{y}}$ to the solution of a linear system 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 properly scaled random vectors $\mathbf{u}$ and $\mathbf{v}$ (cf. [6, Examples 4.1–4.6], [7, Sections 4.5 and 5]). In the case of a simple isolated eigenvalue λ, the matrix polynomial $A(\lambda)$ has no small positive singular values. Then according to our study in [6], [7] we can expect that the matrix polynomial $C(\tilde{\lambda})$ is well conditioned, and so we stabilize the IPI numerically. To update the approximate eigenvectors $\tilde{\mathbf{y}}$, we apply the equations (1.1) or $\tilde{\mathbf{y}} = C^{-1}(\tilde{\lambda})\mathbf{u}$. (The latter vector is close to a vector $\mathbf{y} \in N(A(\lambda))$ wherever $\tilde{\lambda} \approx \lambda$. This follows from the equation $C(\lambda)\mathbf{y} = b\mathbf{u}$ for $\mathbf{y} \in N(A(\lambda))$ and $b = \mathbf{v}^H\mathbf{y}$.) We elaborate upon the respective algorithm in the next section and upon its extension to multiple and clustered eigenvalues in Sections 5 and 6.

4 Inverse iteration with APPs of rank one

Specifying our algorithms, we write $||\cdot||_q$ for $q = 2$ or $q = F$ to denote the 2-norm or the Frobenius norm of a matrix, respectively. We call a matrix M normalized if $||M||_2 = 1$. Actually in our algorithms we only need weak normalization, such that $||M||_2$ is neither large nor small. We employ error-free scaling by $\sigma(\tilde{\lambda})$, the powers of two. Equations (4.2) and (4.3) below define our Approaches 1 and 2, respectively, which rely on equations (1.1) and (2.3), (2.4), respectively.

$$C_1(\lambda) = \frac{1}{\sigma(\lambda)}A(\lambda) + UV^H, \quad C_2(\lambda) = \frac{1}{\sigma(\lambda)}A(\lambda) + YV^H, \tag{4.1}$$

$$f_1(A(\lambda), Y) = (C_1^{-1}(\lambda) + C_1(\lambda)UG^{-1}V^H C_1^{-1}(\lambda))Y, \tag{4.2}$$

$$f_2(A(\lambda), Y) = C_2^{-1}(\lambda)Y. \tag{4.3}$$

Algorithm 4.1. Inverse iteration with APPs of rank one *(cf. Remarks 4.1–4.4 and Section 9).*

INPUT: *a matrix M, a crude approximation $\tilde{\lambda}$ to its simple eigenvalue λ, a small positive tolerance value τ, a small positive integer ν (e.g., $\nu = 1$),* *the assignment $q = 2$ or $q = F$, and a Subroutine LIN·SOLVE for solving a nonsingular and well conditioned linear system of equations (e.g., based on PLU factorization or the Conjugate Gradient method).*

Output: *either FAILURE or an approximate eigenpair $(\lambda_{final}, \mathbf{y}_{final})$ of the matrix M such that $||A(\lambda_{final})\mathbf{y}_{final}||_2 \leq \tau ||A(\lambda)||_q$ where $A(\lambda) = \lambda I - M$.*

Initialization: *Fix an integer $g = 1$ or $g = 2$. Write $A(\lambda) = \lambda I - M$. Set $COUNTER \longleftarrow 0$, $\phi \leftarrow 0$, and $C_g(\tilde{\lambda}) \longleftarrow A(\tilde{\lambda})$. Generate random vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{y}$.*

Computations:

1. *If $COUNTER > \nu$, output FAILURE and stop. Otherwise apply Subroutine LIN·SOLVE to compute the vector $\mathbf{z} = C_g^{-1}(\tilde{\lambda})\mathbf{y}$ for $\phi = 0$ or $\mathbf{z} = f_g(A(\lambda), \mathbf{y})$ for $\phi = 1$ where $f_g(A(\lambda), \mathbf{y})$ is defined by equations (4.1)–(4.3) for $U = \mathbf{u}$, $V = \mathbf{v}$, and $Y = \mathbf{y}$.*
2. *If this application fails (that is, if the matrix $C_g(\tilde{\lambda})$ is singular or ill conditioned), then set $\phi \leftarrow 1$, compute a crude approximation $\sigma(\tilde{\lambda})$ by a power of two to the norm $||A(\tilde{\lambda})||_q$, generate random vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{y}$, set $COUNTER \longleftarrow COUNTER + 1$, and go to Stage 1.*
3. *Set $COUNTER \longleftarrow 0$. Compute the vector $\mathbf{x} = \mathbf{z}/||\mathbf{z}||_2$.*
4. *Compute the Rayleigh quotient $\gamma = \mathbf{x}^H M \mathbf{x}$.*
5. *If $||A(\gamma)\mathbf{x}||_2 \leq \tau ||A(\gamma)||_F$ (that is, if the residual norm is small enough), output $\lambda_{final} = \gamma$, $\mathbf{y}_{final} = \mathbf{x}$ and stop. Otherwise set $\tilde{\lambda} \longleftarrow \gamma$ and $\mathbf{y} \longleftarrow \mathbf{x}$ and go to Stage 1.*

Equations (1.1), (2.3), (2.4), and (4.1)–(4.3) link the algorithm to the classical IPI and imply its correctness. In fact Algorithm 4.1 turns into the customary IPI provided the Subroutine LIN·SOLVE never fails at Stage 1.

Remark 4.1. *At Stage 2 of the algorithm we can yield effective preconditioning even with the scalar $\sigma(\tilde{\lambda})$ from the previous iteration unchanged unless the norm $||A(\tilde{\lambda})||_q$ changes dramatically.*

Remark 4.2. *In the case where $g = 2$ the vectors $\mathbf{v}$ are not used in the algorithm, and we do not need to generate them.*

Remark 4.3. *By applying Algorithms 4.1 to the matrix M^H, we approximate its right eigenvectors, which are the left eigenvectors of the matrix M associated with the same eigenvalues. We can modify our algorithm and simultaneously approximate the pairs $(\mathbf{w}, \mathbf{x})$ of the left and right eigenvectors, respectively, associated with the eigenvalue λ. The computations would rely on the factorization of the same matrix $C_g(\tilde{\lambda})$. In this case at Stage 3 of Algorithm 4.1 one can compute the generalized Rayleigh quotients $\mathbf{w}^T A(\lambda)\mathbf{x}$ (cf. [20, Section 4.4]).*

Remark 4.4. *One can perform the iteration loop of Algorithm 4.1 concurrently for the two vectors $f_g(A(\lambda), \mathbf{y})$ for $g = 1$ and $g = 2$ and arrive at Stage 5 with two pairs $(\gamma, \mathbf{x})$ of the value γ and the vector $\mathbf{x}$, defined by the two values $g = 1$ and $g = 2$. Then one should continue the computations with the pair that supports the minimum value of the residual norm $||A(\gamma)\mathbf{x}||_2$ (cf. Section 9).*

All these remarks can be readily extended to the algorithms in the next two sections. In Section 9 we discuss some natural modifications of Algorithms 4.1 and of our next algorithms.

Algorithm 4.1 outputs FAILURE if the Subroutine LIN·SOLVE fails for the coefficient matrices $A(\tilde{\lambda})$ and ν computed instances of $C_g(\tilde{\lambda})$ where $\tilde{\lambda}$ is our current approximation to an eigenvalue λ. According to the study in [6], [7], this is unlikely to occur for a pair of random vectors $\mathbf{v}$ and $\mathbf{u}$ or $\mathbf{v}$ and $\mathbf{y}$ and a simple isolated eigenvalue λ.

5 Inverse iteration with APPs of a fixed rank

The following algorithm extends Algorithm 4.1 to approximating simultaneously a fixed number h of the eigenvalues and the associated eigenspace. In particular this handles the cases where $h = 2$ and we seek a pair of complex conjugate eigenvalues of a real matrix (cf. [20, page 97], [30]) as well as where we seek a cluster of h simple eigenvalues or a single eigenvalue of multiplicity h and an $n \times h$ matrix basis for the associated invariant space of eigenvectors. Here as well as in Remark 6.5 we call a matrix X of full column rank a *matrix basis* for range(X). The algorithm can be viewed as an APP-based modification of the IR–RI. As in the IR-RI, we assume that the h selected eigenvalues do not lie near the other eigenvalues of the input matrix.

Algorithm 5.1. Inverse iteration with APPs of a fixed rank. *(Cf. Remarks 4.1–4.4, 5.1–5.3, and Section 9.)*

INPUT: *a positive integer h, a matrix M, a crude approximation $\tilde{\Lambda} = (\tilde{\lambda}_i)_{i=1}^{g}$ to the vector $\Lambda = (\lambda_i)_{i=1}^{h}$ of its h eigenvalues, a small positive tolerance value τ and a small integer ν (e.g., $\nu = 1$), the assignment $q = 2$ or $q = F$, and a Subroutine LIN·SOLVE for solving a nonsingular and well conditioned linear system of equations.*

OUTPUT: *either FAILURE or a pair $(\Lambda_{final}, Y_{final})$ where Λ_{final} approximates the vector Λ of the h fixed eigenvalues of the matrix M and* range(Y_{final}) *approximates the associated eigenspace so that*

$$||A(\Lambda_{final})Y_{final}||_2 \leq \tau||A(\lambda)||_q \quad \text{for} \quad A(\Lambda) = \prod_{i=1}^{h}(\lambda_i I - M).$$

INITIALIZATION: *Fix an integer $g = 1$ or $g = 2$. Write $A(\Lambda) = \prod_{i=1}^{h}(\lambda_i I - M)$. Set $COUNTER \longleftarrow 0$ and $C_g(\tilde{\Lambda}) \longleftarrow A(\tilde{\Lambda})$. Generate weakly random $n \times h$ matrices U, V, and Y.*

COMPUTATIONS:

1. *If COUNTER $> \nu$, output FAILURE and stop. Otherwise apply Subroutine LIN·SOLVE to compute the matrix $Z = (C_g^{-1}(\tilde{\lambda}))Y$ for $\phi = 0$ or $Z = f_g(A(\lambda), Y)$ for $\phi = 1$ where $f_g(A(\lambda), Y)$ is defined by equations (4.1)–(4.3).*
2. *If this application fails (that is, if the matrix $C_g(\tilde{\Lambda})$ is singular or ill conditioned), then set $\phi \leftarrow 1$, compute a crude approximation $\sigma(\tilde{\Lambda})$ by a power of two to the norm $||A(\tilde{\Lambda})||_q$, generate weakly random $n \times h$ matrices U, V, and Y.*
3. *Set COUNTER $\longleftarrow 0$. Set $X \longleftarrow$ the Q-factor in the QR factorization of the matrix Z.*
4. *Compute the vector $\Gamma = (\gamma_i)_{i=1}^h$ of the eigenvalues of the $h \times h$ matrix $X^H M X$. Compute the matrix polynomial $A(\Gamma) = \prod_{i=1}^h (\gamma_i I - M)$.*
5. *If $||A(\Gamma)X||_2 \le \tau ||A(\Gamma)||_F$ (that is, if the residual norm is small enough), output $\Lambda_{final} = \Gamma$, $Y_{final} = X$ and stop. Otherwise set $\tilde{\Lambda} \longleftarrow \Gamma$ and $Y \longleftarrow X$ and go to Stage 1.*

Remark 5.1. *For the approximation of a single multiple eigenvalue, we can apply Algorithm 5.1 with the linear matrix polynomial $A(\lambda) = \lambda I - M$ instead of its h-th power $A(\lambda) = (\lambda I - M)^h$. Similarly we can simplify the algorithm if we seek the average of the h eigenvalues of a cluster separated from the other eigenvalues. In both cases in the description of the algorithm we would also replace the vectors Λ, $\tilde{\Lambda}$, and Γ with the scalars λ, $\tilde{\lambda}$, and $\gamma = (1/h)\,\mathrm{trace}(X^H A(\tilde{\Lambda})X)$, respectively (cf. our Algorithm 6.1).*

Remark 5.2. *For a real matrix M the vector Λ should consist of pairs of nonreal complex conjugate eigenvalues (say, $(\lambda_{2j-1}, \lambda_{2j})$ for $j = 1, \ldots, k$) and real eigenvalues (say, λ_i for $i = 2k+1, \ldots, 2k+l$). Then our matrix polynomial $A(\Lambda) = \prod_{j=1}^k ((\lambda_{2j-1}I - M)(\lambda_{2j}I - M)) \prod_{i=2k+1}^{2k+l} (\lambda_i I - M)$ would have real coefficients, and we could avoid involving nonreal values into our computations. In particular for $k = 1$ and $l = 0$, we would approximate a pair of complex conjugate eigenvalues λ_1 and λ_2 by working with the matrix polynomial $A(\Lambda) = M^2 - \alpha M + \beta I$ where $\alpha = \lambda_1 + \lambda_2$ and $\beta = \lambda_1 \lambda_2$.*

Remark 5.3. *The explicit formation of a matrix polynomial $A(\Lambda) = \prod_{i=1}^h (\lambda_i I - M)$ takes $O(hn^3)$ flops for general matrix M. This is dominated at the other stages of the computation if h is small and if we seek all n eigenvalues. Furthermore, in some cases such explicit formation can be avoided. For a sparse or structured matrix M, an APP UV^H (resp. YV^H) of a smaller rank r, and the well conditioned matrix $C_g(\Lambda)$ in (4.1), we can readily compute the matrix $C_1^{-1}(\Lambda)U$ (resp. $C_2^{-1}(\Lambda)Y$) by applying iterative algorithms. For $h = 2$ and a dense Hessenberg matrix M, we can compute the QR factorization of the matrix $A(\Lambda)$ by applying Francis implicit double shifts and then extend it to the QR factorization of the matrix $C_g(\Lambda)$ by using $O(n^2)$ flops overall [4, Sections 7.5.5 and 12.5.2].*

6 Inverse iteration with APPs of adjusted ranks

Suppose we wish to extend Algorithm 4.1 to approximating an isolated multiple eigenvalue λ together with the associated eigenspace, but do not know the multiplicity of the eigenvalue. Then we can apply linear or binary search for the multiplicity. E.g. for $g = 1$ (resp. $g = 2$) we can recursively generate the pairs of scaled random vectors $\mathbf{u}$ and $\mathbf{v}$ (resp. $\mathbf{y}$ and $\mathbf{v}$) and add their outer products $\mathbf{u}\mathbf{v}^H$ (resp. $\mathbf{y}\mathbf{v}^H$) to the matrix $C_g(\tilde{\lambda})$ until it becomes nonsingular and well conditioned. We can apply similar recipes to approximating the average λ of all eigenvalues in an isolated cluster whose cardinality is unknown. Then again the resulting algorithm below can be viewed as a modification of the IR–RI that employs APPs.

Algorithm 6.1. Inverse iteration with APPs of adjusted ranks *(cf. Remarks 4.1–4.4, 6.1–6.5, and Section 9).*

INPUT: *an $n \times n$ matrix M, an approximation $\tilde{\lambda}$ to its eigenvalue λ having unknown multiplicity, a small positive tolerance value τ and a small integer ν, e.g., $\nu = 1$, the assignment $q = 2$ or $q = F$, and a Subroutine LIN·SOLVE for solving a nonsingular and well conditioned linear system of equations.*

OUTPUT: *either FAILURE or an approximation $(\lambda_{final}, Y_{final})$ to an eigenpair (λ, Y) of the matrix M (where λ is an eigenvalue and Y is a matrix basis for the associated eigenspace) such that $||(\lambda_{final}I - M)Y_{final}||_2 \leq \tau||\lambda_{final}I - M||_q$.*

INITIALIZATION: *Fix an integer $g = 1$ or $g = 2$. Write $A(\lambda) = \lambda I - M$. Set $C_g(\tilde{\lambda}) \longleftarrow A(\tilde{\lambda})$, $COUNTER \longleftarrow 0$, and $\phi \longleftarrow 0$. Generate a triple of random vectors $V = \mathbf{v}$, $U = \mathbf{u}$, and $Y = \mathbf{y}$.*

COMPUTATIONS:

1. *If $COUNTER > \nu$, output FAILURE and stop. Otherwise apply the Subroutine LIN·SOLVE to compute the matrix $Z = (C_g^{-1}(\tilde{\lambda}))Y$ for $\phi = 0$ or $Z = f_g(A(\lambda), Y)$ for $\phi = 1$ where $f_g(A(\lambda), Y)$ is defined by equations (4.1)–(4.3).*
2. *If the latter application fails (that is, if the matrix $C_g(\tilde{\lambda})$ is singular or ill conditioned), then set $\phi \leftarrow 1$ and choose a triple of normalized random vectors $\tilde{\mathbf{u}}$, $\tilde{\mathbf{v}}$, and $\tilde{\mathbf{y}}$. Require that these vectors not lie in the ranges of the matrices U, V, and Y, respectively. If $\phi = 0$, set $\phi \longleftarrow 1$ and set either $(V, U) = (\mathbf{v}, \mathbf{u}) \longleftarrow (\tilde{\mathbf{v}}, \tilde{\mathbf{u}})$ for $g = 1$ or $(V, Y) = (\mathbf{v}, \mathbf{y}) \longleftarrow (\tilde{\mathbf{v}}, \tilde{\mathbf{y}})$ for $g = 2$. Otherwise (that is if $\phi = 1$) append appropriate column vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{y}$ to the matrices U, V, and Y, respectively, keeping the matrices unitary. Namely compute unitary matrices $(V, \mathbf{v})$ and either $(U, \mathbf{u})$ for $g = 1$ or $(Y, \mathbf{y})$ for $g = 2$ where $\mathbf{v} = V\mathbf{c} + a\tilde{\mathbf{v}}$ and either $\mathbf{u} = U\mathbf{d} + b\tilde{\mathbf{u}}$ for $g = 1$ or*

$\mathbf{y} = Y\mathbf{d} + b\tilde{\mathbf{y}}$ for $g = 2$ and where $\mathbf{c}$ and $\mathbf{d}$ are vectors and a and b are nonzero scalars. Set $V \longleftarrow (V, \mathbf{v})$ and either $U \longleftarrow (U, \mathbf{u})$ for $g = 1$ or $Y \longleftarrow (Y, \mathbf{y})$ for $g = 2$. In both cases compute a crude approximation $\sigma(\tilde{\lambda})$ by a power of two to the norm $||A(\tilde{\lambda})||_q$, set $COUNTER \longleftarrow COUNTER + 1$, and go to Stage 1.

3. *Set $COUNTER \longleftarrow 0$ and compute the Q-factor X in the QR factorization of the matrix Z where the R-factor has positive diagonal entries.*
4. *Compute the Rayleigh quotient $\gamma = (1/h)\,\mathrm{trace}(X^H M X)$.*
5. *If $||A(\gamma)X||_2 \leq \tau||A(\gamma)||_F$ (that is, if the residual norm is small enough), output $\lambda_{final} = \gamma$, $Y_{final} = X$ and stop. Otherwise set $Y \longleftarrow X$ and $\tilde{\lambda} \longleftarrow \gamma$ and go to Stage 1.*

Remark 6.1. *Algorithm 6.1 can be applied to approximating the average value λ in a cluster of eigenvalues isolated from the other eigenvalues of the matrix. The algorithm needs no changes besides the change of the meaning of the value λ and its approximations $\tilde{\lambda}$ and γ. Indeed the average value λ is a multiple eigenvalue of a nearby matrix.*

Remark 6.2. *Unlike the classical inverse iteration, adjusting the rank of the APPs in Algorithm 6.1 and in its latter extension enables us to detect the multiplicity of a multiple eigenvalue and the number of eigenvalues in a cluster, respectively.*

Remark 6.3. *At Stage 2 one can compute and update the matrix $C_g(\tilde{\lambda})$, together with its QR factorization, at the cost of $O(n^2)$ flops per update (cf. [4, Section 12.5.1]).*

Remark 6.4. *Some alternative techniques of linear and bilinear search for the proper size of an APP can be found in [22]. E. g., one can begin with an APP UV^H for $g = 1$ (resp. YV^H for $g = 2$) of a larger rank to yield a well conditioned matrix $C_g(\tilde{\lambda})$. Then one can generate APPs UV^H (resp. YV^H for $g = 2$) with the ranks recursively decreasing as long as the resulting matrix $C_g(\tilde{\lambda})$ remains well conditioned.*

Remark 6.5. *We can readily extend Algorithm 6.1 to approximate the individual eigenvalues in the cluster by combining the IR–RI with A-preconditioning. The changes versus Algorithm 6.1 would essentially amount to replacing the eigenvalue λ and its approximations $\tilde{\lambda}$ and γ with the vectors $\Lambda = (\lambda_i)_{i=1}^h$ of the eigenvalues and the vectors $\tilde{\Lambda} = (\tilde{\lambda}_i)_{i=1}^h$ and $\Gamma = (\gamma_i)_{i=1}^h$ of its approximations, replacing the matrix polynomial $A(\lambda) = \lambda I - M$ with the matrix polynomial $A(\Lambda) = \prod_{i=1}^h (\lambda_i I - M)$, and replacing Stage 4 of this algorithm with Stage 4 of Algorithm 5.1. With the IR-RI techniques (cf. [20, Section 4.4]) we can also compute some matrix bases Y_i for the eigenspaces associated with these eigenvalues λ_i. We should just compute some matrix bases W_i for the respective associated eigenspaces of the matrix $X^H M X$ and then output the matrices*

$Y_i = XW_i$ for $i = 1, \dots, n$. Similarly we can extend the algorithm to approximating the eigenvalues and the associated eigenspaces for a pair of complex conjugate clusters of the eigenvalues of a real matrix and more generally to approximating any fixed set of isolated clusters of matrix eigenvalues and the associated eigenspaces.

7 Perturbations and errors in the modified inverse iteration

Although the iteration steps of Algorithms 4.1, 5.1, and 6.1 are better conditioned and thus computationally simpler than the steps of the IPI and IR–RI, they still rapidly converge to the eigenspaces of a matrix M or multilinear matrix polynomial $A(\lambda)$. Next we prove local quadratic convergence of these algorithms applied to the classical algebraic eigenproblem, where

$$A = A(\lambda) = \lambda I - M, \;\; \tilde{A} = A(\tilde{\lambda}) = \tilde{\lambda} I - M, \tag{7.1}$$

and the algorithms recursively refine approximations $\tilde{\lambda}$ to an eigenvalue λ and $\tilde{Y}$ to a matrix basis Y for the associated eigenspace. The same proof of quadratic convergence applies to the multilinear inverse iteration in the previous two sections (cf. Lemma 7.1).

We first express the errors in the Rayleigh quotients via the eigenvectors errors (without assuming equations (7.1)).

Theorem 7.1. *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 7.2. *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) $\tilde{C}^{-1}\tilde{Y} = YB^{-1} - \tilde{C}^{-1}EYB^{-1}$.

b) Furthermore, suppose that

$$\operatorname{range}(EY) \subseteq \operatorname{range} Y = N(A) \tag{7.2}$$

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

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 $CY = (A + \tilde{Y}V^H)Y = \tilde{Y}(V^H Y) = \tilde{Y}B$, $C^{-1}\tilde{Y} = YB^{-1}$. 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 validates assumption (7.2) in part b) for linear and multilinear inputs $A(\lambda)$.

Lemma 7.1. *Under (7.1) as well as for a multilinear matrix polynomial $A(\Lambda)$ of Sections 4 and 6, we have*

$$E = (\tilde{\lambda} - \lambda)I, \quad F = (\tilde{\lambda} - \lambda)B^{-1}, \tag{7.3}$$

and assumption (7.2) in part b) holds.

Theorem 7.2 implies the following estimates for the residual norm.

Corollary 7.1. *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 7.2 a) whereas the norm $||\tilde{C}^{-1}\tilde{Y} - YB^{-1}(I - F)||$ is in $O((||\Delta|| + ||F||)||F||)$ under the assumptions of Theorem 7.2 b).*

Combining Theorem 7.1, Lemma 7.1, and Corollary 7.1 immediately implies quadratic convergence of Algorithms 4.1, 5.1, and 6.1 to the eigenvalue/eigenspace pair assuming (7.1), the choice of $V = \tilde{Y}$, and a close initial approximation to the eigenvalue λ (but not necessarily to the associated eigenspace).

Remark 7.1. *In Theorem 7.2 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 7.2. *Let $V = \tilde{Y}$ be a unitary matrix and let $||\Delta||_2 < 1$. Then the matrix B is nonsingular and $||B^{-1}||_2 \leq \frac{1}{1-||\Delta||_2}$.*

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. □

8 Experimental iteration count for the inverse iteration and our algorithm

Tables 8.1 and 8.2 show the numbers of iterations required for the convergence of the IPI and Algorithm 4.1. We display the average (mean) values and the 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 (cf. [27, Section 28.3]).

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 $[0, 1]$.

We have also tested Algorithm 5.1 versus the classical inverse iteration for $n \times n$ real input matrices $M = G^{-1}TG$ having complex conjugate eigenvalue pairs (λ_1, λ_2). Here G was random orthogonal, T was diagonal with 2×2 blocks $T = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$, $a = r\cos(s)$, $b = r\sin(s)$, r was random in the range $[0, 10]$, s was random in the range $[0, 2p\sqrt{-1}]$. The initial value λ was chosen random in the square $\{z : -10 < \Re[z], \Im[z] < 10\}$.

1000 runs for $n = 100$ were performed for both classical inverse iteration and Algorithm 5.1 for $\nu = 1$, $\tau = 1e - 10$, and each $g = 1$ and $g = 2$. The classical IPI required on the average 5.803 iterations until convergence with the standard deviation 5.612, versus the average 5.753 (resp. 8.013) and the standard deviation 4.358 (resp. 3.563) for Algorithm 5.1 for $g = 1$ (resp. $g = 2$).

In yet another series of tests, $n \times n$ matrices M had clustered triples of eigenvalues λ_j, $j = 1, 2, 3$. They were generated as $M = G^{-1}TG$ for random orthogonal matrices G and diagonal matrices T with each random element in the range $[0, 10]$ repeated three times. The initial λ was random in the range $[0, 10]$. The classical inverse iteration was compared with our Algorithms 4.1, 5.1, and 6.1 for $n = 100$, $\nu = 1$, $\tau = 1e - 10$, and for both $g = 1$ and $g = 2$. Then again 1000 tests were performed for each algorithm. For $g = 1$ the iteration count showed the average 4.101 and the standard deviation 2.022 for Algorithm 4.1; 3.973 and 3.658 for Algorithm 5.1; 4.302 and 3.769 for Algorithm 6.1, and 4.195 and 1.435 for the classical inverse iteration. For $g = 2$ Algorithms 4.1, 5.1, and

6.1 diverged in about 70 percents of his tests. In the cases of convergence the iteration count was similar to the case $g = 1$.

Apart from the latter cases of divergence, in all test results the inverse iteration and our algorithms converged with about the same rate for the same inputs, even though our algorithms consistently involved better conditioned matrices.

Table 8.1: Iteration count for IPI and Algorithm 4.1 with unitary matrix G

Matrix Classes	n	Algorithm 4.1		IPI	
		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

Table 8.2: Iteration count for IPI and Algorithm 4.1 with random matrices G

Matrix Classes	n	Algorithm 4.1		IPI	
		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

9 Some extensions

In this paper we demonstrated the power of A-preconditioning for improving the inverse iteration, and this should motivate its further elaboration and analyzis, in particular for its application to clusters and nested clusters of the eigenvalues.

For simplicity we restricted the presentation to the classical eigenproblem and the inverse iteration, but our weakly randomized A-preconditioning can be readily extended to the generalized eigenproblem and to various other eigensolvers that involve ill conditioned linear systems of equations. This includes the

Jacobi–Davidson algorithm, the shift-and-invert enhancements of the Arnoldi and Lanczos algorithms [20], and the deflation stage of the QR algorithm.

There is a variety of natural modifications of both of our approaches.

- Instead of choosing random vectors $\mathbf{v}$ or matrices V in Algorithms 4.1, 5.1, and 6.1 one can simply set $\mathbf{v} = \mathbf{u}$ and $V = U$ for $g = 1$ or $\mathbf{v} = \mathbf{y}$ and $V = Y$ for $g = 2$, thus using fewer random parameters. Our preliminary tests show that this modification does not affect the convergence rate.

- In Approach 1 one can rely on the following simplification of the inversion formula (1.1), provided one seeks just the solution Y to a matrix equation $AY = B$ rather than the inverse A^{-1} (cf. [31], [32], [33]):

  Choose the matrices F and V of fixed appropriate sizes and successively compute the matrices

  $$UF = B, \quad C = A + UV^H, \quad G = I_r - V^H C^{-1} U, \quad \text{and} \quad Y = C^{-1} U G^{-1} F.$$

  To solve a linear system $A\mathbf{y} = \mathbf{b}$, we would apply this modification for vectors $B = \mathbf{b}$, $F = \mathbf{f}$, and $Y = \mathbf{y}$.

- In Approach 2 one can relax updating the vectors $\mathbf{y}$ and matrices Y at Stage 5 of our algorithms.

- One can incorporate a modification of additive preprocessing in [33, Section 12], called *preconditioning by expansion*, which a little simplifies the computations of and with the A-modification C at the expense of some controlled increase of its size.

Finally a critical problem of the initialization of the IPI/IR–RI can be attacked by concurrent application of these algorithms at a number of initial values $\tilde{\lambda}_i$. At least some of these concurrent processes can be expected to converge. Concurrent application of our present algorithms for $g = 1$ and $g = 2$ and possibly the classical inverse iteration could help us to improve global convergence further (cf. Remark 4.4). One can try to enhance the chances for more rapid convergence by extending these concurrent processes with further modifications of our algorithms, such as those listed above.

Appendix

A The impact 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

$$F(\lambda) = I_r - V^H C^{-1} U = O_{r,r} \qquad (A.1)$$

satisfied by the eigenvalues λ. 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 randomized 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. *[34] (cf. also [35], [36]). 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 $\frac{d}{|\Sigma|}$.*

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 $1 - \frac{2r}{|\Sigma|}$ 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. $\square$

It follows that randomized A-preprocessing of a rank r is likely to decrease the geometric multiplicity of a multiple eigenvalue λ by $\min\{r,\ g.m._A(\lambda) - 1\}$, 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 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 [12], [13], [37]–[43], and the bibliography therein). The papers [12] and [13] exploit the structure of the input companion or generalized companion matrix to yield linear time per iteration versus quadratic time in the preceeding papers [38]–[40]. The root-finder relies on the IPI and, according to the test results in [13], is already slightly superior to Durand–Kerner's (Weierstrass') celebrated root-finder. We can expect further accelleration if we incorporate the unsymmetric Lanczos algorithm with the shift-and-invert enhancement instead of the IPI (cf. [43]). Application of A-preconditioning and aggregation should further enhance the power of this approach with both IPI and Lanczos bases. Even more promising acceleration relies on repeated squaring of the Frobenius companion matrix of an input polynomial [44]. Every squaring step essentially amounts to performing a small number of FFTs at the cost of $O(n \log n)$ flops. The resulting superfast version of the inverse iteration converges to a nearest polynomial root with quadratic rate right from the start, and this can be repeated for the next root by applying implicit deflation.

C A-modification with approximate eigenvectors

Recall the variants of our Approach 2 with or without updating the vector $\mathbf{y}$ and matrix Y (cf. Section 9). Apply them in Algorithm 4.1 where $g = 2$ and the vector $\mathbf{y}$ is not random, but approximates an eigenvector associated with an eigenvalue λ. The matrix $A(\lambda)$ can stay ill conditioned in the transition $A(\lambda) \to A(\lambda) + \mathbf{y}\mathbf{v}^H$ wherever

a) the vector $\mathbf{y}$ lies in or near the range of the matrix $A(\lambda)$ or

b) λ is a multiple eigenvalue of the matrix polynomial $A(\lambda)$ or lies near another eigenvalue.

Actually property b) follows from property a) due to the following simple lemma and well known theorem.

Lemma C.1. *Let a matrix $A = \lambda I - M$ have eigenvalue zero, let $\mathbf{w}$ be the associated left eigenvector, and let $\mathbf{y} = A\mathbf{z} + \Delta$. Then $\mathbf{w}^H\mathbf{y} = \mathbf{w}^H\Delta$.*

Proof. $\mathbf{w}^H\mathbf{y} = \mathbf{v}^H(A\mathbf{z} + \Delta) = \mathbf{w}^H A\mathbf{z} + \mathbf{w}^H\Delta$. Substitute $\mathbf{w}^H A = \mathbf{0}^H$. □

Theorem C.1. *(Cf. [45].) Suppose a simple eigenvalue λ of a matrix M is associated with the pair of normalized left and right eigenvectors $\mathbf{w}$ and $\mathbf{y}$ and suppose that the condition $\delta = |\mathbf{w}^H\mathbf{y}|$ of this eigenvalue is exceeded by one. Then there is a matrix E such that $\frac{||E||_2}{||M||_2} \le \frac{\delta}{\sqrt{1-\delta^2}}$ and λ is a multiple eigenvalue of the matrix $M + E$.*

Unless properties a) and b) hold, the above modifications of Algorithm 4.1 preserve its power, but if these properties hold, then the modified algorithm can indeed readily diverge. Xinmao Wang at USTC in Hefei, China has devised a specific 10×10 matrix having properties a) and b) on which the algorithm outputs FAILURE in computations with the IEEE standard double precision for any choice of the integer ν.

Acknowledgements

Dr. Xinmao Wang tested our algorithms for bilinear and multilinear inputs (see Section 8), countered Algorithm 4.1 with a specific hard input (see Appendix C), and pointed out the advantages of employing equation (4.2) and choosing $g = 1$. Valuable comments by Dr. V. B. Khazanov helped us to improve our original draft of this paper substantially.

References

[1] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[2] M. Benzi, Preconditioning Techniques for Large Linear Systems: a Survey, *J. of Computational Physics*, **182**, 418–477, 2002.

[3] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, Cambridge, England, 2005.

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

[5] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, I. Taj-Eddin, Y. Tang, X. Yan, Additive Preconditioning and Aggregation in Matrix Computations, *Computers and Mathematics (with Applications)*, **55**, **8**, 1870–1886, 2008.

[6] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, Technical Report TR 2008004, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2008. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[7] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, *Proc. of the Third International Computer Science Symposium in Russia (CSR 2008)*, *Lecture Notes in Computer Science (LNCS)*, **5010**, 372–383, 2008.

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

[9] 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.

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

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

[12] V. Y. Pan, Univariate Polynomial Root-finding with Lower Precision and Higher Convergence Rate, Technical Report TR 2002003, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2002. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[13] 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 Report TR 2002020, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2002, available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352).

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

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

[16] 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.

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

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

[19] 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.)

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

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

[22] V. Y. Pan, Computations in the Null Spaces with Additive Preconditioning, Technical Report TR 2007009, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2007. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[23] V. Y. Pan, G. Qian, Solving Homogeneous Linear Systems with Weakly Randomized Additive Preprocessing, Technical Report TR 2008009, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2008. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[24] V. Y. Pan, X. Yan, Null Space and Eigenspace Computations with Additive Preprocessing, *Proceedings of the Third International Workshop on Symbolic–Numeric Computation (SNC 2007)*, 152–160, July 2007, London, Ontario, Canada (Jan Verschelde and Stephen Watt eds.), ACM Press, New York, 2007.

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

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

[27] N. J. Higham, *Accuracy and Stability in Numerical Analysis*, SIAM, Philadelphia, 2002 (second edition).

[28] 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.

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

[30] J. H. Wilkinson, Global Convergence of Tridiagonal QR Algorithm with Origin Shifts, *Linear Algebra and Its Applications*, **1**, 409–420, 1968.

[31] 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.

[32] V. Y. Pan, B. Murphy, R. E. Rosholt, M. Tabanjeh, The Schur Aggregation for Solving Linear Systems of Equations, *Proceedings of the Third International Workshop on Symbolic–Numeric Computation (SNC 2007)*, 142–151, July 2007, London, Ontario, Canada, (Jan Vercshelde and Stephen Watt eds.), ACM Press, New York, 2007.

[33] V. Y. Pan, D. Grady, B. Murphy, G. Qian, R. E. Rosholt, A. Ruslanov, Schur Aggregation for Linear Systems and Determinants, *Theoretical Computer Science*, *Special Issue on Symbolic–Numerical Algorithms* (D. A. Bini, V. Y. Pan, and J. Verschelde editors), accepted. Also

Technical Report TR 2008008, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2008. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

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

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

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

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

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

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

[40] 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.)

[41] 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.)

[42] 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.

[43] 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.

[44] V. Y. Pan, Amended DSeSC Power Method for Polynomial Root-finding, *Computers and Mathematics with Applications*, **49**, **9-10**, 1515-1524, 2005.

[45] J. H. Wilkinson, Note on Matrices with a Very Ill-Conditioned Eigenproblem, *Numerische Math.*, **19**, 176–178, 1972.