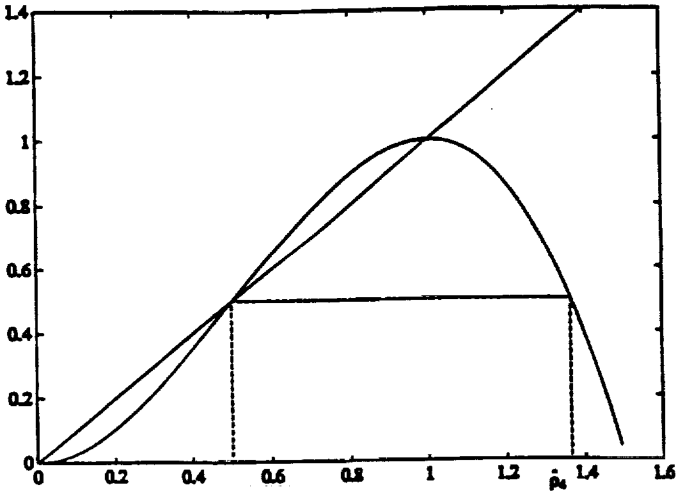
RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

IN-64
43049
p 40

An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications

Victor Pan **Robert Schreiber**



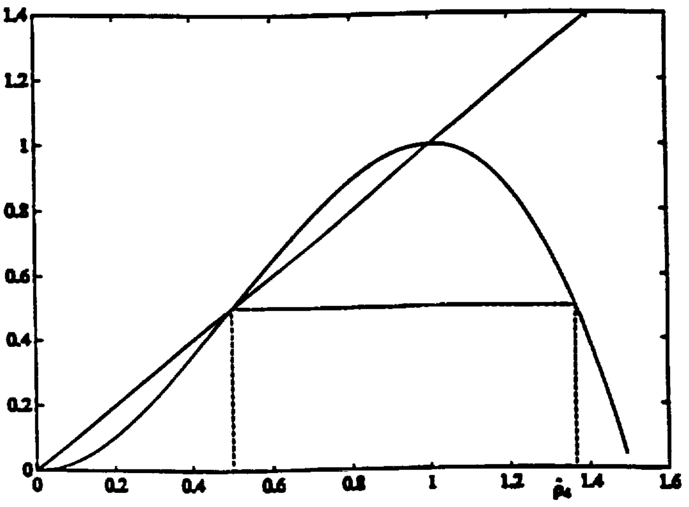
RIACS Technical Report 90.16 **March, 1990**

To appear: SIAM Journal on Scientific and Statistical Computing

(NASA-CR-186983) AN IMPROVED NEWTON ITERATION FOR THE GENERALIZED INVERSE OF A MATRIX, WITH APPLICATIONS (Research Inst. for Advanced Computer Science) 40 pCSCL 12A
N92-11723
Unclas
G3/64 0043049

An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications

Victor Pan **Robert Schreiber**



The Research Institute of Advanced Computer Science is operated by Universities Space Research Association, The American City Building, Suite 311, Columbia, MD 244, (301)730-2656

Work reported herein was supported by the NAS Systems Division of NASA and DARPA via Cooperative Agreement NCC 2-387 between NASA and the University Space Research Association (USRA). Work was performed at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, CA 94035.

An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications *

Victor Pan

Computer Science Department, SUNY
Albany, NY 12222
and
Mathematics Department, Lehman College
CUNY, Bronx, NY 10468
(Supported by NSF Grant CCR-8805782 and PSC-CUNY Award 668541)
and

Robert Schreiber

Research Institute for Advanced Computer Science
Moffett Field, CA 94035
(Supported by ONR Contract number N00014-86-K-0610 and
ARO Grant DAAL03-86-K-0112 to the Rensselaer Polytechnic Institute,
and by the NAS Systems Division via
Cooperative Agreement NCC2-387 between NASA and the
Universities Space Research Association.)

Abstract

Pan and Reif have shown that Newton iteration may be used to compute the inverse of an nxn, well-conditioned matrix in parallel time $O(\log^2 n)$ and that this computation is processor efficient. Since the algorithm essentially amounts to a sequence of matrix-matrix multiplications, it can be implemented with great efficiency on systolic arrays and parallel computers.

Newton's method is expensive in terms of the arithmetic operation count. In this paper we reduce the cost of Newton's method with several new acceleration procedures. We obtain a speedup by a factor of two for arbitrary input matrices; for symmetric positive definite matrices, the factor is four. We also show that the accelerated procedure is a form of Tchebychev acceleration, while Newton's method uses instead a Neumann series approximation.

---

* Submitted to the SIAM Journal on Scientific and Statistical Computing.

In addition, we develop Newton-like procedures for a number of important related problems. We show how to compute the nearest matrices of lower rank to a given matrix A, the generalized inverses of these nearby matrices, their ranks (as a function of their distances from A), and projections onto subspaces spanned by singular vectors; such computations are important in signal processing applications. Furthermore, we show that the numerical instability of Newton's method when applied to a singular matrix is absent from these improved methods. Finally, we show how to use these tools to devise new polylog time parallel algorithms for the singular value decomposition.

Key Words: Matrix computation, Moore-Penrose generalized inverse, singular value decomposition, Newton iteration.

Acknowledgements. We would like to thank the referees for many valuable suggestions, especially for helping to simplify and clarify the proofs; Roland Freund and Youcef Saad for suggesting the Tchebychev analysis; and Sally Goodall for expertly typing the manuscript.

1. Introduction

Pan and Reif [11], [12] have shown that Newton iteration may be used to compute the inverse of an nxn, well-conditioned matrix in parallel time proportional to $\log^2 n$ using a number of processors that is within a factor of log n of the optimum. Newton iteration is simple to describe and to analyze, and is strongly numerically stable for nonsingular input matrices; this is not true of earlier polylog time matrix inversion methods. Moreover, since it is rich in matrix-matrix multiplications, it can be implemented with great efficiency on systolic arrays and parallel computers. [1]

Unfortunately, the computation can be rather expensive. It has been shown that the number of matrix multiplications required can be as high as 4 log $\kappa(A)$, where $\kappa(A) \equiv ||A||_2 \, ||A^{-1}||_2$.

Here we show how to reduce this cost, in some cases quite dramatically. We accelerate and extend the usual Newton iteration in several ways. In particular, we

- substantially accelerate the convergence of the iteration;
- insure its stability even when A is singular;
- show how to compute the matrix $A(\varepsilon)$ obtained from A by setting to zero any of its singular values that are less than ε. Thus $A(\varepsilon)$ is a closest lower rank approximation to A (see Theorem 2.1 below);
- compute $A^+(\varepsilon)$, the Moore-Penrose generalized inverse (also called the pseudo-inverse) of $A(\varepsilon)$;
- compute the rank of $A(\varepsilon)$;

---

[1] (On the Connection Machine [7] matrix products may be computed at 5×10^9 operations per second, but matrix computations done in standard languages rarely can exceed 10^8 operations per second. Furthermore, computer manufacturers are currently being encouraged to provide the fastest matrix product software possible, since other matrix computations (QR and LU decomposition, in particular) may be computed with algorithms that are rich in matrix multiply [4].)

- compute the matrix $P(\varepsilon)$ that projects orthogonally onto the range of $A(\varepsilon)$.

Concerning acceleration, we obtain a twofold speedup by scaling the iterates at each step, and we prove that the scaled iterates are defined by Tchebychev polynomials that are, in the usual minimax sense, optimal in a subspace of polynomials in A^TA. In the case of symmetric positive definite matrices, we get another speedup by a factor of two through a new means of constructing the initial iterate. We also consider adaptive procedures for which we prove no such *a priori* lower bound on speedup, but which promise to be useful in practice. Our results have further applications, in particular, to signal processing and to computing the SVD of a matrix.

Concerning efficiency in highly parallel computing environments, we do not claim that the present methods are always advantageous. The alternative, for most computations discussed here, is a parallel computation of the full SVD of A. The computation of the SVD in a highly parallel environment is best done using a square array of $\frac{n}{2}\times\frac{n}{2}$ processors, implementing a Jacobi-like method due to Kogbetliantz [3]. (This method is not competitive with standard methods in terms of operation count, but the standard methods are not well suited to highly parallel architectures). Good experimental evidence is available to show that from six to ten sweeps of the Kogbetliantz method are needed. For real A, each sweep requires $8n^3$ multiplications. The sequential operation count of this method is therefore the same as that of 32 — 52 Newton iterations. Thus, Newton's method is competitive with a Kogbetliantz SVD for these problems on highly parallel computers. It is a clear winner if the condition number of $A(\varepsilon)$ is not too large, or if the adaptive acceleration we employ is especially successful, so that far fewer than 30 Newton steps are needed, or if we can exploit sparsity or other properties of A to reduce its cost. Furthermore, it is often the case on parallel machines that matrix products, the core of the Newton iteration, can be computed especially fast. (On the Connection Machine, microcoded matrix multiply runs an order of magnitude faster than code written in Fortran, C*, or *LISP.)

Of course, other parallel methods for the SVD may arise. And when the number of processors is not large, so that $O(n^2)$ processors are not available, then more modestly parallel, but less costly methods for the SVD are better.

1.1 Contents.

We organize the paper as follows: Section 2 is for definitions. In Section 3, we recall the customary Newton iteration for matrix inversion; in Section 4 we present a Tchebychev acceleration procedure and in Section 5 an adaptive acceleration using cubic polynomials rather than the quadratics used in Newton's method. We give a method for finding an improved initial iterate for symmetric positive definite A in Section 6. In Sections 7-9 we give methods for computing the matrices $A(\varepsilon)$ and $A^+(\varepsilon)$ (see above), prove the stability of these computations, show how to find subspaces spanned by singular vectors, and show some further applications. Some numerical experiments are presented in Section 10.

2. Notation

We denote most matrices by upper case Roman letters, diagonal matrices by upper case Greek letters, and the elements of a matrix by the corresponding lower case Greek letter. We also use lower case Greek letters for various scalars, lower case bold Roman letters for vectors and in particular, columns of matrices. The letters i, j, k, m, n, r, and s are used for integers. We assume that all the quantities are real; extension to the complex case is straightforward.

The basis for our analysis is the *singular value decomposition* (hereafter referred to as the *SVD*) of $A \in R^{m \times n}$,

$$A = U \Sigma V^T , \tag{2.1}$$

where $U = [\mathbf{u}_1, \cdots, \mathbf{u}_m]$ and $V = [\mathbf{v}_1, \cdots, \mathbf{v}_n]$ are square orthogonal matrices and $\Sigma = \mathrm{diag}(\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0)$. Here, $r = \mathrm{rank}(A)$. The generalized inverse of A is

$$A^+ = V \Sigma^+ U^T$$

where

$$\Sigma^+ \equiv \mathrm{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \cdots, \sigma_r^{-1}, 0, \cdots, 0).$$

If A is symmetric and positive definite, then $m = n$ and $U = V$. In this case, the eigenvalues of A are σ_j and the eigenvectors are $\mathbf{v}_j$, $1 \leq j \leq n$.

We shall use the Euclidean vector norm $\|\mathbf{x}\|_2 \equiv (\mathbf{x}^T \mathbf{x})^{1/2}$ and the following matrix norms:

$$\|A\|_2 \equiv \max_{\mathbf{x} \neq 0} \|A\mathbf{x}\|_2 / \|\mathbf{x}\|_2 ,$$

$$||A||_1 \equiv \max_j \sum_i |\alpha_{ij}| ,$$

$$||A||_\infty \equiv \max_i \sum_j |\alpha_{ij}| ,$$

$$||A||_F \equiv \left[\sum_{i,j} \alpha_{ij}^2 \right]^{1/2} .$$

It is known [6] that if A has the SVD (2.1) then $||A||_2 = \sigma_1$, and $||A||_F = ||\Sigma||_F = (\sum_j \sigma_j^2)^{1/2}$. The following theorem (the Eckart-Young Theorem) can be found in [6]. p. 19.

Theorem 2.1. *Let* A *be a matrix of rank* r *with SVD (2.1). Let* $s \leq r$ *be an integer, and let*

$$A_s \equiv \sum_{j=1}^{s} u_j \sigma_j v_j^T .$$

Then

$$||A - A_s||_2 = \min_{\mathrm{rank}(B)=s} ||A - B||_2 = \sigma_{s+1} .$$

We shall make use, too, of the spectral condition number $\kappa(A)$ of A, which is defined by

$$\kappa(A) \equiv ||A||_2 \, ||A^+||_2 = \sigma_1 / \sigma_r. \tag{2.2}$$

The chief reason for being interested in the generalized inverse is that the solution to the least squares problem

$$\min ||Ax - b||_2$$

having smallest norm $||x||_2$ is A^+b.

3. Newton iteration

Newton iteration

$$X_{k+1} = X_k(2I - AX_k) = (2I - X_kA)X_k \tag{3.1}$$

was proposed by Schulz in 1933 for computing the inverse of a nonsingular matrix A [16]. Much later, Ben-Israel and Cohen proved that the iteration converges to A^+ provided that

$$X_0 = \alpha_0 A^T, \tag{3.2}$$

with α_0 positive and sufficiently small [1], [2].

All the following discussion employs an SVD-based analysis of the method, first developed by Soderstrom and Stewart [17], who observed that (2.1), (3.1) and (3.2) imply that each iterate has an SVD of the form

$$X_k = V \Xi_k U^T$$

with the matrices U and V of (2.1). The products $X_k A$ therefore satisfy

$$X_k A = V\, R_k V^T \tag{3.3}$$

where $R_k = \Xi_k \Sigma \equiv \mathrm{diag}(\rho_1^{(k)}, \rho_2^{(k)}, \cdots, \rho_n^{(k)})$; moreover, for all $1 \le j \le n$ and all $k \ge 0$,

$$1 - \rho_j^{(k+1)} = (\rho_j^{(k)} - 1)^2. \tag{3.4}$$

Clearly, $\rho_j^{(0)} = \alpha_0 \sigma_j^2$. For any $\alpha_0 < 2/\sigma_1^2$, (3.4) implies the quadratic convergence of $\rho_j^{(k)}$ to one for all j, $1 \le j \le r$, and, therefore, of X_k to A^+. The optimum choice of α_0 in (3.2), which minimizes $\| I - X_0 A \|_2$ by making $\rho_1^{(0)} - 1 = 1 - \rho_r^{(0)}$, is

$$\alpha_0 = \frac{2}{\sigma_1^2 + \sigma_r^2}. \tag{3.5}$$

Let $\kappa(A)$ be given by (2.2). Then with the choice (3.5),

$$\rho_r^{(0)} = \frac{2}{1 + \kappa(A)^2}, \tag{3.6}$$

so that $\rho_r^{(0)} \ll 1$ if $\kappa(A)$ is large. In practice, information about σ_r is hard to get. We may instead use a suboptimal but nevertheless safe alternative, such as

$$\alpha_0 = \frac{1}{||A||_1 ||A||_\infty}. \tag{3.7}$$

Other choices of α_0 which do not require an estimate of σ_r are available [1], [2], [11], [12].

It follows from (3.4) that small singular values $\rho_j^{(k)}$ approximately double at every step. Therefore, it takes about $2\log_2\kappa(A)$ steps of (3.1) to get to the point where $\rho_r^{(k)} \geq \frac{1}{2}$ and an additional $\log\log(1/\varepsilon)$ iterations for convergence of all the ρ_j to within ε of 1. Thus, if the floating-point operations are the measure of cost, the method is more expensive than the conventional alternatives. The reason for our interest in this method is that (3.1) essentially amounts to two matrix multiplications, which can be very efficiently implemented on systolic arrays and on vector and parallel computers ([13], [14]). Further savings are possible: if A is sparse, then X_kA is less costly to compute; in the case of Toeplitz and many other structured matrices, AX_k amounts to a few matrix-vector products (see [8], [9], [10]). Finally, as only the product of A and some vectors is required, we do not need to form A, which is convenient in some applications.

In operations counts, we use the term "flop" to mean a multiply-add pair. For unstructured $m \times n$ matrices, each iteration step (3.1) essentially amounts to two matrix multiplications, that is to $1.5mn^2$ flops, exploiting the symmetry of AX_k; in Section 5 we show how to reduce this to about $mn^2 + \frac{1}{2}n^3$ in a practical implementation.

4. Convergence Acceleration by Scaling

Schreiber [14] presented the scaled iteration

$$X_{k+1} = \alpha_{k+1}(2I - X_kA)X_k, \qquad k = 0, 1, \cdots \tag{4.1}$$

where X_0 is given by (3.2). Here, we employ an acceleration parameter $\alpha_{k+1} \in [1,2]$ chosen so as to minimize a bound on the maximum distance of any nonzero singular value of $X_{k+1}A$ from one. Let us assume that we know bounds $\sigma_{\min}$ and $\sigma_{\max}$ on the singular values of A satisfying

$$\sigma_{\min} \le \sigma_r^2 \le \sigma_1^2 \le \sigma_{\max}.$$

In this case, every eigenvalue of A^TA lies in the interval $[\sigma_{\min}, \sigma_{\max}]$. (We assume this interval has nonzero length; otherwise A is a scalar multiple of an orthogonal matrix and X_0 is its inverse.) We then choose X_0 according to (3.2) with

$$\alpha_0 = \frac{2}{\sigma_{\min} + \sigma_{\max}}. \tag{4.2}$$

It follows from (2.1), (3.2), (4.1) and (4.2) that for all j and k,

$$\rho_j^{(0)} = \frac{2\sigma_j^2}{\sigma_{\min} + \sigma_{\max}},$$

and

$$\rho_j^{(k+1)} = \alpha_{k+1}(2 - \rho_j^{(k)})\rho_j^{(k)}.$$

To determine the acceleration parameters α_k, for k > 0, we let

$$\underline{\rho}^{(0)} = \alpha_0\sigma_{\min} = \frac{2\sigma_{\min}}{\sigma_{\min} + \sigma_{\max}} \tag{4.3}$$

and

$$\bar{\rho}^{(0)} = \alpha_0\sigma_{\max} = \frac{2\sigma_{\max}}{\sigma_{\min} + \sigma_{\max}}; \tag{4.4}$$

these being lower and upper bounds on the singular values of X_0A. Then take

$$\alpha_{k+1} = \bar{\rho}^{(k+1)} = \frac{2}{1+(2-\underline{\rho}^{(k)})\underline{\rho}^{(k)}}, \tag{4.5}$$

which is both an acceleration parameter and an upper bound on $\{\rho_j^{(k+1)}\}$, and

$$\underline{\rho}^{(k+1)} = \alpha_{k+1}(2-\underline{\rho}^{(k)})\underline{\rho}^{(k)}, \tag{4.6}$$

which is a lower bound on $\{\rho_j^{(k+1)}\}$. The definitions (4.3)-(4.6) imply that for all $1 \le j \le r$ and $k \ge 1$,

$$\underline{\rho}^{(k)} \le \rho_j^{(k)} \le \bar{\rho}^{(k)},$$

and

$$\underline{\rho}^{(k)} = 2 - \bar{\rho}^{(k)}. \tag{4.7}$$

(Note that (4.7) follows from immediately (4.5) and (4.6). The upper bound $\rho_j^{(k)} \le \bar{\rho}^{(k)}$ is likewise straightforward. Finally, if

$$0 \le \underline{\rho}^{(k)} \le \rho_j^{(k)} \le (2-\underline{\rho}^{(k)}),$$

then

$$(2-\underline{\rho}^{(k)})\underline{\rho}^{(k)} \le (2-\rho_j^{(k)})\rho_j^{(k)} \le 1,$$

whence, by (4.6) and the definition of $\rho_j^{(k)}$, the lower bound $\underline{\rho}^{(k)} \le \rho_j^{(k)}$ follows.)

Except for the last few iterations before convergence, $\underline{\rho}^{(k)} \ll 1$, which implies that $\alpha_{k+1} \approx 2$. Thus, $\rho_r^{(k+1)} \approx 4\rho_r^{(k)}$. Therefore,

$$-\log_4 \rho_r^{(0)} \approx \log_2[(\kappa(A)^2 + 1)^{1/2}] = \log_2 \kappa(A) + O(1/\kappa(A)^2)$$

steps suffice to bring all the singular values of $X_k A$ up to ½, which is half as many as for the unaccelerated version.

We shall now derive a theorem concerning the optimality of the acceleration parameters given by (4.5). Let the symmetric residual matrix initially be

$$E \equiv I - \alpha_0 A^T A = I - X_0 A. \tag{4.8}$$

It is straightforward to show that Newton's method, starting with $X_0 = \alpha_0 A^T$, produces iterates X_k satisfying

$$X_k A = I - E^m \tag{4.9}$$

where $m = 2^k$. For a nonsingular matrix A and for the choice (4.2) for α_0, the eigenvalues of E lie in the open interval $(-1,1)$, and we have that $E^m \to 0$ and, therefore, $X_k A \to I$ as $k \to \infty$.

Furthermore, (4.8) and (4.9) imply that

$$X_k = (I + E + \cdots + E^{m-1})\, \alpha_0 A^T.$$

Thus, Newton's method is related to the Neumann series expansion

$$(I-E)^{-1} = I + E + E^2 + \cdots$$

We therefore ask whether the accelerated method (3.2), (4.1) — (4.6) is related to a better polynomial approximation to $(I-E)^{-1}$. In fact, it is exactly equivalent to approximation of this inverse by a Tchebychev polynomial in E, as we now show.

Let

$$T_{2^k}(\zeta) \equiv \cos\,(2^k \cos^{-1}\zeta)$$

be the Tchebychev polynomial of degree 2^k on $(-1,1)$. Recall that $T_0(\zeta) = 1$, $T_1(\zeta) = \zeta$, and

$$T_{2^{k+1}}(\zeta) \equiv 2\, T_{2^k}^2(\zeta) - 1. \tag{4.10}$$

The scaled Tchebychev polynomials on $(\sigma_{\min},\sigma_{\max})$ are defined by

$$t_k(\zeta) \equiv \frac{T_{2^k}(\gamma\zeta + \delta)}{T_{2^k}(\delta)}$$

where $\gamma = \frac{2}{(\sigma_{\max} - \sigma_{\min})}$ and $\delta = \frac{-(\sigma_{\max} + \sigma_{\min})}{(\sigma_{\max} - \sigma_{\min})}$. Surely, t_k is a polynomial of degree 2^k, and $t_k(0) = 1$, so that

$$t_k(\zeta) = 1 - \zeta\,\bar{t}_k(\zeta)$$

for some polynomial $\bar{t}_k$ of degree 2^k-1; furthermore, it is a classical result that among all such polynomials, t_k minimizes the norm $\sup_{\sigma_{\min} \le \zeta \le \sigma_{\max}} |t_k(\zeta)| \equiv \| t_k \|_{\infty,(\sigma_{\min},\sigma_{\max})}$. They also satisfy a recurrence like (4.10). Let $\beta_k \equiv T_{2^k}(\delta)$. Then by (4.10),

$$\beta_k t_k(\zeta) = 2(\beta_{k-1} t_{k-1}(\zeta))^2 - 1. \tag{4.11}$$

Theorem 4.1. *Let the sequence of matrices* X_k, $k = 0, 1, \cdots$ *be generated by* (3.2), (4.1) — (4.6). *Then*

$$X_k = \bar{p}_k\,(A^T A) A^T \tag{4.12}$$

where $\bar{p}_k(\zeta)$ *is a polynomial of degree* 2^k-1. *The polynomial* $1 - \zeta\,\bar{p}_k(\zeta)$ *is the scaled Tchebychev polynomial* $t_k(\zeta)$ *of degree* 2^k *on* $(\sigma_{\min},\sigma_{\max})$.

Remark. Before proving the theorem, we point out that (4.12) implies that

$$I - X_k A = I - p_k(A^T A),$$

and therefore that

$$\| I - X_k A \|_2 = \max_{1 \le j \le n} | 1 - p_k(\sigma_j^2) |,$$

which shows the relevance of the theorem's conclusion.

An analogue of this result also holds for X_0 any matrix of the form $r(A^TA)A^T$, with r a polynomial, such that the 2-norm of $I - X_0A$ is less than one.

Proof. The claim (4.12) is clearly true when $k=0$, with $\bar{p}_0(\zeta) \equiv \alpha_0$. With the choice (4.2), $1 - \zeta\bar{p}_0(\zeta) = 1 - 2\zeta / (\sigma_{max}+\sigma_{min}) = t_0(\zeta)$ is the appropriate scaled Tchebychev polynomial.

Now use induction on k. A straightforward calculation using (4.5) and (4.7) shows that, for all $k \geq 1$, $1<\alpha_k<2$ and $\beta_k = \frac{1}{\alpha_k - 1}$, or equivalently that $\alpha_k = \frac{(1+\beta_k)}{\beta_k}$. It also follows from (4.1) and the inductive hypothesis in the form (4.12), after multiplying by A, that (4.12) holds for the polynomial

$$\bar{p}_k(\zeta) = \alpha_k(2-\zeta\bar{p}_{k-1}(\zeta))\bar{p}_{k-1}(\zeta).$$

From this and the relations between α_k and β_k we derive the recurrence

$$\beta_k(1 - \zeta\bar{p}_k) = 2\beta_{k-1}^2(1 - \zeta\bar{p}_{k-1})^2 - 1.$$

Thus, $\beta_k(1 - \zeta\bar{p}_k)$ satisfies the Tchebychev recurrence (4.11), which proves the theorem.

QED

5. Convergence Acceleration with Cubic Polynomials

In Section 3 we saw that with the unaccelerated Newton method, the convergence of $\rho_j^{(k)}$ to one is slow for all j such that $\rho_j^{(0)}$ lies near zero or two. For many input matrices, and for moderately large k, the set $\{\rho_j^{(k)}\}_{j=0}^{n}$ produced by Newton's method without acceleration consists of two clusters, one lying near zero and the other near one. In this section, we present an alternative to the acceleration method of Section 4 that, in the case of such a large gap in the spectrum of $X_k A$, results in much faster convergence.

Let X be a fixed matrix satisfying $XA = VRV^T$, $R = \mathrm{diag}(\rho_1, \cdots, \rho_n)$ where $0 \le \rho_j \le 2$ for all j. We seek an improved approximation X_1 to A^+ of the form

$$X_1 = (\gamma_3 (XA)^2 + \gamma_2 XA + \gamma_1 I)X.$$

We choose $\{\gamma_1, \gamma_2, \gamma_3\}$ so that the cubic polynomial $c(\rho) = \gamma_1 \rho + \gamma_2 \rho^2 + \gamma_3 \rho^3$ satisfies

(i) $c(1) = 1$,

(ii) $c'(1) = 0$,

(iii) $c(0) = 0$,

(iv) $c'(0) \gg 2$.

The idea here is that small singular values are amplified by the factor $c'(0)$ while those near one continue to converge. It is quite evident, however, that $c(\rho)$ will take large values for some $\rho \in (0,1)$, so we must exercise caution.

We begin by finding $\underline{\rho} > 0$ such that we are certain that there are no eigenvalues ρ_j in $(\underline{\rho}, 1-\underline{\rho})$ or in $(1+\underline{\rho}, 2]$. Let $T = XA$ and compute T^2 and $\delta \equiv \|T - T^2\|_F$. Now, it follows from (3.3) that

$$\delta^2 = \sum_{j=1}^{n} [\rho_j(1-\rho_j)]^2; \tag{5.1}$$

whence, for all $1 \le j \le n$,

$$\rho_j \, | 1 - \rho_j | \le \delta .$$

If $\delta \ge 1/4$, this provides us with no useful information. If, on the contrary, $0 < \delta < 1/4$, then we conclude that all the eigenvalues ρ_j lie in the two closed intervals: $[0, \underline{\rho}]$ and $[1 - \underline{\rho}, 1 + \bar{\rho}]$, where

$$0 \le \underline{\rho} \equiv 1/2 - \sqrt{1/4 - \delta} < 1/2 ,$$
$$1/2 < 1 - \underline{\rho} = 1/2 + \sqrt{1/4 - \delta} ,$$

and

$$1 \le 1 + \bar{\rho} = 1/2 + \sqrt{1/4 + \delta} < 1 + \underline{\rho} .$$

(See Figure 1).

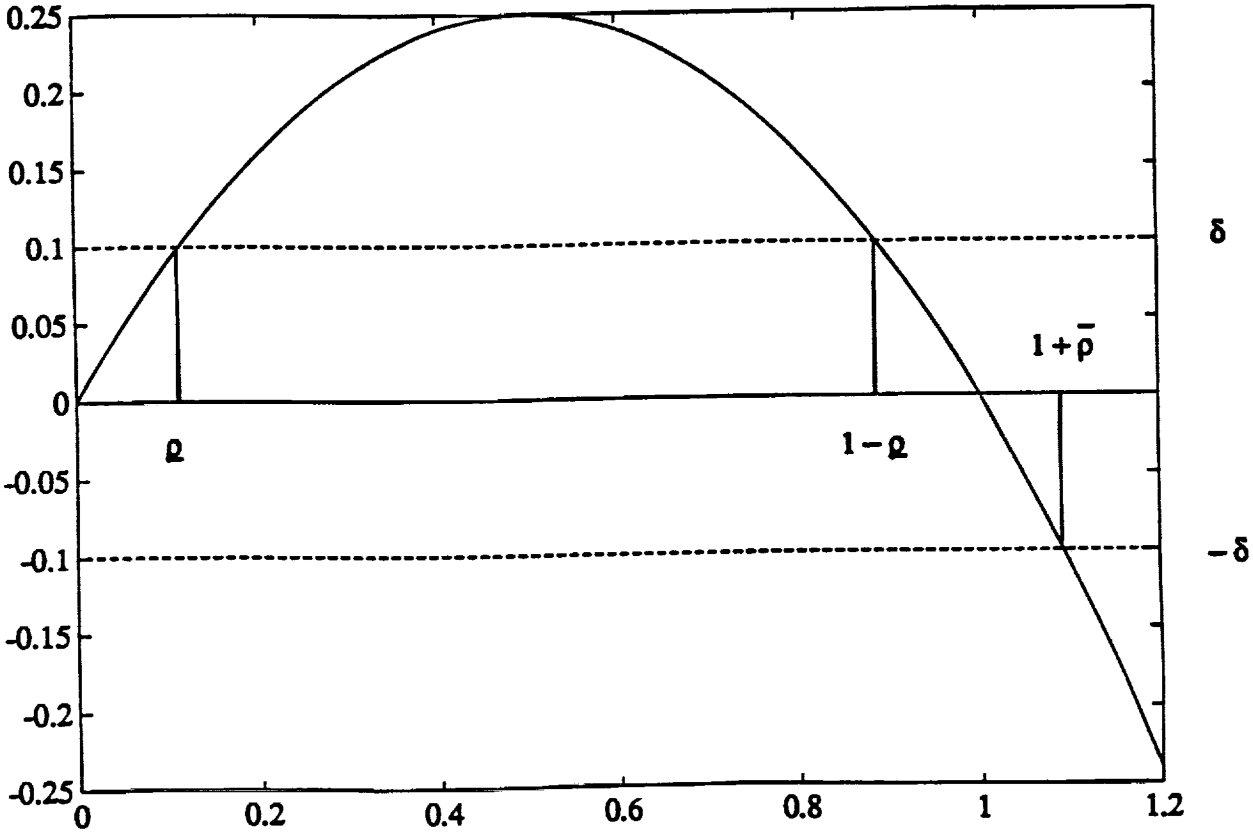


Figure 1. The intervals $[0, \underline{\rho}]$ and $[1 - \underline{\rho}, 1 + \bar{\rho}]$ contain all the ρ_j when $\delta = \| XA - (XA)^2 \|_F < 1/4$.

Thus, for $j = 1, \cdots, n$,

$$\rho_j \in [0, \underline{\rho}] \cup [1 - \underline{\rho}, 1 + \bar{\rho}] . \tag{5.2}$$

in other words ρ_j is in neither $(\underline{\rho},1-\underline{\rho})$ nor $(1+\bar{\rho},2]$. Now we show how to choose $c(\rho)$ so as to satisfy the criteria (i)-(iv), as well as the equally important criteria

(v) $c: [0,\underline{\rho}] \rightarrow [0,1]$,

(vi) $c: [1-\underline{\rho},1+\bar{\rho}) \rightarrow [1,1+\underline{\rho})$.

To determine c, we enforce (i)-(iii) together with

(vii) $c(\underline{\rho}) = 1$.

The unique solution is

$$c(\rho) = \frac{1}{\underline{\rho}}(\rho^2 - (2+\underline{\rho})\rho + (1+2\underline{\rho}))\rho,$$

which may be rewritten as

$$c(\rho) = \frac{1}{\underline{\rho}}((\rho-1)^3 + (1-\underline{\rho})(\rho-1)^2) + 1.$$

Theorem 5.1. *Let* $0 \leq \underline{\rho} \leq \frac{1}{2}$. *If* $c(\rho)$ *is the unique cubic satisfying (i), (ii), (iii), and (vii) above, then (v) and (vi) hold.*

Proof: To show that (v) holds, we recall that a nonvanishing cubic polynomial $c(\rho)$ has at most two critical points (where $c'(\rho) \equiv 0$). One of these is $\rho = 1$ (condition (ii)). By (i), (vii) and Rolle's theorem, the other one is in $(\underline{\rho},1)$, so $c(\rho)$ is monotone on $[0,\underline{\rho}]$ and must therefore map $[0,\underline{\rho}]$ onto $[0,1]$.

To establish (vi), note that

$$c(\rho) = 1 + \frac{(\rho-1)^2}{\underline{\rho}}(\rho - \underline{\rho}).$$

Now let $\rho = 1 - \varepsilon$ with $|\varepsilon| \leq \underline{\rho} < 1/2$. Then $c(\rho) - 1 = \varepsilon^2(1 - \varepsilon - \underline{\rho}) / \underline{\rho}$, and since the second factor is bounded by 0 and 1, the right-hand side is bounded by 0 and $\underline{\rho}$, establishing (vi).

QED

Thus, if $\delta \geq 1/4$ we cannot accelerate. In this case, we let $X_1 = (2I - T)X$ and proceed to the next iteration (i.e., X_1 is the result of a Newton step (3.1)). On the other hand, if $\delta < 1/4$ we compute $\underline{\rho} := 1/2 - \sqrt{1/4 - \delta}$ and let $X_1 = \frac{1}{\underline{\rho}}(T^2 - (2+\underline{\rho})T + (1+2\underline{\rho})I)X$ (i.e., X_1 is the result of a cubic step).

Here is a practical algorithm incorporating this idea for computing A^+.

```
ALGORITHM CUINV.
INPUT:  $A$
OUTPUT:  $A^+$

STEP 0 [INITIALIZE]:
      Choose $X = \alpha_0 A^T$, with $\alpha_0$ given by (4.2) or (3.7);
      $T := XA$;
      T2VALID := false;

STEP 1 [NEWTON STEP]:
      if X is sufficiently close to $A^+$ then return(X);
      $X := (2I-T)X$;
      if T2VALID then
            $T := 2T-T_2$
      else
            $T := XA$;
      endif
      T2VALID := false;
      if trace(T) $\geq n - \frac{1}{2}$ then
            goto STEP 1;

STEP 2 [TEST FOR SMALL CHANGE]:
      $T_2 := T^2$;
      $\delta := \| T-T_2 \|_F$;
      if $\delta \geq \frac{1}{4}$ then
            T2VALID := true;
            goto STEP 1;
      else
            goto STEP 3;

STEP 3 [USE ACCELERATION]:
      $\rho := \frac{1}{2} - \sqrt{\frac{1}{4} - \delta}$;
      $X := \frac{1}{\rho}(T_2-(2+\rho)T+(1+2\rho)I)\, X$;
      $T := XA$;
      goto STEP 1;
```

COMMENTS on ALGORITHM CUINV.

(1) Stopping criteria are discussed by Soderstrom and Stewart [17].

(2) First, we have made use of the fact that after a Newton step (3.1),

$$\begin{aligned} T_{k+1} &\equiv X_{k+1}A \\ &= (2I-X_kA)X_kA \\ &= (2I-T_k)T_k. \end{aligned}$$

Thus, if we decide to reject the use of a cubic acceleration step, the computation of T^2 in STEP 2 is not wasted, because it saves us at least that much work in the following Newton step (STEP 1).

(3) We do not use cubic steps exclusively; we do need at least one Newton step for each cubic step because without Newton steps, we cannot find intervals $[0,\underline{\rho}]$ and $[1-\underline{\rho}, 1+\underline{\rho}]$ known to contain all the eigenvalues. Furthermore, the Newton steps finish the job of forcing eigenvalues near one to actually converge.

(4) We use only Newton steps when all the singular values of XA are close to one. This is signaled by the convergence of trace(XA) to within one half of its limiting value of n.

If A is an m×n matrix, then we assume that the cost of computing the product XA is $\frac{1}{2}mn^2$ flops (we exploit the symmetry of XA), the cost of computing the product T^2 is $\frac{1}{2}n^3$ flops, and the cost of computing X in STEP 1 or STEP 3 is mn^2 flops. Then, if we skip STEP 3, an iteration of CUINV costs

$$n^2m + \tfrac{1}{2}n^3 \text{ flops,}$$

assuming that T2VALID was true. The cost of an iteration in which we take STEPS 1 — 3 is

$$3mn^2 + \tfrac{1}{2}n^3,$$

assuming that T2VALID was false. These assumptions are warranted since it is the iterations that take STEP 3 that cause T2VALID to be false. Thus, for $n \approx m$, we pay a premium of only about 16% compared with two Newton steps.

What is the effect of a full iteration assuming STEP 3 is taken? Formally, the eigenvalues of XA are mapped as follows:

$$\rho \rightarrow c((2-\rho)\rho) = 1+\frac{(1-\underline{\rho})}{\underline{\rho}}(1-\rho)^4 - \frac{1}{\underline{\rho}}(1-\rho)^6 \equiv C(\rho;\underline{\rho}). \tag{5.3}$$

For small ρ, $C(\rho;\underline{\rho}) \approx (4+\frac{2}{\underline{\rho}})\rho$. And because it will often be the case that $\underline{\rho} \ll 1$, the combined iteration greatly amplifies the small eigenvalues. Those near one continue to converge superlinearly; equation (5.3) shows that

$$|1-C(\rho;\underline{\rho})| = \frac{1-\underline{\rho}}{\underline{\rho}}(1-\rho)^4 + O((1-\rho)^6)$$

$$\leq (1-\underline{\rho})(1-\rho)^3 + O((1-\rho)^6) = O((1-\rho)^3)\,.$$

since any eigenvalue ρ of XA exceeding $\underline{\rho}$ must be in $[1-\underline{\rho}, 1+\underline{\rho}]$, so $\underline{\rho} \geq 1-\rho$.

See Section 10 for some experimental evidence of the efficiency and reliability of ALGORITHM CUINV.

6. An Improved Initial Iterate in the Symmetric Positive Definite Case

Let us consider the case of a symmetric positive definite matrix A, which is important in many applications [6]. Given a matrix A with the SVD

$$A = V\Sigma V^T, \qquad \Sigma = \mathrm{diag}(\sigma_1 \geq \cdots \geq \sigma_n > 0) \tag{6.1}$$

(which is its eigendecomposition, too) we shall choose

$$X_0 = \beta I + \alpha_0 A \tag{6.2}$$

so as to optimally place the singular values of X_0A. From (6.1) and (6.2)

$$X_0A = V(\beta\Sigma + \alpha_0\Sigma^2)V^T\,.$$

We choose (β,α_0) so that, with $p(\sigma) \equiv \beta\sigma + \alpha_0\sigma^2$, the largest possible initial error

$$\|I - X_0A\|_2 \equiv \max_{\sigma_n \leq \sigma \leq \sigma_1} |p(\sigma) - 1|$$

is minimized. By standard arguments (see [5], ch. 9), $1-p$ is a scaled Tchebychev polynomial on (σ_n,σ_1):

$$1 - p(\sigma) = T_2(\sigma) = \frac{2(\sigma-\sigma_{mid})^2 - \delta^2}{2\sigma_{mid}^2 - \delta^2}$$

where $\sigma_{mid} = ½(\sigma_n + \sigma_1)$ and $\delta = \sigma_1 - \sigma_{mid}$. Moreover,

$$\|I - X_0A\|_2 = \max_{\sigma_n \leq \sigma \leq \sigma_1} |T_2(\sigma)| = T_2(\sigma_1) = \frac{\delta^2}{2\sigma_{mid}^2 - \delta^2}\,.$$

We are most concerned about $\rho_n^{(0)}$ the smallest singular value of X_0A, which is

$$p(\sigma_n) = \frac{2(\sigma_{mid}^2 - \delta^2)}{2\sigma_{mid}^2 - \delta^2}\,.$$

Let $\omega \equiv \delta/\sigma_{mid} < 1$. The condition number κ is given by $\kappa = \sigma_1/\sigma_n$. Thus

$$\kappa = \frac{\sigma_{mid} + \delta}{\sigma_{mid} - \delta} = \frac{1+\omega}{1-\omega}\,;$$

whence

$$\omega = \frac{\kappa - 1}{\kappa + 1} .$$

Thus, the smallest initial singular value of X_0A is

$$\begin{aligned} p(\sigma_n) &= \frac{2(1-\omega^2)}{2-\omega^2} \\ &= \frac{2((\kappa+1)^2-(\kappa-1)^2)}{2(\kappa+1)^2-(\kappa-1)^2} \\ &= \frac{8\kappa}{\kappa^2+6\kappa+1} . \end{aligned} \tag{6.3}$$

This asymptotically is $8\kappa^{-1} + O(\kappa^{-2})$, which is far larger than the $2\kappa^{-2}$ provided by the "optimal" choice $X_0 = \alpha_0 A$ (see (3.6))! To find β, α_0, and hence X_0, we use the equations

$$\begin{aligned} \beta\sigma + \alpha_0\sigma^2 = p(\sigma) &= 1 - T_2(\sigma) \\ &= \frac{2\sigma_{mid}^2 - \delta^2 - 2(\sigma - \sigma_{mid})^2 + \delta^2}{2\sigma_{mid}^2 - \delta^2} \\ &= \frac{-2\sigma^2 + 4\sigma\sigma_{mid}}{2\sigma_{mid}^2 - \delta^2} . \end{aligned}$$

Hence

$$\beta \equiv \frac{4\sigma_{mid}}{2\sigma_{mid}^2 - \delta^2}; \qquad \alpha_0 \equiv -\frac{2}{2\sigma_{mid}^2 - \delta^2} . \tag{6.4}$$

Remark. Even without computing estimates or bounds for the extreme singular values we may improve the initial approximation when A is symmetric positive definite by choosing $X_0 = (1 / \| A \|_F) I$; for in this case $\| I - X_0A \|_2 \le 1 - 1/(n^{1/2}\kappa)$.

7. Suppressing the Smaller Singular Values

In this section, given a matrix A and a positive scalar ε, we show how to compute $A(\varepsilon)$ and $A^+(\varepsilon)$, where $A(\varepsilon)$ is obtained from A by suppressing (that is, setting to zero) all the singular values of A that do not exceed ε . (As noted in Section 2 above, $A(\varepsilon)$ is a closest approximation to A by matrices whose rank is that of $A(\varepsilon)$. In practice, solving least squares problems often requires the use of this form of regularization; for when A is very badly conditioned, its generalized inverse is largely unknowable due to perturbations in A, but the reduced-rank generalized inverses are much better conditioned. This idea is discussed more fully by Golub and Van Loan

[6, Section 5.5].)

In order to compute these rank-reduced generalized inverses, we require a polynomial $c(\rho)$ that gives fast convergence to 0 near $\rho = 0$ and to 1 near $\rho = 1$. Consider first the iteration

$$X_{k+1/2} = (2I - X_k A)X_k, \qquad X_{k+1} = X_{k+1/2}\, A X_{k+1/2} \tag{7.1}$$

that was proposed by the second author [14]. The eigenvalues of $X_k A$ satisfy the equations $\rho_j^{(k+1)} = ((2-\rho_j^{(k)})\rho_j^{(k)})^2$ for all j and k. Figure 2 illustrates the effect of this mapping. The fixed points are $\hat{\rho}_0 = 0$, $\hat{\rho}_1 = (3-\sqrt{5})/2 = .3819 \cdots$, $\hat{\rho}_2 = 1$, and $\hat{\rho}_3 = (3+\sqrt{5})/2 = 2.618 \cdots$ which are the roots of the quartic $(2-\hat{\rho})^2\hat{\rho}^2 = \hat{\rho}$.

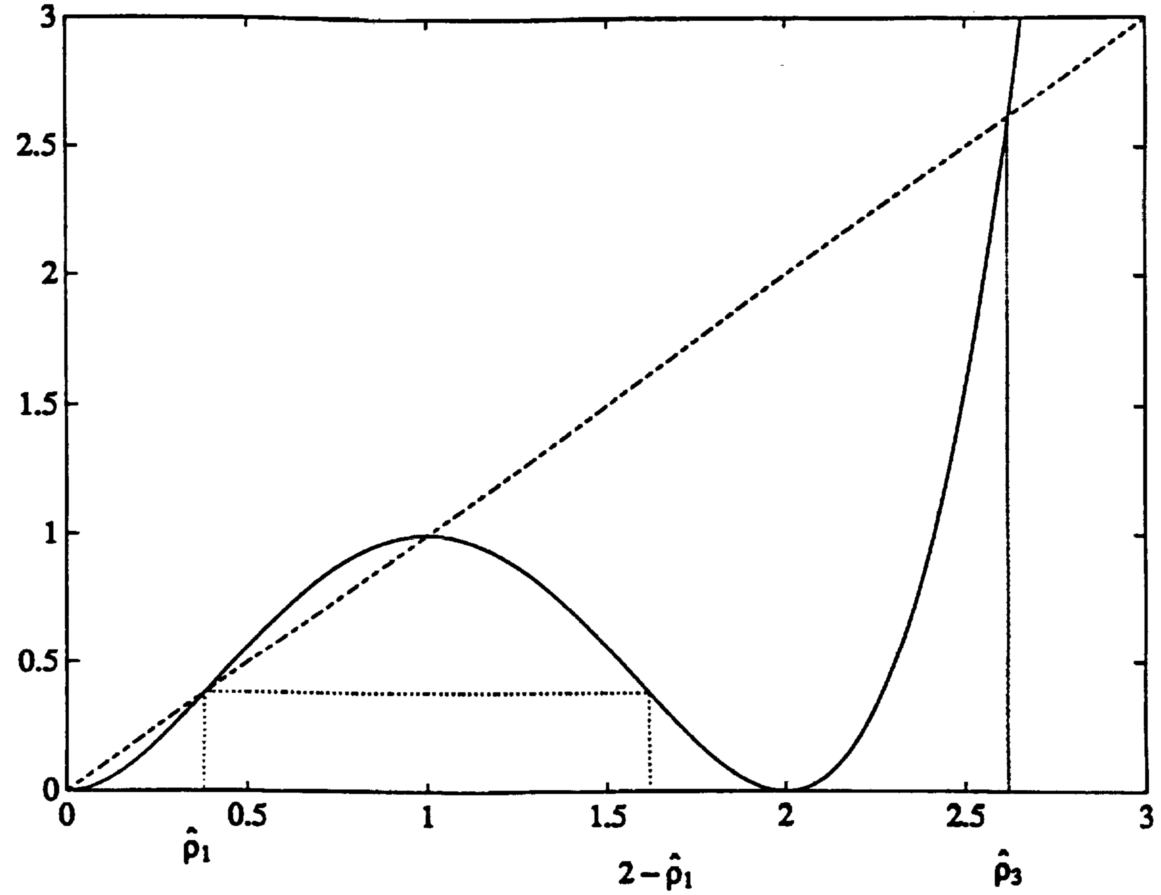


Figure 2. The mapping $(2-\rho)^2\rho^2$.

Consider especially the intervals $\{\rho: 0 < \rho < \hat{\rho}_1\}$ and $\{\rho: \hat{\rho}_1 < \rho < 2-\hat{\rho}_1\}$; note that $2-\hat{\rho}_1 = (1+\sqrt{5})/2 = 1.618 \cdots$ Evidently, the eigenvalues from the former interval are sent towards zero and from the latter interval towards one. The convergence is ultimately quadratic but is slow near $\hat{\rho}_1$ and $2-\hat{\rho}_1$. To compute $A^+(\varepsilon)$, it suffices to apply the iteration (4.1) — (4.6) with appropriate σ_{min} and σ_{max} until the following relations hold:

$$0 \le \rho_j^{(k)} < \hat{\rho}_1 \text{ if and only if } \sigma_j < \varepsilon, \tag{7.2}$$

$$\hat{\rho}_1 < \rho_j^{(k)} < 2-\hat{\rho}_1 \text{ otherwise.} \tag{7.3}$$

We can satisfy (7.2) for $k=0$ by setting $\alpha_0 = \hat{\rho}_1 / \varepsilon^2$, but then (7.3) may not hold. Therefore, we proceed as follows:

0) Compute an upper bound $\sigma_{\max}$ on σ_1^2.

1) Set $\alpha_0 = \min\{2 / (\sigma_{\max} + \varepsilon^2), \hat{\rho}_1 / \varepsilon^2\}$. This insures that the eigenvalues $\rho_j^{(0)}$ (of X_0A) corresponding to the singular values $\sigma_j \ge \varepsilon$ (of A) are all closer to one than the smaller eigenvalues. Set $\bar{\rho}^{(0)} = \alpha_0\sigma_{\max}$, $\underline{\rho}^{(0)} = \alpha_0\varepsilon^2$.

2) Apply the iteration (4.1) with parameters given by (4.5) and (4.6) until $\underline{\rho}^{(k)} \ge \hat{\rho}_1$.

3) Set $X_k := \left[\dfrac{\hat{\rho}_1}{\underline{\rho}^{(k)}}\right] X_k$.

4) Apply the iteration (7.1) until the matrix $A^+(\varepsilon)$ has been computed with the desired accuracy.

The scaling at Step (3) is done to insure that all the small singular values (less than ε) are in fact suppressed.

The iteration (7.1) associated with the quartic polynomial $(\rho(2-\rho))^2$ is not the most efficient way of computing $A^+(\varepsilon)$. The same objective can be achieved by using the iteration

$$X^{(k+1)} = (-2X^{(k)}A+3I)X^{(k)}AX^{(k)} \tag{7.4}$$

associated with the cubic polynomial $\tilde{c}(\rho) = -2\rho^3+3\rho^2$ (see Figure 3). Note that $\tilde{c}(1) = 1$, $\tilde{c}(0) = \tilde{c}'(0) = \tilde{c}'(1) = 0$, $\tilde{c}(1/2) = 1/2$, so that the mapping $\tilde{c}(\rho)$ has three nonnegative fixed points 0, 1/2 and 1. Let $\hat{\rho}_4 = (1+\sqrt{3})/2 = 1.366 \cdots$ be the unique solution to $-2\hat{\rho}_4^3 + 3\hat{\rho}_4^2 = 1/2$ greater than one. The eigenvalues of $X^{(k)}A$ in the interval $\{\rho: 0 \le \rho < 1/2\}$ are sent towards zero, and the eigenvalues in the interval $\{\rho: 1/2 < \rho \le \hat{\rho}_4\}$ are sent towards one; the convergence to zero and one is ultimately quadratic but is slow near 1/2 and $\hat{\rho}_4$.

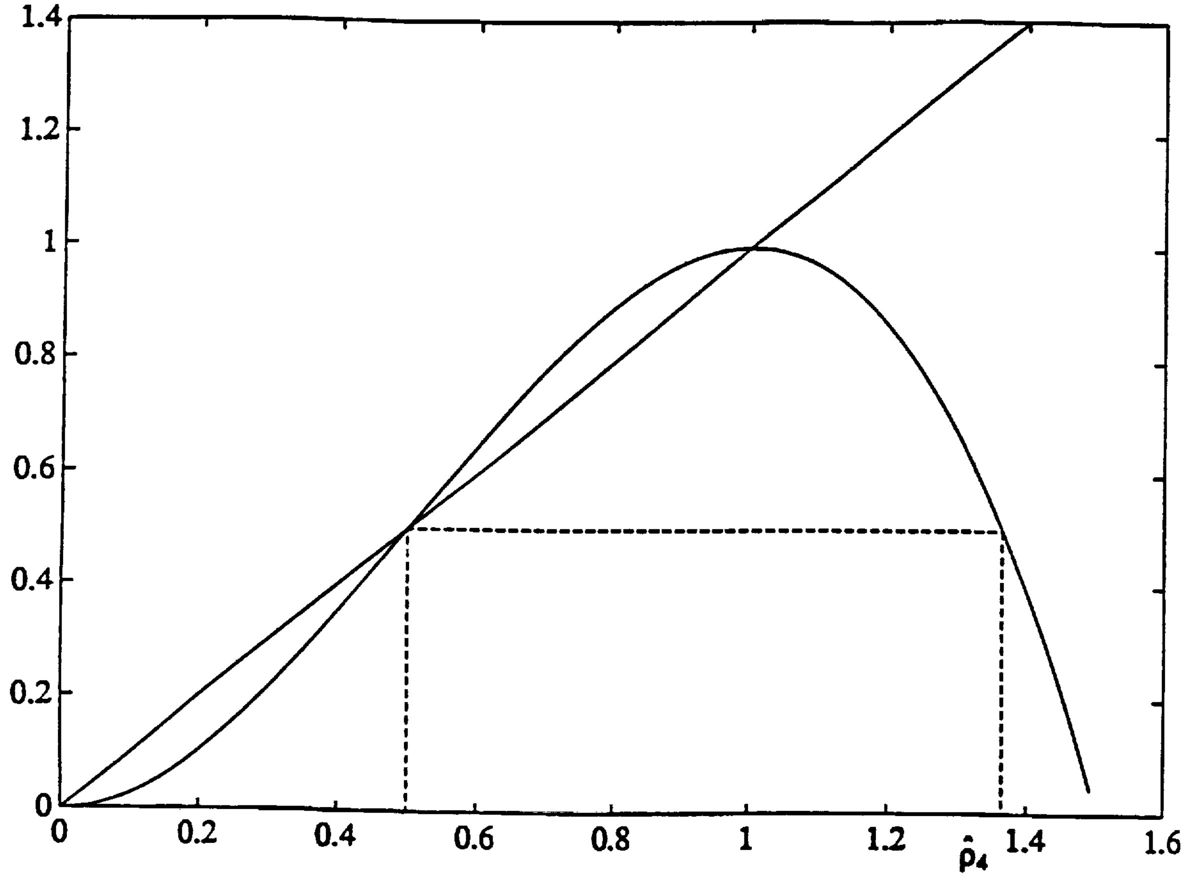


Figure 3. The mapping $-2\rho^3 + 3\rho^2$.

The iteration (7.4) is simpler than (7.1): it requires three matrix multiplications, one less than is needed in (7.1).

Remark 7.1. We may also compute $A(\varepsilon)$ itself since

$$A(\varepsilon) = AA^{+}(\varepsilon)A\,. \tag{7.5}$$

Remark 7.2. We may use either of the methods (7.1) or (7.4) to "split" a matrix A into two better-conditioned matrices $A(\varepsilon)$ and $\bar{A}(\varepsilon)$ such that

$$A = A(\varepsilon) + \bar{A}(\varepsilon)\,.$$

Moreover, if ε is placed where there is a large gap in the singular values of A, then faster convergence is possible. For A^+ may be computed by the formula

$$A^{+} = A^{+}(\varepsilon) + \bar{A}^{+}(\varepsilon)\,.$$

The matrices $A(\varepsilon)$ and $\bar{A}(\varepsilon)$ may be much better conditioned than A.

8. Stability of the Basic and Modified Iterations

It is well-known that Newton iteration (3.1) is numerically stable and even self-correcting if the input matrix A is nonsingular. If A is singular, however, then it is very mildly unstable.

Let A and A^+ have the SVDs

$$A = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^T,$$

and

$$A^+ = v \begin{bmatrix} \Sigma^+ & 0 \\ 0 & 0 \end{bmatrix} U^T,$$

In order to analyze the propagation of errors by (3.1), we assume that

$$X_k = A^+ + \hat{E} = V \begin{bmatrix} \Sigma^+ + E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix} U^T. \tag{8.1}$$

where $\hat{E} = VEU^T$ is the current error in X_k. We shall consider whether or not the Newton iteration amplifies these errors. Throughout this section we shall drop all terms of second order in E.

Using (8.1) it is simple to compute that

$$\begin{aligned} X_{k+1} &= X_k + (I - X_k A) X_k \\ &= \begin{bmatrix} \Sigma^+ & E_{12} \\ E_{21} & 2E_{22} \end{bmatrix}. \end{aligned}$$

Due to the block $2E_{22}$, the iteration (3.1) is mildly unstable if A is singular (in which case the (2,2) block above is not empty). After $2\log_2\kappa(A)$ iterations, rounding errors of order $\kappa^2(A)$ can accumulate. In Figure 4, this phenomenon is illustrated. Here, A was 6×6 with condition number 30 and rank four. The method converges in 17 iterations. All logarithms are base 10 and the Frobenius norm is used. Note that the norm of the off-diagonal part of VXV^T grows exponentially. It reaches a value about four orders of magnitude above machine precision (which is roughly 10^{-16}); a loss of four digits of precision results.

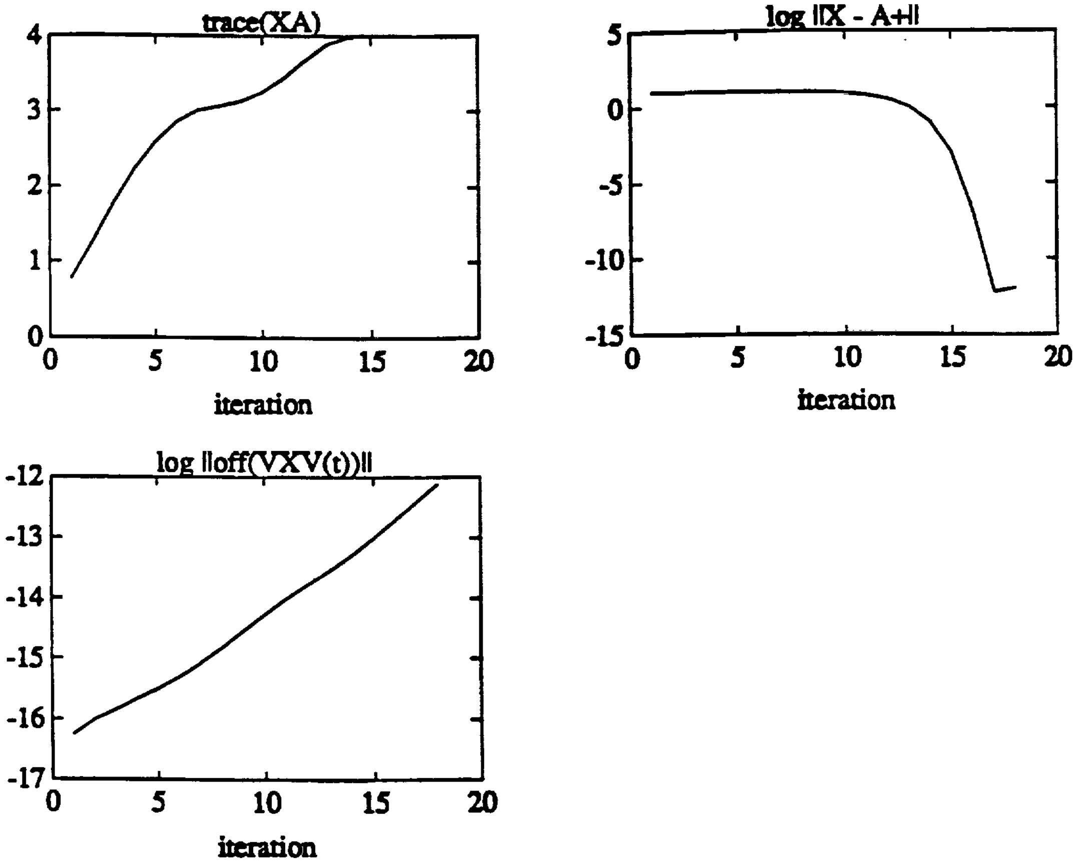


Figure 4. Mild Instability of Newton Iterations in the Rank-Deficient Case.

On the other hand, the iteration (7.1) is stable even for singular A. Indeed, by (7.1) and (8.1)

$$X^{(k+1/2)} = V \begin{bmatrix} \Sigma^{+} & E_{12} \\ E_{21} & 2E_{22} \end{bmatrix} U^T ,$$

hence

$$X^{(k+1/2)}A = V \begin{bmatrix} I & E_{12} \\ E_{21}\Sigma & 2E_{22} \end{bmatrix} V^T ,$$

and therefore,

$$\begin{aligned} X^{(k+1)} &= X^{(k+1/2)}A\, X^{(k+1/2)} \\ &= V \begin{bmatrix} \Sigma^{+} & E_{12} \\ E_{21} & 0 \end{bmatrix} U^T . \end{aligned}$$

Thus we deduce that the iteration (7.1) is stable for any matrix A. Similarly, we deduce that the iteration (7.4) is stable for any matrix A. Thus, the methods of Section 7 have the dual advantage of stability and a well-conditioned solution, in contrast to the use of Newton iteration (or its accelerated form) on A. (If one wishes to compute A^+, then the instability of Newton's method can be partly removed by using a few iterations (7.1) or (7.4) after all the significant singular values have converged. Some practical details of this technique are discussed in Section 10.)

9. Computing the Projection onto a Subspace Spanned by Singular Vectors

In this section we discuss a modification of the Newton iterations discussed above that allows us to compute the orthogonal projection matrices onto subspaces spanned by the singular vectors corresponding to either the dominant or the smallest singular values at less expense than computing the generalized inverse of A. Important applications to spectral estimation and direction finding with antenna arrays were developed by Schmidt [15].

Let us define the matrices

$$P(\varepsilon) = AA^{+}(\varepsilon) = U\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} U^{T} \tag{9.1}$$

and

$$P^{*}(\varepsilon) = A^{+}(\varepsilon)A = V\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} V^{T} \tag{9.2}$$

where the matrices U and V are from (2.1), I is the $r(\varepsilon)\times r(\varepsilon)$ identity block, and $r(\varepsilon)$ is the number of the singular values of A that are not less than ε. Then $P(\varepsilon)$ and $P^{*}(\varepsilon)$ are the orthogonal projections onto the subspaces spanned by the first $r(\varepsilon)$ columns of the matrices U and V, respectively. Our previous results already give us some iterative algorithms for computing $A(\varepsilon) = (A^{+}(\varepsilon))^{+} = AA(\varepsilon)^{+}A$, as well as $P(\varepsilon)$ and $P^{*}(\varepsilon)$, but there are simpler and more efficient algorithms that we shall give shortly. In addition to the signal processing application mentioned above, we may use this technique as an alternative to the methods of Section 7 for computing $A(\varepsilon)$, since

$$A(\varepsilon) = P(\varepsilon)A = AP^{*}(\varepsilon)\,. \tag{9.3}$$

The following iteration extends (7.4) and converges to $P(\varepsilon)$, unless ε is a singular value of A. Let

$$P_0 = \alpha_0 AA^{T} + \beta I\,, \tag{9.4}$$

where we choose $\alpha_0 > 0$, $\beta \geq 0$, to satisfy

$$\alpha_0\varepsilon^2 + \beta = 1/2; \quad \alpha_0\sigma_1^2 + \beta < \hat{\rho}_4 = 1.37 \cdots . \tag{9.5}$$

then iterate as follows:

$$P_{k+1} = (-2P_k + 3I)P_k^2 = (I - 2(P_k - I))P_k^2 \qquad k = 0, 1, \cdots \tag{9.6}$$

The convergence of P_k to $P(\varepsilon)$ immediately follows from the considerations of Section 7.

The iteration (9.4)-(9.6) converges to $P^*(\varepsilon)$ if we replace AA^T by A^TA in (9.4). Furthermore, we may compute $A(\varepsilon)$ by using (9.3), which is superior to the solution given in Section 7 because we now need to compute a single generalized inverse (rather than two). Also, each iteration step (9.6) only involves two matrix multiplications. And finally, if A is rectangular then one of these iterations is less expensive than (7.4) (for example) because it involves smaller symmetric matrices.

Remark 9.1. The stability analysis of Section 8 can be immediately extended to the iteration (9.6).

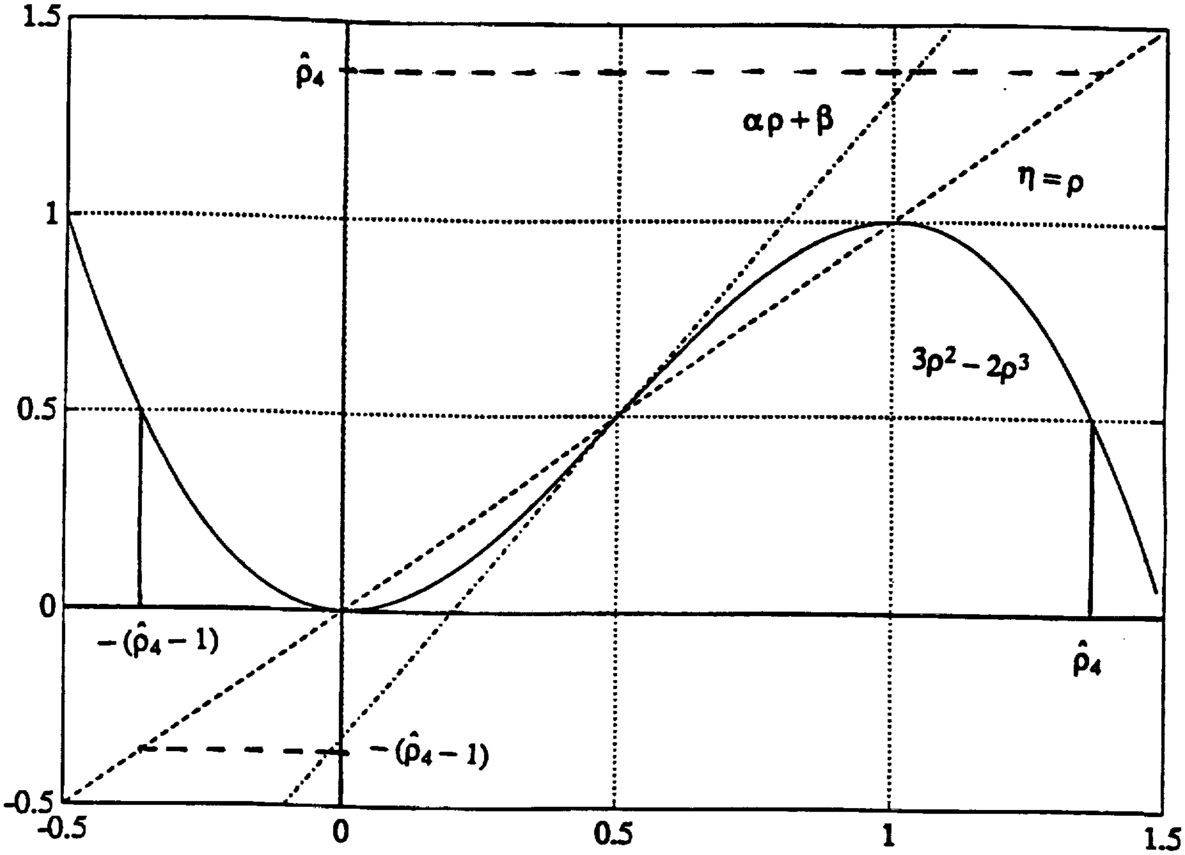


Figure 5. Acceleration by scaling and shifting, $P := \alpha P + \beta I$.

Remark 9.2. We may accelerate the cubic iteration for $P(\varepsilon)$ as follows. At the early stages of the iteration, it is more important to move singular values away from 0.5. After a step $\tilde{P} = 3P^2 - 2P^3$, we have that the spectrum of $\tilde{P}$ lies in the closed interval [0,1]. We then replace $\tilde{P}$ by $\alpha\tilde{P} + \beta I$ where $\alpha\rho + \beta$ is a line that maps [0,1] into $(-(\hat{\rho}_4 - 1), \hat{\rho}_4)$. To get the best possible speedup by this method, we choose such a line and also require that $\alpha(½) + \beta = ½$.

Remark 9.3. Our ability to compute the projectors $P(\varepsilon)$ allows us, with a little additional computation, to do the following:

(1) The projector $P(\varepsilon)$ defines the rank of the matrices $A(\varepsilon)$ and $A^+(\varepsilon)$, for

$$\text{rank } A(\varepsilon) = \text{rank } A^+(\varepsilon) = \| P(\varepsilon) \|_F = \text{trace}(P(\varepsilon)) .$$

This observation may be used as the basis of a bisection strategy for computing the singular values of A in polylog time. Indeed, the singular values of A are those of $A(\varepsilon)$ together with those of $A - A(\varepsilon)$. We may in this way reduce the problem to that of computing the positive singular value of a matrix with only one positive singular value. This we discuss in point (4). It is straightforward to develop a similar polylog algorithm for the eigenvalues of any symmetric matrix.

(2) We may compute projectors $P(\varepsilon_1,\varepsilon_2)$ onto subspaces spanned by singular vectors belonging to all the singular values in $[\varepsilon_1,\varepsilon_2)$, since $P(\varepsilon_1,\varepsilon_2) = P(\varepsilon_1) - P(\varepsilon_2)$.

(3) We may determine easily, for a given vector x, whether or not $x \in S(\varepsilon)$ where $S(\varepsilon)$ is the span of the singular vectors corresponding to singular values greater than or equal to ε, by checking whether or not $x = P(\varepsilon)x$.

(4) We may rapidly compute any singular value, regardless of multiplicity, as soon as we have found an interval $[\varepsilon_1,\varepsilon_2]$ that contains this singular value, σ, and no other. For σ is the only singular value of $A(\varepsilon_1,\varepsilon_2) = (P(\varepsilon_1) - P(\varepsilon_2))A$. Its multiplicity k is given by $\text{trace}(P(\varepsilon_1) - P(\varepsilon_2))$. And $\sigma^2 = \text{trace}(A^T(\varepsilon_1,\varepsilon_2)A(\varepsilon_1,\varepsilon_2)) / k$.

10. Experimental Results

We generated a random 64×64 matrix, then changed its singular values so as to create an ill-conditioned matrix A whose singular values lie in two clusters. There are 32 singular values in the interval [1, 7.6] and 32 others in the interval [10^{-7},10^{-6}].

iter	trace(XA)	$\lVert XA-I \rVert$	ρ
0	8.6801e-01	1.0000e+00	0
1	1.6882e+00	1.0000e+00	0
2	3.1987e+00	1.0000e+00	0
3	5.7784e+00	1.0000e+00	0
4	9.6436e+00	1.0000e+00	0
5	1.4398e+01	1.0000e+00	0
6	1.9113e+01	1.0000e+00	0
7	2.3340e+01	1.0000e+00	0
8	2.7108e+01	1.0000e+00	0
9	3.0015e+01	1.0000e+00	0
10	3.2111e+01	1.0000e+00	2.0481e-01
11	3.2003e+01	1.0000e+00	3.0331e-03
12	3.2014e+01	9.9998e-01	8.4572e-06
13	3.5992e+01	9.9357e-01	7.0355e-03
14	3.8974e+01	9.8718e-01	0
15	4.3075e+01	9.7452e-01	0
16	4.7663e+01	9.4970e-01	0
17	5.2046e+01	9.0192e-01	0
18	5.5954e+01	8.1346e-01	0
19	5.9225e+01	6.6172e-01	0
20	6.1749e+01	4.3787e-01	0
21	6.3309e+01	1.9173e-01	0
22	6.3906e+01	3.6761e-02	0
23	6.3997e+01	1.3514e-03	0
24	6.4000e+01	1.8267e-06	0
25	6.4000e+01	7.6424e-09	0

Table 1: Results for First Test Matrix

We used algorithm CUINV to compute A^{-1}. The initial iterate was that of (3.2) and (3.7). The results are shown in Figure 6; the data are in Table 1, which gives trace(X_kA), $\lVert X_kA - I \rVert_2$, and the computed bound ρ for $k=0, 1, \cdots ,12$. Where $\rho = 0$ the algorithm skipped the cubic acceleration step. Early on this is because δ is too large, as the first 32 large singular values converge. By way of comparison, Newton's method takes 60 iterations to obtain the solution produced by CUINV in 25.

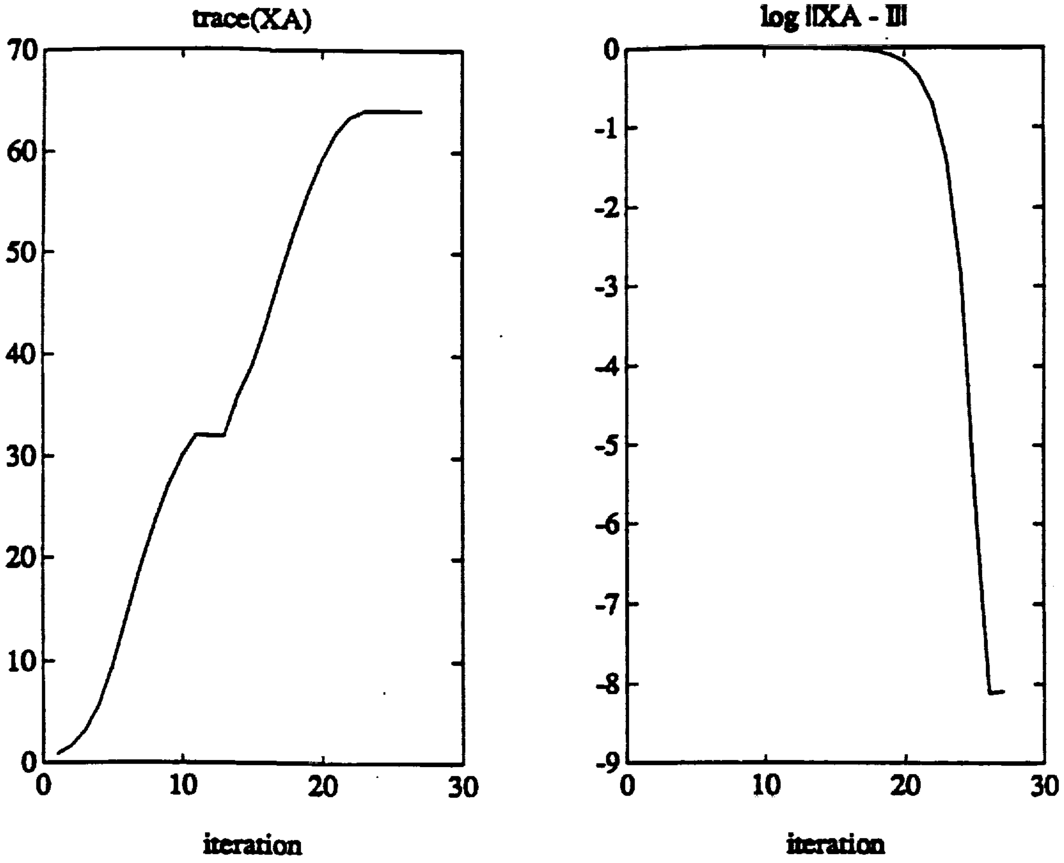


Figure 6. Convergence history for first test problem.

Next, we repeated this experiment with a second random matrix of order 64, having all its singular values in [0.066, 1]. The method chose unaccelerated Newton steps (19 are required) every time. Cubic steps were never used; this was due to the absence of any gap in spectrum of X_kA.

We therefore modified the algorithm so that when cubic acceleration is ruled out it uses a step of adaptive Tchebychev acceleration as discussed in Section 4. For the practical application of this idea, we require a means of finding a bound $\rho_*>0$ such that $\rho_r^{(k)}<\rho_*$. We then use as the acceleration parameter

$$\alpha_k = \frac{2}{1+(2-\rho_*)\rho_*} \,.$$

This insures that the acceleration process does not cause a large singular value to be mapped to the left of the smallest, thereby making the problem more difficult. Assume that the test $\delta < ¼$ at STEP 2 of CUINV fails. Let

$$\underline{\delta} = \delta / \sqrt{n} .$$

Then, if $\underline{\delta} < ¼$, we take

$$\rho_* = ½ - \sqrt{¼ - \underline{\delta}}$$

Now, since (5.1) holds,

$$\min_{1 \le j \le n} (\rho_j - \rho_j^2) \le \underline{\delta} .$$

Now it must be the case that ρ_r (the smallest positive singular value of $X_k A$) is less than ρ_* or else all of the singular values are in $(1 - \rho_*, 1]$. To rule out the latter possibility, we check whether

$$\text{trace}(X_k A) \ge n\,(1 - \rho_*)$$

and, if not, we accelerate.

With this change, the number of iterations required for convergence dropped to 13 for this problem; the operation count went from 20.5e6 to 14.6e6.

Evidently, even for well-conditioned matrices, modified CUINV can be far more efficient than Newton's method; for moderately ill-conditioned matrices, the differences are pronounced.

Next, we generated a random 64×64 matrix, then changed its singular values so as to create an ill-conditioned matrix A with 54 singular values in $[10^{-16}, 10^{-11}]$ and the remaining 10 in $[0.01, 1]$. We computed the generalized inverse of A(1.e-10), the matrix of rank 10 obtained by suppressing the small singular values of A. We first used algorithm CUINV with A, choosing $\alpha_0 = 1$. In addition we monitored the growth of $\bar{\varepsilon}$ which was initially set equal to $\varepsilon^2 = 10^{-20}$ and which is updated according to the recursions:

$$\bar{\varepsilon} := (2 - \bar{\varepsilon})\bar{\varepsilon}$$

in Step 1 and

$$\bar{\varepsilon} := \frac{1}{\underline{\rho}}(\bar{\varepsilon}^2 - (2 + \underline{\rho})\bar{\varepsilon} + (1 + 2\underline{\rho}))\,\bar{\varepsilon}$$

in Step 3. Thus, $\bar{\varepsilon}$ separates the singular values of XA that we wish to suppress from those that we wish to map to one. At the first iteration in which the parameter $\underline{\rho}$, computed in Step 3, is less than $\bar{\varepsilon}$, we stop and switch to the iteration (7.4) for, since (5.2) holds, the unwanted singular values of XA must now be smaller than ½. In this example, the switch occurred after the seventeenth iteration. Table 2 gives the results; see also Figure 7.

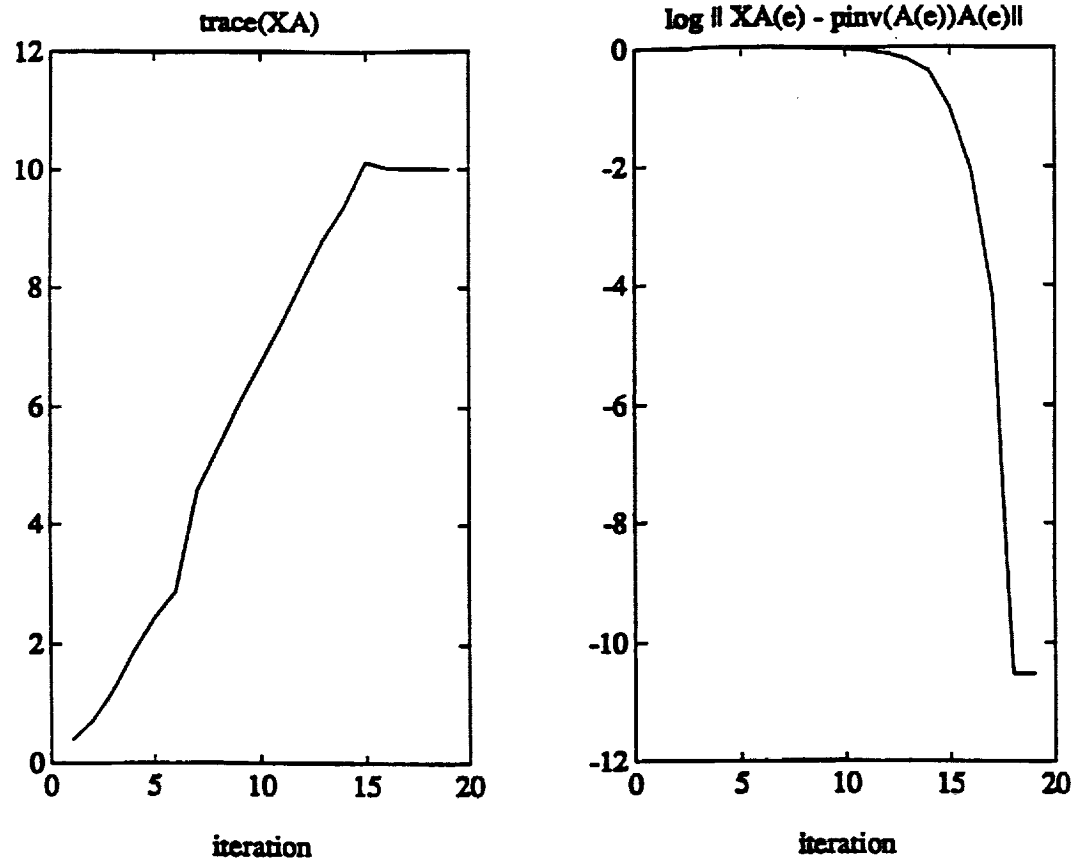


Figure 7. Convergence history for second test problem.

iter	trace(XA)	$\|\|XA(\varepsilon) - A(\varepsilon)^+A\|\|$	ε	ρ
1	3.9667e-01	9.9997e-01	1.0000e-20	0
2	7.2843e-01	9.9995e-01	2.0000e-20	0
3	1.2410e+00	9.9989e-01	4.0000e-20	0
4	1.8730e+00	9.9978e-01	8.0000e-20	0
5	2.4361e+00	9.9957e-01	1.6000e-19	0
6	2.8749e+00	9.9913e-01	3.2000e-19	0
7	4.5728e+00	9.9271e-01	2.6999e-18	4.5074e-01
8	5.3217e+00	9.8548e-01	5.3998e-18	0
9	6.0588e+00	9.7117e-01	1.0800e-17	0
10	6.7139e+00	9.4318e-01	2.1599e-17	0
11	7.3764e+00	8.8958e-01	4.3198e-17	0
12	8.1084e+00	7.9136e-01	8.6396e-17	0
13	8.7958e+00	6.2625e-01	1.7279e-16	0
14	9.3559e+00	3.9219e-01	3.4558e-16	0
15	1.0110e+01	9.2682e-02	5.3991e-15	1.7207e-01
16	1.0008e+01	8.4375e-03	1.2779e-12	8.5950e-03
17	1.0000e+01	7.1182e-05	3.5906e-08	7.1192e-05
18	1.0000e+01	3.0452e-11	...	...
19	1.0000e+01	3.0452e-11	...	...

Table 2: Computing $A(\varepsilon)^+$

The computed generalized inverse is accurate to about 11 digits. This is somewhat fewer than we could wish, given that the condition number of this problem was 100. The loss of accuracy is due to the fact that the accelerated Newton process (CUINV) went "too far", (raising $\bar{\varepsilon}$ by 12 orders of magnitude) before detecting the possibility of switching to the stable procedure (7.4).

11. Discussion

It has been our purpose here to clarify and illustrate the potential for the use of variants of Newton's method to solve problems of practical interest on highly parallel computers. We have shown how to accelerate the method substantially. We have shown how to modify it to successfully cope with ill-conditioned matrices. We have developed practical implementations. We conclude that Newton's method can be of value for some interesting computations, especially in parallel and other computing environments in which matrix products are especially easy to work with.

References

[1] A. Ben-Israel, "A Note on Iterative Method for Generalized Inversion of Matrices," *Math. Computation*, vol. 20, pp. 439-440, 1966.

[2] A. Ben-Israel and D. Cohen, "On Iterative Computation of Generalized Inverses and Associated Projections," *SIAM J. on Numerical Analysis*, vol. 3, pp. 410-419, 1966.

[3] R.P. Brent, F.T. Luk and C. Van Loan, "Computation of the Singular Value Decomposition Using Mesh-Connected Processors," *Journal of VLSI Computer Systems*, vol. 1, pp. 242-270, 1985.

[4] J. Dongarra, J. DuCroz, I. Duff, and S. Hammarling, "A Set of Level 3 Basic Linear Algebra Subprograms," ANL-MCS-P88-1, *Argonne National Laboratory*, Argonne, Illinois, 1988.

[5] D.K. Faddeev and V.N. Faddeeva, *Computational Methods of Linear Algebra*, W.H. Freeman, San Francisco, 1963.

[6] G.H. Golub and C.F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, Second edition, 1989.

[7] W. D. Hillis, *The Connection Machine*, MIT Press, Cambridge, Massachusetts, 1985.

[8] V. Pan, "Fast and Efficient Parallel Inversion of Toeplitz and Block Toeplitz Matrices," TR 88-8, *Computer Science Dept., SUNY Albany*, Albany, N.Y., 1988.

[9] V. Pan, "New Effective Methods for Computations with Structured Matrices," *Technical Report 88-28, Computer Science Dept., SUNY Albany*, 1988.

[10] V. Pan, "A New Acceleration of the Hilbert-Vandermonde Matrix Computations by Their Reduction to Hankel-Toeplitz Computations," Tech. Rep. TR

88-34, Computer Science Dept., SUNY Albany, Albany, NY, 1988.

[11] V. Pan and J. Reif, "Efficient Parallel Solution of Linear Systems," *Proc. 17-th Ann. ACM Symp. on Theory of Computing*, pp. 143-152, 1985.

[12] V. Pan and J. Reif, "Fast and Efficient Parallel Solution of Dense Linear Systems," to appear in *Computers & Math (with Applications)*, 1989.

[13] M.J. Quinn, *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill, New York, 1987.

[14] R. Schreiber , *Computing Generalized Inverses and Eigenvalues of Symmetric Matrices Using Systolic Arrays* , Computing Methods in Applied Science and Engineering, R. Glowinski and J.-L. Lions, editors, North Holland , Amsterdam, 1984 .

[15] R. O. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, pp. 276-280, 1986.

[16] G. Schultz, "Iterative Berechnung der Reziproken Matrix," *Z. Angew. Math. Mech.*, vol. 13, pp. 57-59, 1933.

[17] Torsten Soderstrom and G. W. Stewart, "On the Numerical Properties of an Iterative Method for Computing the Moore-Penrose Generalized Inverse," *SIAM J. Numer. Anal.*, vol. 11, pp. 61-74, 1974.