

Newton-like iteration for general and structured matrices

Gianni Codevico

Victor Y. Pan

Marc Van Barel

Xinmao Wang

Ai-Long Zheng

Report TW 372, November 2003

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Newton-like iteration for general and structured matrices

Gianni Codevico

Victor Y. Pan

Marc Van Barel

Xinmao Wang

Ai-Long Zheng

Report TW 372, November 2003

Department of Computer Science, K.U.Leuven

Abstract

We review and amend the known applications of Newton's iteration computing the inverse or Moore–Penrose generalized inverse of a matrix. Then we specialize this approach to the case of structured (and particularly Toeplitz or Toeplitz-like) matrices where all input, output and intermediate auxiliary matrices are represented in a compressed form, via their short displacement generators. We briefly recall the known policies of compression via the truncation of the smallest singular values of the displacement and via substitution and elaborate upon the very recent policy of the least-squares compression. Numerical experiments demonstrate the effectiveness of our new Newton-like iteration based on a cubic polynomial.

Keywords : inverse of a matrix, Moore-Penrose generalized inverse, Newton-like iteration, structured matrices, displacement structure.

AMS(MOS) Classification : Primary : 65F10, Secondary : 65F20.

Newton-like iteration for general and structured matrices ^{*}

Gianni Codevico [†], Victor Y. Pan [‡], Marc Van Barel [§],
Xinmao Wang [¶] and Ai-Long Zheng ^{||}

November 19, 2003

Abstract

We review and amend the known applications of Newton's iteration computing the inverse or Moore–Penrose generalized inverse of a matrix. Then we specialize this approach to the case of structured (and particularly Toeplitz or Toeplitz-like) matrices where all input,

^{*}The research of the second and fourth author was supported by NSF Grant CCR 9732206 and PSC CUNY Award 66406-0033. The research of the first and third author was supported by the Research Council K.U.Leuven, project OT/00/16 (SLAP: Structured Linear Algebra Package), by the Fund for Scientific Research–Flanders (Belgium), projects G.0078.01 (SMA: Structured Matrices and their Applications), G.0176.02 (AN-CILA: Asymptotic aNalysis of the Convergence behavior of Iterative methods in numerical Linear Algebra), and G.0184.02 (CORFU: Constructive study of Orthogonal Functions), and by the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture, project IUAP V-22 (Dynamical Systems and Control: Computation, Identification & Modelling). The scientific responsibility rests with the authors.

[†]Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven (Heverlee), Belgium (Gianni.Codevico@cs.kuleuven.ac.be).

[‡]Department of Mathematics and Computer Science, Lehman College of CUNY, Bronx, NY 10468, USA (vpan@lehman.cuny.edu).

[§]Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven (Heverlee), Belgium (Marc.VanBarel@cs.kuleuven.ac.be).

[¶]Ph.D. Program in Mathematics, Graduate School of CUNY, New York, NY 10016, USA (xwang2@gc.cuny.edu).

^{||}Ph.D. Program in Mathematics, Graduate School of CUNY, New York, NY 10016, USA (ailongz@yahoo.com).

output and intermediate auxiliary matrices are represented in a compressed form, via their short displacement generators. We briefly recall the known policies of compression via the truncation of the smallest singular values of the displacement and via substitution and elaborate upon the very recent policy of the least-squares compression. Numerical experiments demonstrate the effectiveness of our new Newton-like iteration based on a cubic polynomial.

short running title: Newton-like iteration

keywords: inverse of a matrix, Moore-Penrose generalized inverse, Newton-like iteration, structured matrices, displacement structure

AMS (MOS) classification: 65F10, 65F20

1 Introduction

Structured matrices are omnipresent in computations in sciences, engineering and signal and image processing (see some bibliography on these computations in [16], [19], [20]). Table 3.1 shows the four most popular classes of structured matrices but many more classes appear in practical computations. Computing the inverses and the Moore–Penrose generalized inverses of structured matrices are among the most important practical computational problems. Iterative methods for matrix inversion and Moore–Penrose generalized inversion such as Newton’s iteration have been long and well known [25], [2], [3], [26], [23] and are attractive because of very strong numerical stability, local quadratic convergence, and convenience for parallel implementation. Their application in the case of general input matrices was limited, however, because initially the convergence is generally quite slow for an ill-conditioned input (in spite of some acceleration techniques in [23]) and because each iteration step involves two expensive operations of matrix multiplication. The latter problem disappears when the matrices are structured [20], but another problem arises, that is, one must preserve the matrix structure performing the iteration. Some nontrivial techniques for this problem have been proposed and analyzed (see Chapter 6 in [20], [7] and the bibliography therein). In this paper we first survey the state of the art, then show some refinements of the known methods for general matrices, and finally unify and strengthen the cited nontrivial techniques for dealing with structured input. We propose a new iteration method based on a cubic poly-

nomial. Our numerical experiments show its efficiency and a nontrivial and still enigmatic effect of the improvement of approximation by compression in the case of structured input; we call this effect autocorrection.

2 Newton's iteration for general matrices

2.1 Unscaled iteration

Newton's iteration

$$X_{i+1} = 2X_i - X_i M X_i, \quad i = 0, 1, \dots, \quad (2.1)$$

converges to the inverse M^{-1} or the Moore–Penrose generalized inverse M^+ of a matrix M provided that all eigenvalues of the residual matrices $I - X_i M$ and/or $I - M X_i$ are less than 1. Write

$$\overline{R}_i = R_{left,i} = I - X_i M, \quad \underline{R}_i = R_{right,i} = I - M X_i$$

for all i , and denote both \overline{R}_i and \underline{R}_i by R_i wherever the discussed properties apply to both \overline{R}_i and \underline{R}_i . Equation (2.1) immediately implies that $R_i = R_{i-1}^2 = R_0^{2^i}$ for all i , and so the convergence in (2.1) is quadratic. Assume that the approximations X_i share singular vectors with M^T and write the SVD's as follows:

$$M = U \Sigma V^T, \quad X_i = V \Sigma_i U^T, \quad X_i M = V \Sigma_i \Sigma V^T,$$

where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$, $\Sigma_i \Sigma = \text{diag}(\rho_1^{(i)}, \dots, \rho_r^{(i)})$. Write

$$\overline{R}_i = V(I - \Sigma_i \Sigma)V^T.$$

Hence,

$$\overline{R}_i = \overline{R}_0^{2^i} = V(I - \Sigma_0 \Sigma)^{2^i} V^T.$$

Let us first assume that both σ_1 and σ_r are available (see Subsection 2.6). To make $\|R_0\|_2$ as small as possible for X_0 of the form $X_0 = a_0 M^T$, we choose

$$X_0 = a_0 M^T, \quad a_0 = 2/(\sigma_1^2 + \sigma_r^2).$$

Then $\rho_j^{(0)} = a_0 \sigma_j^2$ for all j ; therefore, the diagonal matrix $I - \Sigma_0 \Sigma$ has all diagonal entries $1 - \rho_j^{(0)}$ in the range $[-\nu, \nu]$ for

$$\nu = \|R_0\|_2 = \frac{\sigma_1^2 - \sigma_r^2}{\sigma_1^2 + \sigma_r^2} = \frac{\kappa^2 - 1}{\kappa^2 + 1}, \quad \kappa = \sigma_1/\sigma_r.$$

It follows that for any positive $\epsilon \leq 1/2$, we have $\|R_h\|_2 \leq 1/2$, $\|R_{h+i}\|_2 \leq \epsilon$ where $h = \lceil \ln((\kappa^2 + 1)/2) \rceil \leq \lceil 2 \ln \kappa \rceil$, $i = \lceil \log_2 \ln(1/\epsilon) \rceil$.

Remark 2.1. The computational cost of every step (2.1) slightly decreases in its variations

$$\begin{aligned} X_{i+1} &= X_i(2I - MX_i) = (2I - X_iM)X_i, \\ Y_{i+1} &= Y_i(2I + MY_i) = (2I + Y_iM)Y_i \end{aligned}$$

where Y_i converges to $-M^{-1}$ or $-M^\dagger$.

2.2 Scaled iteration

Let $c_j(x)$ denote the Chebyshev polynomial of degree j . To halve the number of iterations to obtain the desired output precision, i.e., to decrease the bound h of Subsection 2.1 to $\log_2 \kappa + O(1/\kappa^2)$, we can use a scaled version of (2.1),

$$X_{i+1} = a_{i+1}(2X_i - X_iMX_i), \quad i = 0, 1, \dots, \quad (2.2)$$

for appropriate scalars a_i such that $\bar{R}_i = I - X_iM = c_{2^i, \Gamma^2}(M^T M)$, where $c_{2^i, \Gamma^2}(x) = c_{2^i}(\gamma x + \sigma)/c_{2^i}(\sigma)$, for $\gamma = 2/(\sigma_1^2 - \sigma_r^2)$, $\sigma = -(\sigma_1^2 + \sigma_r^2)/(\sigma_1^2 - \sigma_r^2)$, is the scaled Chebyshev polynomial of degree 2^i on the interval $\Gamma^2 = [\sigma_r^2, \sigma_1^2]$. In [23, Section 4] the recurrence expressions for a_i have been specified as follows:

$$\begin{aligned} a_0 &= \frac{2}{\sigma_1^2 + \sigma_r^2}, \quad a_i = (1 + b_i)/b_i, \quad b_i = c_{2^i}(\sigma), \quad b_i = 2b_{i-1}^2 - 1, \\ i &= 1, 2, \dots, \quad b_0 = c_1(\sigma) = \sigma = -\frac{\sigma_1^2 + \sigma_r^2}{\sigma_1^2 - \sigma_r^2}. \end{aligned} \quad (2.3)$$

Therefore,

$$a_1 = 1 + \frac{1}{2\sigma^2 - 1}, \quad a_{i+1} = \frac{2}{1 + (2 - a_i)a_i}, \quad i = 1, 2, \dots \quad (2.4)$$

The minimax property of Chebyshev polynomials implies that this scaling minimizes the overall number of flops required to yield a fixed bound on the residual norm (see [23, Section 4]) unless we have some additional information about the singular values of M . In the next two subsections we exploit some information of this kind, by following [23, Sections 5 and 6].

2.3 Symmetric positive definite input

Let M be symmetric positive definite and let $c_{2^i, \Gamma}(x) = c_{2^i}(\bar{\gamma}x + \bar{\sigma})/c_{2^i}(\bar{\sigma})$ denote the scaled Chebyshev polynomial of degree 2^i on the interval $\Gamma = [\sigma_r, \sigma_1]$, where $\bar{\gamma} = 2/(\sigma_1 - \sigma_r)$, $\bar{\sigma} = -(\sigma_1 + \sigma_r)/(\sigma_1 - \sigma_r)$. Our analysis in the previous section can be extended with σ_i replacing σ_i^2 for $i = 1$ and $i = r$ and with $\Gamma = [\sigma_r, \sigma_1]$ replacing Γ^2 . We specify X_1 (instead of X_0) as follows,

$$X_1 = (8/\delta)(-M + (\sigma_1 + \sigma_r)I), \quad \delta = 4\sigma_1\sigma_r + (\sigma_1 + \sigma_r)^2.$$

Then $X_1 = p(M)$, where $1 - xp(x) = c_{2, \Gamma}(x)$. Under this choice,

$$\|R_1\|_2 = 1 - 8\kappa/(\kappa^2 + 6\kappa + 1)$$

is minimum over all X_1 linear in M . By extending the scaling policy (2.2)–(2.4), we recursively compute X_{i+1} in (2.2) for

$$a_i = \frac{1 + \bar{b}_i}{\bar{b}_i}, \quad \bar{b}_i = c_{2^i}(\bar{\sigma}) = 2\bar{b}_{i-1}^2 - 1, \quad \bar{b}_1 = \bar{\sigma} = -\frac{\sigma_1 + \sigma_r}{\sigma_1 - \sigma_r}, \quad i = 2, 3, \dots$$

Then $R_i = c_{2^i, \Gamma}(M)$, for all i , and again the extremal property of Chebyshev polynomials implies that the overall number of flops required to yield a fixed upper bound on the singular values of R_i is minimized over all scaled processes (2.2), all symmetric positive definite input matrices M , and all choices of X_1 linear in M . The maximum values of $|c_{2^{i+1}, \Gamma}(x)|$ on Γ are less than the maximum values of $|c_{2^i, \Gamma^2}(x)|$ on Γ^2 . Therefore, the residual norm now decreases faster, and roughly by twice fewer iteration steps are required to ensure the same residual norm bound.

2.4 Ill-conditioned input with a gap for the eigenvalues

Suppose that iteration (2.1) computes X_i such that the set of all eigenvalues $\rho_1^{(i)}, \dots, \rho_r^{(i)}$ consists of two clusters about 0 and 1. This phenomenon typically occurs for many ill-conditioned matrices M and moderate k and can be detected by observing that

$$\delta = \|X_i M - (X_i M)^2\|_F \leq 1/4, \quad (2.5)$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm. Write $\rho_- = 1/2 - \sqrt{1/4 - \delta}$, $0 \leq \rho_- < 1/2$, $1 + \rho_+ = 1/2 + \sqrt{1/4 + \delta}$. As soon as (2.5) is observed, we know

that all the eigenvalues $\rho_j^{(i)}$ lie in the two intervals: $[0, \rho_-]$ and $[1 - \rho_-, 1 + \rho_+]$. In this case the acceleration of the convergence can be achieved by computing the matrices

$$X_{i+1} = b_i(X_i M)^2 X_i + c_i X_i M X_i + d_i X_i \quad (2.6)$$

where $b_i = 1/\rho_-$, $c_i = -(2 + \rho_-)/\rho_-$ and $d_i = (1 + 2\rho_-)/\rho_-$. The polynomial $p(y) = b_i y^3 + c_i y^2 + d_i y$ satisfies the relations $p(1) = 1$, $p'(1) = 0$ (which should make the value 1 an attraction point for the larger eigenvalues $\rho_j^{(i)}$), $p(0) = 0$, $p'(0) = 2 + 1/\rho_- > 4$ is large which pushes the smaller eigenvalues $\rho_j^{(i)}$ towards 1, $0 \leq p(x) \leq 1$ for $0 \leq x \leq \rho_-$, $1 \leq p(x) \leq 1 + \rho_-$ for $1 - \rho_- \leq x \leq 1 + \rho_+$ and $p(\rho_-) = 1$. Subsequent steps either repeat (2.6) if (2.5) holds or invoke (2.1) otherwise. For more details, we refer the interested reader to [23, Section 5].

Remark 2.2. One may ask how much more could we push the smaller eigenvalues $\rho_j^{(i)}$ towards 1 by choosing an appropriate polynomial $p(y)$. Quantitatively, this is measured by the magnitude of $p'(0)$, and a limitation comes from the upper bound $p'(0) \leq d^2$, which holds for any polynomial $p(y)$ of degree d satisfying $0 \leq p(y) \leq 1$ for $0 \leq y \leq 1$ and $p(0) = 0$ (cf. [22, Part VI, No.83]); this bound is achieved in the case of the shifted Chebyshev polynomials $p(y) = (1 - c_d(1 - 2y))/2$. They do not satisfy the important requirement $p'(1) = 0$, but this is not critical if the application purpose is just to move the eigenvalues deeper into the interval of convergence, away from its endpoints 0 and 1 (or 0 and 2) and then to shift to steps (2.1). For instance, shifting to the range $[0, 2]$, we may use the polynomial $p(x) = 1 - c_3(x - 1) = 4x^3 - 12x^2 + 9x$, such that $p'(0) = p'(2) = 9$, $0 \leq p(x) \leq 2$ when $0 \leq x \leq 2$.

2.5 Homotopic (continuation) approach

Combining the iteration processes (2.1) and (2.2) with the homotopic (continuation) approach simplifies the choice of the initial approximations and enables us to accelerate the computation in the case where M is a symmetric indefinite matrix [20, Section 6.9], [21].

2.6 How closely should we approximate the extremal singular values?

Some policies of initializing and scaling involve the singular values σ_1 and σ_r of M . The largest singular value σ_1 can be effectively approximated by

using the power method or (better) the Lanczos method [10]. Then the same methods can be applied to approximate the largest eigenvalue $\nu - \sigma_r^2$ of the symmetric positive definite matrix $\nu I - M^T M$ for $\nu > \sigma_1^2$, and so the value σ_r can be easily approximated. The approximation error is proportional to $\nu - \sigma_r^2$ and thus contaminates the output value σ_r if σ_1/σ_r is large, that is, if the matrix M is ill-conditioned.

Fortunately, we yield almost the same upper bounds on $\|\bar{R}_i\|_2$ and $\|R_i\|_2$ for quite a few first steps i by relying on (2.2) and (2.5) or (2.6) and (2.8) where σ_1 and σ_r are replaced by their upper estimates σ_1^+ and σ_r^+ within the absolute error bound $\epsilon\sigma_1$ for a small positive ϵ of the order of the machine precision.

Finally, even without approximating σ_1 and σ_r , we still may apply the processes (2.1) and (2.6) (but not (2.2)) using other choices for X_0 . In particular, we may easily compute $X_0 = M^T/(\|M\|_1\|M^T\|_1)$ yielding the bound $\|R_0\|_2 \leq 1 - 1/(n\kappa^2)$ for an $n \times n$ matrix M . For an $n \times n$ symmetric positive definite matrix M , we may yield the bound $\|R_0\|_2 \leq 1 - 1/(n^{1/2}\kappa)$ by choosing $X_0 = I/\|M\|_F$.

3 Newton's iteration for structured matrices

3.1 Structured matrices

(Scaled) Newton's iteration and other iterative inversion methods are most effective for structured matrices, for which matrix-by-matrix and matrix-by-vector multiplication can be performed at a low computational cost (using $O(n \log n)$ or $O(n \log^2 n)$ flops, versus the order of n^3 for $n \times n$ general matrices).

Recall the most popular classes of structured matrices displayed in Table 3.1. Each of these $n \times n$ matrices is completely defined by $n, 2n - 1$, or $2n$ parameters. More generally, many other matrices with similar structures can be represented with $O(n)$ parameters as follows. Associate with a fixed class of structured matrices M a pair of *operator matrices* A and B such that the *Stein* and/or *Sylvester displacements* of M ,

$$\Delta_{A,B}(M) = M - AMB = GH^T, \quad \nabla_{A,B}(M) = AM - MB = GH^T, \quad (3.1)$$

respectively, have small rank r (called the *displacement rank* of M). The $n \times n$ matrix M can be effectively expressed via the columns of the matrices G and

Table 3.1: **Four classes of structured matrices**

Toeplitz matrices $(t_{i-j})_{i,j=0}^{n-1}$ $\begin{pmatrix} t_0 & t_{-1} & \cdots & t_{1-n} \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & t_1 & t_0 \end{pmatrix}$	Hankel matrices $(h_{i+j})_{i,j=0}^{n-1}$ $\begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & \ddots & h_n \\ \vdots & \ddots & \ddots & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{pmatrix}$
Vandermonde matrices $(t_i^j)_{i,j=0}^{n-1}$ $\begin{pmatrix} 1 & t_0 & \cdots & t_0^{n-1} \\ 1 & t_1 & \cdots & t_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_{n-1} & \cdots & t_{n-1}^{n-1} \end{pmatrix}$	Cauchy matrices $\left(\frac{1}{s_i-t_j}\right)_{i,j=0}^{n-1}$ $\begin{pmatrix} \frac{1}{s_0-t_0} & \cdots & \frac{1}{s_0-t_{n-1}} \\ \frac{1}{s_1-t_0} & \cdots & \frac{1}{s_1-t_{n-1}} \\ \vdots & \ddots & \vdots \\ \frac{1}{s_{n-1}-t_0} & \cdots & \frac{1}{s_{n-1}-t_{n-1}} \end{pmatrix}$

H of size $n \times r$. We say that the pair (G, H) is a (nonunique) *displacement generator* of length r for M . One may operate with structured matrices represented in such a compressed form. We trace this important approach back to [17], [13], [14], [12] (cf. also [9], [18]); it has a huge bibliography (cf. [16], [20], [15], [6] for surveys and details). In this paper, we cover the Sylvester displacement representation referring the reader to [20] and the references therein on the dual Stein displacement representation.

Table 3.2: **Structured matrix classes and the operator matrices**

Matrix Class	Pair of Operator Matrices	Displacement Rank	# of Flops for Multiplication by Vector
Toeplitz $(t_{i-j})_{i,j}$	$Z_e, Z_f, e \neq f$	≤ 2	$O(n \log n)$
Hankel $(h_{i+j})_{i,j}$	$Z_e, Z_f^T, ef \neq 1$	≤ 2	$O(n \log n)$
Vandermonde $(t_i^j)_{i,j}$	D_t, Z_f^T	≤ 1	$O(n \log^2 n)$
Cauchy $\left(\frac{1}{s_i-t_j}\right)_{i,j}$	D_s, D_t	≤ 1	$O(n \log^2 n)$

Typical operator matrices are the unit f -circulant matrices,

$$Z_f = (z_{i,j})_{i,j=0}^{n-1} \quad (3.2)$$

where $z_{i+1,i} = 1$, $i = 0, 1, \dots, n-2$, $z_{0,n-1} = f$, $z_{i,j} = 0$ if $(i-j) \bmod n \neq 1$, and diagonal matrices, $D_{\mathbf{s}} = \text{diag}(s_i)_{i=0}^{n-1}$ for $\mathbf{s} = (s_i)_{i=0}^{n-1}$. Table 3.2 shows the operator matrices associated with some classes of structured matrices, the displacement rank, and the cost in flops for multiplication by a vector. For these operator matrices, the arithmetic cost of multiplication of a structured matrix by a vector lies in $O(rn \log n)$ with the operators ∇_{Z_e, Z_f} and ∇_{Z_e, Z_f^T} (that is, in the *Toeplitz/Hankel*, *T/H*, case) and in $O(rn \log^2 n)$ with the operators $\nabla_{D(t), Z_f^T}$ and $\nabla_{D(s), D(t)}$ (that is, in the *Vandermonde/Cauchy*, *V/C*, case). One may operate with structured matrices given in compressed form by their displacement generators according to some simple rules (see, e.g., [20, Section 1.5]) and then if needed recover the output from its displacement generator based on the formulas in [20, Sections 4.4, 4.5] and [24].

3.2 Compressed Newton's iteration for structured matrices

The acceleration of iterative inversion of structured matrices is achieved by reducing processes (2.1), (2.2) to structured matrix-by-vector multiplication. Similarly to (3.1), write

$$\nabla_{B,A}(X_i) = G_i H_i^T \quad (3.3)$$

where G_i, H_i are $n \times l$ matrices, $r \leq l = l(i) \leq n$, and observe that iteration (2.2) can be performed by computing the generators

$$G_{i+1} = a_{i+1} \left(-X_i G, (I - X_i M) G_i, G_i \right), \quad (3.4)$$

$$H_{i+1} = \left(X_i^T H, H_i, (I - M X_i)^T H_i \right) \quad (3.5)$$

for X_{i+1} . For $a_{i+1} = 1$, this turns into a compressed version of iteration (2.1). Equations (3.4) and (3.5) reduce step (2.2) or (2.1) to multiplication of the matrices M, M^T, X_i , and X_i^T by $l, l, l+r$, and $l+r$ vectors, respectively, where $l = l(i)$ is the length of the displacement generator for X_i . This means

$$c_{l,n,r} = O((l+r)^2 n \log^d n) \quad (3.6)$$

flops per step (2.2) where $d = 1$ (in the T/H case) or $d = 2$ (in the V/C case), and so it is crucial to bound $l = l(i)$ to make the iteration effective. Typical initial choices of X_0 achieve $l_0 \leq r$, but the compression steps (3.4), (3.5) may inflate this to $l_i = 3^i l_0$. Therefore special care should be taken periodically to keep computations effective.

To compress the iterates X_{i+1} one should modify (2.1), (2.2), and (3.3)–(3.5) as follows:

$$\hat{X}_{i+1} = g(X_i, M), \quad X_{i+1} = f(\hat{X}_{i+1}) \quad (3.7)$$

where $g(X_i, M)$ is the iteration defined by (2.1) or (2.2), and the function $f(W)$ defines a compression rule.

Two examples of compression rules are given by

1. truncation of the smallest singular values of the displacement $\nabla_i = \nabla_{B,A}(\hat{X}_i)$ such that $X_i = \nabla_{B,A}^{-1}(\text{tr}\nabla_i)$, that is, X_i is computed by applying the inverse of the displacement operator $\nabla_{B,A}$ to $\text{tr}(\nabla_i)$, the truncated displacement ∇_i having a fixed rank (say, r , $2r$, or $3r$), and
2. substitution, in which case X_{i+1} is represented by its displacement generator G_{i+1}, H_{i+1} where

$$\begin{aligned} G_{i+1} &= -\hat{X}_{i+1}G \quad , \quad H_{i+1} = \hat{X}_{i+1}^T H, \\ \nabla_{A,B}(M) &= GH^T \quad , \quad \nabla_{B,A}(X_{i+1}) = G_{i+1}H_{i+1}^T. \end{aligned}$$

Under both policies, compression generally perturbs the computed approximate inverse \hat{X}_{i+1} towards or outwards M^{-1} . Assuming the worst, that is, the perturbation outwards, we still may count on rapid convergence if the residual norm remains small enough so that its subsequent squaring in step (2.1) overcompensates us for the perturbation affect. Upper bounds on the perturbation distance and the worst case affect on coverage have been estimated in many papers, most recently in [21] (see [20, Chapter 6] and the bibliography therein on other works).

Extensive experiments with Toeplitz matrices M reported in [20, Section 6.11] and [21] show that the perturbation actually tends to preserve the convergence stronger than the worst case theoretical study suggests, that is, the perturbation frequently pushes the approximate inverses towards the inverse. In particular, in about 25% of tests, the Newton steps (2.1) with compression converged where the residual norms at times exceeded 1 and sometimes substantially (see Table 6.21 in [20]). Theoretical explanation of

this phenomenon is still a challenge (see, however, the end of Section 4). We call this effect autocorrection of the compression because with no compression the residuals and therefore their norms are squared in each iteration step which excludes this effect in absence of compression.

A more minor negative affect of the compression persists in both theory and our experiments with Toeplitz matrices M . That is, already with the first compression step we lose the theorems in [23] on the acceleration by twice using scaling (2.3). Thus scaling can help only in a few steps before the first compression according to both theory and the experiments.

3.3 Simplified compression in the Toeplitz case

General policies of compression can be made more efficient for specific classes of structured matrices. Here is an example where the compression via substitution is simplified in the case of Toeplitz matrices M (the number of multiplications by vectors decreases by twice versus the general policy). Define $Z_f(\mathbf{v}) = \sum_{i=0}^{n-1} v_i Z_f^i$, the f -circulant matrix with the first column $\mathbf{v} = (v_i)_{i=0}^{n-1}$, for Z_f as in (3.2). For a fixed nonsingular Toeplitz matrix $M = (t_{i-j})_{i,j=0}^{n-1}$ and three constants, $b \neq 0$, a and t , define the vector $\mathbf{t} = (t, at_1 - bt_{1-n}, \dots, at_{n-1} - bt_{-1})$. Let $\mathbf{e}_0 = (\delta_{i0})_{i=0}^{n-1}$ be the first coordinate vector, $\delta_{00} = 1$, $\delta_{i0} = 0$ for $i > 0$. Write $P = [\mathbf{e}_0, \mathbf{t}]$.

Theorem 3.1. [11] *Let $M\mathbf{x} = \mathbf{e}_0$, $M\mathbf{y} = \mathbf{t}$. Let $bd \neq 1$. Then*

$$(bd - 1)M^{-1} = Z_d(\mathbf{x})Z_{1/b}(\mathbf{y}) - Z_d(\mathbf{y} - (1 - bd)\mathbf{e}_0)Z_{1/b}(\mathbf{x}).$$

Define Newton's approximation X_i by extending the latter equation as follows:

$$(bd - 1)X_i = Z_d(\mathbf{x}_i)Z_{1/b}(\mathbf{y}_i) - Z_d(\mathbf{y}_i - (1 - bd)\mathbf{e}_0)Z_{1/b}(\mathbf{x}_i),$$

for certain vectors \mathbf{x}_i and \mathbf{y}_i .

Theorem 3.2. *Let $|b| \geq 1 \geq |d|$. Then $\|X_i - M^{-1}\|_h \leq |bd - 1|^{-1}(\|\mathbf{x} - \mathbf{x}_i\|_1(|1 - bd| + 2\|\mathbf{y}_i\|_1) + 2\|\mathbf{y}_i - \mathbf{y}\|_1\|\mathbf{x}\|_1)$ for $h = 1$ and $h = \infty$.*

Proof. First recall that $Z_f(\mathbf{u}) - Z_f(\mathbf{v}) = Z_f(\mathbf{u} - \mathbf{v})$, for all $f, \mathbf{u}, \mathbf{v}$. Subtract the above expression for $(bd - 1)M^{-1}$ from that for $(bd - 1)X_i$ and deduce

that

$$\begin{aligned}
(bd - 1)(X_i - M^{-1}) &= Z_d(\mathbf{x}_i)Z_{1/b}(\mathbf{y}_i) - Z_d(\mathbf{x})Z_{1/b}(\mathbf{y}) \\
&+ Z_d(\mathbf{y} - (1 - bd)\mathbf{e}_0)Z_{1/b}(\mathbf{x}) - Z_d(\mathbf{y}_i - (1 - bd)\mathbf{e}_0)Z_{1/b}(\mathbf{x}_i) \\
&= Z_d(\mathbf{x}_i - \mathbf{x})Z_{1/b}(\mathbf{y}_i) + Z_d(\mathbf{x})Z_{1/b}(\mathbf{y}_i - \mathbf{y}) \\
&+ Z_d(\mathbf{y} - \mathbf{y}_i)Z_{1/b}(\mathbf{x}) + Z_d(\mathbf{y}_i - (1 - bd)\mathbf{e}_0)Z_{1/b}(\mathbf{x} - \mathbf{x}_i).
\end{aligned}$$

Now recall that $\|Z_f(\mathbf{u})\|_h \leq \|\mathbf{u}\|_1$ for $|f| \leq 1$ and any \mathbf{u} where $h = 1$ or $h = \infty$, and from this, the result of the theorem can be deduced. \square

Corollary 3.3. $X_i \rightarrow M^{-1}$ as $\mathbf{x}_i \rightarrow \mathbf{x}$, $\mathbf{y}_i \rightarrow \mathbf{y}$.

Trying to accelerate the convergence, we may compute a least-squares solution to the matrix equation $M[\mathbf{x}_i, \mathbf{y}_i] = [\mathbf{e}_0, \mathbf{t}]$; that is, we may replace the vector pair $[\mathbf{x}_i, \mathbf{y}_i]$ by the pair $[\mathbf{u}_i, \mathbf{v}_i] = (S_i^T S_i)^{-1} S_i^T [\mathbf{e}_0, \mathbf{t}]$, $S_i = M[\mathbf{x}_i, \mathbf{y}_i]$, which minimizes the norm $N_i = \|M[\mathbf{x}_i, \mathbf{y}_i]B + [\mathbf{e}_0, \mathbf{t}]\|_2$ over all pairs $[\mathbf{u}_i, \mathbf{v}_i] = [\mathbf{x}_i, \mathbf{y}_i]B$ and 2×2 matrices B . This is an example of the general techniques of least-squares improvement of approximations developed in [7] and in our next subsection.

3.4 Least-squares compression/refinement

The third policy of compression is to compute a least-squares refinement G_{i+1}, H_{i+1} of the displacement generator $\hat{G}_{i+1}, \hat{H}_{i+1}$ of the computed approximation \hat{X}_{i+1} to M^{-1} such that

$$\nabla_{B,A}(\hat{X}_{i+1}) = \hat{G}_{i+1} \hat{H}_{i+1}^T, \quad (3.8)$$

$$G_{i+1} = \hat{G}_{i+1} Y_{i+1,G}, \quad H_{i+1} = \hat{H}_{i+1} Y_{i+1,H}, \quad (3.9)$$

and the norms

$$r_{i+1,G} = \|G + M \hat{G}_{i+1} Y_{i+1,G}\|_2 \quad (3.10)$$

and

$$r_{i+1,H} = \|H - M^T \hat{H}_{i+1} Y_{i+1,H}\|_2 \quad (3.11)$$

are minimum over all $l \times r$ matrices $Y_{i+1,G}$ and $Y_{i+1,H}$. The pair G_{i+1}, H_{i+1} is used as a displacement generator representing the matrix

$$X_{i+1} = \nabla_{B,A}^{-1}(G_{i+1} H_{i+1}^T) \quad (3.12)$$

(cf. (3.3)). (For motivation, compare (3.10) and (3.11) with the following equations implied by (3.1): $\nabla_{B,A}(M^{-1}) = G_- H_-^T = -M^{-1} \nabla_{A,B}(M) M^{-1} = -M^{-1} G H^T M^{-1}$, $G = -M G_-$, $H = M^T H_-$.) Clearly, policy (3.8)–(3.12) compresses the generator $(\hat{G}_{i+1}, \hat{H}_{i+1})$ to yield the same length as we have for (G, H) . Besides compression, this policy also intends to refine the approximation \hat{X}_{i+1} to M^{-1} , as follows from the next three theorems and a corollary; the first two theorems are easily deduced based on combining (3.8)–(3.12).

Theorem 3.4.

$$\begin{aligned} M \nabla_{B,A}(X_{i+1} - M^{-1})M &= G H^T + M G_{i+1} H_{i+1}^T M \\ &= (G + M G_{i+1}) H^T - M G_{i+1} (H^T - H_{i+1}^T M) \\ &= G (H^T - H_{i+1}^T M) + (G + M G_{i+1}) H_{i+1}^T M \\ &= G (H^T - H_{i+1}^T M) + (G + M G_{i+1}) H^T - (G + M G_{i+1}) (H^T - H_{i+1}^T M). \end{aligned}$$

Corollary 3.5. Write

$$r_{i+1,G,H} = \|G H^T + M G_{i+1} H_{i+1}^T M\|_2. \quad (3.13)$$

Then

$$\begin{aligned} r_{i+1,G,H} &\leq r_{i+1,G} \|H\|_2 + r_{i+1,H} \|M G_{i+1}\|_2, \\ r_{i+1,G,H} &\leq r_{i+1,G} \|H_{i+1}^T M\|_2 + r_{i+1,H} \|G\|_2, \\ r_{i+1,G,H} &\leq r_{i+1,G} \|H\|_2 + r_{i+1,H} \|G\|_2 + r_{i+1,G} r_{i+1,H}. \end{aligned}$$

The corollary bounds the norms $r_{i+1,G,H}$ via $r_{i+1,G}$ and $r_{i+1,H}$. The next theorem bounds the error and residual norms of approximation to M^{-1} via $r_{i+1,G,H}$.

Theorem 3.6. Assume $r_{i+1,G,H}$ as in Corollary 3.5 and write $\|\nabla_{B,A}^{-1}\|_{2r} = \sup_{\text{rank } Y \leq 2r} \|\nabla_{B,A}^{-1}(Y)\|_2 / \|Y\|_2$. Let $\text{rank } \nabla_{B,A}(X_{i+1}) \leq r$, $\text{rank } \nabla_{A,B}(M) \leq r$. Then we have

$$\begin{aligned} e_{i+1} &= \|X_{i+1} - M^{-1}\|_2 \leq r_{i+1,G,H} \|M^{-1}\|_2^2 \|\nabla_{B,A}^{-1}\|_{2r}, \\ r_{i+1,left} &= \|I - X_{i+1} M\|_2 \leq e_{i+1} \|M\|_2, \\ r_{i+1,right} &= \|I - M X_{i+1}\|_2 \leq e_{i+1} \|M\|_2. \end{aligned}$$

Remark 3.7. In [20, Section 6.6] and [24] quite tight upper and lower bounds on $\|\nabla_{B,A}^{-1}\|_{2r}$ are shown for various common pairs of the operator matrices A and B . In particular by Corollary 8.10 in [24] we have

$$\|\nabla_{Z_e, Z_f}^{-1}\|_{2r} \leq \sqrt{2r} \frac{|\tilde{e}\tilde{f}|^{\frac{n-1}{n}}}{\min_{i,j} |e^{\frac{1}{n}}\omega^i - f^{\frac{1}{n}}\omega^j|}$$

provided that $\tilde{e} = \max\{|e|, 1/|e|\}$, $\tilde{f} = \max\{|f|, 1/|f|\}$, $\omega = \exp(2\pi\sqrt{-1}/n)$.

Theorem 3.8. *The assignment $Y_{i+1} = Y_{i+1,G}Y_{i+1,H}^T$ minimizes the norm*

$$\hat{r}_{i+1,G,H} = \|GH^T + M\hat{G}_{i+1}Y_{i+1}\hat{H}_{i+1}^T M\|_2$$

over all $l \times l$ matrices Y_{i+1} , so $\hat{r}_{i+1,G,H} = r_{i+1,G,H}$ of Corollary 3.5.

Proof. For a real matrix X , let $X = U(\Sigma_0)V$ be the singular value decomposition of X , and let $X^+ = V^T(\Sigma_0^{-1})U^T$ be the Moore–Penrose generalized inverse of X . Let $M\hat{G}_{i+1} = U_1(\Sigma_1^0)V_1$ and $\hat{H}_{i+1}^T M = U_2(\Sigma_2^0)V_2$ be the SVD's. Then $\|GH^T + M\hat{G}_{i+1}Y_{i+1}\hat{H}_{i+1}^T M\|_2 = \|GH^T + U_1(\Sigma_1^0)V_1Y_{i+1}U_2(\Sigma_2^0)V_2\|_2 = \|U_1^T GH^T V_1^T + (\Sigma_1^0)V_1Y_{i+1}U_2(\Sigma_2^0)\|_2$ is minimum when $(\Sigma_1^0)V_1Y_{i+1}U_2(\Sigma_2^0) = -(\begin{smallmatrix} I & \\ & 0 \end{smallmatrix})U_1^T GH^T V_1^T (\begin{smallmatrix} I & \\ & 0 \end{smallmatrix})$. This holds for $Y_{i+1} = -(M\hat{G}_{i+1})^+ GH^T (\hat{H}_{i+1}^T M)^+$, that is, for $Y_{i+1} = Y_{i+1,G}Y_{i+1,H}^T$ where $Y_{i+1,G} = -(M\hat{G}_{i+1})^+ G$ and $Y_{i+1,H} = (M^T \hat{H}_{i+1})^+ H$ minimize the 2-norms $r_{i+1,G} = \|G + M\hat{G}_{i+1}Y_{i+1,G}\|_2$ and $r_{i+1,H} = \|H - M^T \hat{H}_{i+1}Y_{i+1,H}\|_2$, respectively. \square

Theorem 3.8 shows that the policy (3.8)–(3.12) of *compression/refinement via least-squares approximation* extends

- (1) the policy of “the truncation of singular values”, in which case $Y_{i+1} = (\begin{smallmatrix} I & \\ & 0 \end{smallmatrix})$, $\hat{G}_{i+1} = U(\sqrt{\Sigma} \ 0)$, $\hat{H}_{i+1}^T = (\sqrt{\Sigma} \ 0)V$, $U(\Sigma_0)V$ is the SVD of $\nabla_{B,A}(\hat{X}_{i+1})$, and
- (2) the policy of “compression via substitution”, in which case $Y_{i+1} = I$, $G_{i+1} = -g(X_i, M)G$, $H_{i+1}^T = H^T g(X_i, M)$, $\nabla_{B,A}(X_{i+1}) = (G_{i+1}H_{i+1}^T)$ for $g(X_i, M)$ as in (3.7).

In the case (1) where we replace compression via the truncation of singular values by the compression via least-squares approximation, \hat{G}_{i+1} and \hat{H}_{i+1} are

$n \times l$ matrices shown on the right-hand sides of (3.4) and (3.5) for $l \leq 3r$. They are typically rank deficient, and so the least-squares computation of $Y_{i+1,G}$ and $Y_{i+1,H}$ should rely on computing the SVD's of $M\hat{G}_{i+1}$ and $M^T\hat{H}_{i+1}$ at the cost of performing $2(2n + 11l)l^2$ flops (see [10, pages 257,263]) or maybe on rank revealing QR factorization replacing the SVD's. This clearly dominates the $(4l - 2)nr$ flops required for computing the generator (G_{i+1}, H_{i+1}) but typically (for smaller r) is dominated by $c_{l,n,r}$ flops in (3.6) involved in the computations in (3.4) and (3.5).

In the case (2), we may complement the compression via substitution by the refinement via least-squares approximation. Let us change the notation and write $\hat{G}_{i+1} = -g(X_i, M)G$ and $\hat{H}_{i+1} = H^T g(X_i, M)$. Then \hat{G}_{i+1} and \hat{H}_{i+1} are $n \times r$ matrices and typically have full rank. In this case, instead of computing $Y_{i+1,G}$ and $Y_{i+1,H}$ based on the SVD's, one may compute $Y_{i+1,G}$ and $Y_{i+1,H}$ simply from the normal equations

$$(\hat{G}_{i+1}^T M^T) M \hat{G}_{i+1} Y_{i+1,G} = \hat{G}_{i+1}^T M^T G, \quad (3.14)$$

$$(\hat{H}_{i+1}^T M) M^T \hat{H}_{i+1} Y_{i+1,H} = \hat{H}_{i+1}^T M H. \quad (3.15)$$

This amounts to multiplication of each of the matrices M and M^T by r vectors, that is a fraction of $c_{l,n,r}$ flops of (3.6) plus $4lr(2n - 1)$ flops, for computing the coefficients of normal equations (3.14), (3.15), and $(4l^3/3 + 3l^2)r$ flops, for solving these equations. Having $Y_{i+1,G}$ and $Y_{i+1,H}$, one may immediately compute G_{i+1} and H_{i+1} in (3.9).

Although in a certain sense, the policies (1) and (2) are two special cases, each of them may in principle compete with the general policy of the least-squares minimization because neither of the policies is absolutely better than another according to the main criterion of Newton's iteration processes, that is, the minimization of the 2-norm of the residuals. By increasing the computational work a little, one may compare the resulting residual norms after performing an iteration step under two or three distinct policies of compression and then go to the next step with the output where the norm is the smallest.

Remark 3.9. For a given displacement $\nabla_{A,B}(M)$ the choice of the generator pair G, H satisfying (3.1) is not unique, but, clearly, this choice does not affect the norm $\hat{r}_{i+1,G,H}$ in Theorem 3.8, which only depends on GH^T . Likewise, the choice of a pair of \hat{G}_{i+1} and \hat{H}_{i+1} for a fixed displacement $\nabla_{B,A}(\hat{X}_{i+1})$ in (3.8) is not unique, but no transformation into a pair of $\hat{G}_{i+1}^{new} = \hat{G}_{i+1}V$

and $\hat{H}_{i+1}^{new} = \hat{H}_{i+1}W$, for nonsingular matrices V and W , may decrease the norms $r_{i+1,G}$ and $r_{i+1,H}$, because we may choose $Y_{i+1,G}^{old} = V^{-1}Y_{i+1,G}^{new}$, $Y_{i+1,H}^{old} = W^{-1}Y_{i+1,H}^{new}$. A huge area of new opportunities, however, opens when we allow any modification of \hat{G}_{i+1} by $(\hat{G}_{i+1}, \hat{G}_i)$ and \hat{H}_{i+1} by $(\hat{H}_{i+1}, \hat{H}_i)$, or similarly combine two or more other candidate generator pairs. The size of the least-squares problems increases respectively, but this may be a reasonable price to pay for a more accurate compression, particularly at the initial stages of Newton's iteration when one has to save fragile convergence.

3.5 New iteration formula based on a cubic polynomial

For general input matrices, the scaled Newton iteration described in Subsections 2.2 and 2.3 uses the minimum number of iteration steps. However, with compression, this method needs very high displacement ranks for the intermediate results. Otherwise the method diverges. In this section, we design an iteration method based on a cubic polynomial. The numerical experiments in Section 4 show that the compression by truncation for this method is "autocorrective", i.e., even when the residual norm exceeds one, after some steps it becomes smaller than one again, and there is convergence.

Let us consider the following iteration formula:

$$X_{i+1} = aX_i(MX_i)^2 + bX_i(MX_i) + cX_i + dI. \quad (3.16)$$

When X_0 has the form

$$X_0 = V\Sigma_0U^T,$$

with $VU^T \neq I$, X_i also has the form $X_i = V\Sigma_iU^T$ if and only if $d = 0$.

Suppose M is nonsingular. Then the cubic iteration step (3.16) can also be written as follows:

$$Y_{i+1} = aY_i^3 + bY_i^2 + cY_i, \quad \text{with } Y_i = MX_i.$$

Hence, Y_i is of the form $Y_i = U\Sigma\Sigma_iU^T = U\Lambda_iU^T$. We have to choose a , b , and c such that Y_i converges to I , i.e., Λ_i converges to I . Therefore, we impose the following conditions on the iteration function $F(y) = ay^3 + by^2 + cy$:

$$\begin{aligned} F(1) &= 1 && 1 \text{ is a fixed point,} \\ F'(1) &= 0 && \text{quadratic convergence in the neighbourhood of } y = 1. \end{aligned}$$

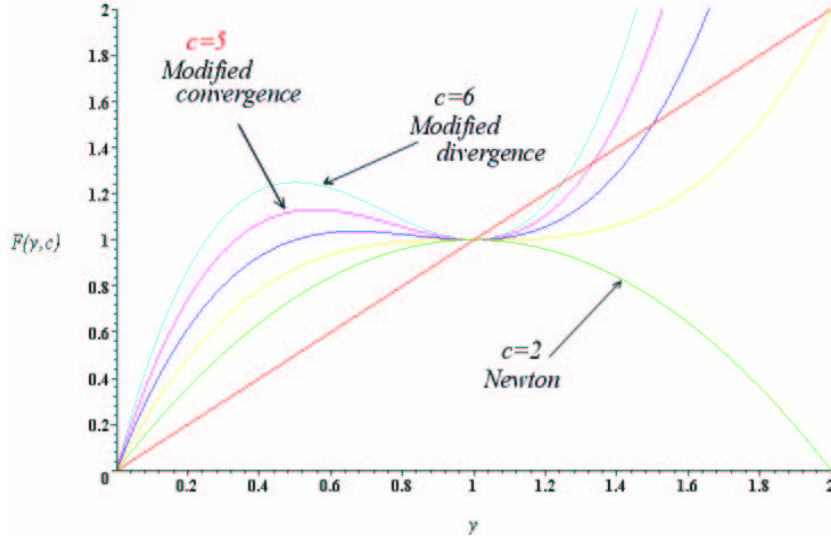


Figure 1: The function $F(y)$ on $[0, 2]$ for $c = 2, 3, 4, 5, 6$

These two conditions give us $F(y)$ parameterized by c :

$$F(y) = (c - 2)y^3 + (3 - 2c)y^2 + cy.$$

Note that for $c = 2$, we obtain again the classical Newton iteration function. Let us now choose the value of c . Figure 1 shows the function $F(y)$ for $c = 2, 3, 4, 5, 6$ with $y \in [0, 2]$. To accelerate the convergence for small initial values of y , we want to make $F'(0) = c$ as large as possible (cf. Subsection 2.4). When $2 \leq c \leq 3$, it is easy to check that $F([0, 1]) = [0, 1]$. When $c > 2$, we have three fixed points

$$y_1 = 0 < y_2 = 1 < y_3 = \frac{c - 1}{c - 2}.$$

To guarantee that $F([0, y_3]) \subset [0, y_3)$ when $c > 3$, we have the sufficient condition

$$\max_{y \in (0, 1)} F(y) = F\left(\frac{c}{3(c - 2)}\right) < y_3 = \frac{c - 1}{c - 2}.$$

Hence, c should satisfy $3 < c < 6$. When $c = 6$, we have

$$\max_{y \in (0, 1)} F(y) = F\left(\frac{c}{3(c - 2)}\right) = y_3.$$

Therefore, to avoid a lot of iteration steps for $y_0 \approx \frac{c}{3(c-2)}$, we should take c not too close to 6. In our experiments (see Section 4) we have chosen $c = 5$. In this case, when $y_0 \approx 0$ and positive, we get $y_k \approx y_0 5^k$, compared to $y_k \approx y_0 2^k$ for the classical Newton iteration (2.1) and $y_k \approx y_0 4^k$ for the scaled iteration (2.3)–(2.5), although we use a cubic polynomial in MX_i versus quadratic polynomials in (2.1) and (2.3). (Compare also the cubic polynomials in (2.16) and Remark 2.2.)

4 Numerical experiments

In the experiments of this section, we use the first compression rule described in Subsection 3.2, i.e., truncation of the smallest singular values of the displacement (cutting). For numerical experiments comparing this compression with the least-squares compression, we refer the reader to [7].

4.1 Norm estimation

Computing the 2-norm would require too much computational effort.

An upper bound on the 2-norm can be computed in an efficient and robust way by applying the following theorem based on the circulant displacement representation [1, 8, 24, 5, 27]:

Theorem 4.1. *Let T be a Toeplitz-like matrix with circulant displacement rank k , i.e. $\text{rank}(\Delta^+(T)) = k$ and displacement representation $\Delta(T) = U\Sigma V^H$ then an upper bound for the 2-norm is:*

$$\|T\| \leq \frac{1}{2} \sum_{i=1}^k \sigma_i \max_j |F^H \mathbf{u}_i| * \max_j |F^H D^H \mathbf{v}_i|$$

where

$$F = (\omega^{(i-1)(j-1)})_{i,j=1,\dots,n}, \quad \omega = \cos(2\pi/n) + \mathbf{i} \sin(2\pi/n),$$

$$D = \text{diag}(1, \theta, \theta^2, \dots, \theta^{n-1}), \quad \theta = \cos(\pi/n) + \mathbf{i} \sin(\pi/n), \quad i = \sqrt{-1}.$$

Proof. The matrix T can be written as $\frac{1}{2} \sum_{i=1}^k C^+(\mathbf{u}_i) C^-(\mathbf{v}_i)$ (see [5]).

Hence, we derive that

$$\begin{aligned}
\|T\| &= \frac{1}{2} \left\| \sum_{i=1}^k \sigma_i F \text{diag}(\mathbf{y}_i) F^H D F \text{diag}(\hat{\mathbf{y}}_i) F^H D^H \right\| \\
&\leq \frac{1}{2} \sum_{i=1}^k \sigma_i \|F\| \|\text{diag}(\mathbf{y}_i)\| \|F^H\| \|D\| \|F\| \\
&\leq \frac{1}{2} \sum_{i=1}^k \sigma_i n \frac{\max |F^H \mathbf{u}_i|}{n} n \frac{\max |F^H D^H \mathbf{v}_i|}{n}.
\end{aligned}$$

□

4.2 Numerical results

The code for our numerical experiments has been written in C++. We have designed an object-oriented library in order to model the main entities used in the Newton iteration procedure and its variants. The main characteristics are:

- The class *matrix* which models a general dense rectangular matrix, and makes use of the fast routines of the *ATLAS package*.
- The class *TLike* which models a Toeplitz-like matrix using its *circulant orthogonal displacement representation*.
- The FFTs are performed using the fast routines of the *FFTW* package.
- The algorithms: *DeltaNorm*, *DeltaScal*, *DeltaSum*, *DeltaProd*, *ODG* and *CUT* which represent fast norm estimation, fast scaling, sum and product operation of TLike objects, reorthogonalization of the generators and compression of the generators via the truncation of the smallest singular values of the displacement representation, respectively.

Based on the previous sections, we devise a modified Newton iteration in terms of the operators in the library (see Algorithm 1).

The iteration in Subsection 3.5, based on a cubic polynomial, requires an extra operation of matrix multiplication and some extra operations of matrix additions and scalings. This may cause a dramatic growth of the displacement rank, up to 30 in each iteration. Such an iteration is not competitive

<p>1: INPUT: The <i>TLike</i> matrix M, the tolerance ϵ and the cutting level $rank$.</p> <p>2: $X_0 = I$; the initial approximate inverse (for a symmetric positive definite matrix M).</p> <p>3: $M = \text{DeltaScal}(M, \text{DeltaNorm}(M))$; normalize the matrix.</p> <p>4: $Res = \text{DeltaSum}(-I, \text{DeltaProd}(X_0, M))$; residual matrix.</p> <p>5: $k = 0$;</p> <p>6: while $\text{DeltaNorm}(Res) > \epsilon$ do</p> <p>7: $k = k + 1$;</p> <p>8: $X_{kM} = \text{DeltaProd}(M, X_{k-1})$; compute the product MX_{k-1}.</p> <p>9: $X_{kM} = \text{DeltaProd}(X_{k-1}, X_{kM})$; compute the product $X_{k-1}MX_{k-1}$.</p> <p>10: $X_k = \text{DeltaSum}(X_{k-1}, \text{DeltaScal}(X_{kM}, -1))$; compute the sum $2X_{k-1} - X_{k-1}AX_{k-1}$. Now X_k represents the result of the kth step of the Newton iteration, and it has to be compressed.</p> <p>11: $X_k = \text{ODG}(X_k)$; reorthogonalization for a fast norm estimation.</p> <p>12: $X_k = \text{CUT}(X_k, rank)$;</p> <p>13: $Res = \text{DeltaSum}(-I, \text{DeltaProd}(X_k, M))$;</p> <p>14: end while</p> <p>15: OUTPUT: an approximate inverse X_k of the normalized matrix M.</p>
--

Algorithm 1: modified Newton's iteration

because the computational cost in each iteration step is too large. We solve this problem by compressing the intermediate results in each iteration step, thus maintaining small displacement rank in each matrix multiplication.

The parameter $over_{rank}$ has been estimated heuristically. From our extensive numerical experiments, it turns out that $over_{rank} = 2$ is a good choice. We can not set $over_{rank}$ equal to zero because in this case we would have lost too much information about the generators, and then generally the iteration diverges. Numerical tests show that two iteration steps of the classical Newton method take 20 to 25 percents more execution time than one new cubic iteration step. The total number of cubic iteration steps for $c = 5$ is always about one half of the number of steps required by the classical Newton method as is shown by the next experiments.

We have compared the new cubic iteration to the classical one for various classes of Toeplitz matrices. Here we show the results for two classes of positive definite Toeplitz matrices. The tests for the other classes of Toeplitz matrices give comparable results. The first class of matrices is generated by the symbol $f(x) = \frac{2x^4}{1+25x^2}$. These matrices have a Euclidean condition number which grows as $O(n^4)$ where n is the dimension of the matrix. In Table 4.1, the number of iteration steps and the norm of the residual attained are shown for different values of the parameter C of the cubic polynomial: $C = 2, 3, 4, 5$.

The second matrix class is generated by $f(x) = \frac{2x^2}{1+25x^2}$. In this case the condition number grows as $O(n^2)$ and in Table 4.2, we show the results.

We also performed numerical tests by using the scaled Newton iteration of Subsections 2.2 and 2.3 which needs the smallest number of iteration steps when no compression is used. However, whenever compression is included using a compression level similar to that in the experiments with cubic iteration, the scaled Newton iteration diverges because the residual norm exceeds one and the method does not recover from that. This seems to be related to the fact that the scaled iteration method in Subsection 2.2 and 2.3 (optimal for general matrices) maps the interval $[\sigma_r^2, \sigma_1^2]$ (or the interval $[\sigma_r, \sigma_1]$ for symmetric positive definite matrices) so that small singular values of the input matrix are exchanged with its large singular values. Together with the compression, this leads to divergence.

```

1: INPUT: The TLike matrix  $M$ , the tolerance  $\epsilon$ , the compression level
    $rank$ , the  $over_{rank}$  and the parameter  $C$ .
2:  $X_0 = I$ ; the initial approximation (for a symmetric positive definite
   matrix  $M$ ).
3:  $M = DeltaScal(M, DeltaNorm(M))$ ; Normalize the matrix.
4:  $Res = DeltaSum(-I, DeltaProd(X_0, M))$ ; residual matrix.
5:  $k = 0$ ;
6: while  $DeltaNorm(I - X_k * M) > \epsilon$  do
7:    $k = k + 1$ ;
8:    $\mathbf{X}_{P1} = DeltaProd(\mathbf{M}, \mathbf{X}_{k-1})$ ; rapidly compute the product  $MX_{k-1}$ .
9:    $ODG(X_{P1})$ ; reorthogonalization.
10:  CUT( $\mathbf{X}_{P1}$ ,  $rank + over_{rank}$ ); compression
11:   $\mathbf{X}_{P2} = DeltaProd(\mathbf{X}_{k-1}, \mathbf{X}_{P1})$ ; rapidly compute the product
    $X_{k-1}MX_{k-1}$ .
12:   $ODG(X_{P2})$ ; reorthogonalization.
13:  CUT( $\mathbf{X}_{P2}$ ,  $rank + over_{rank}$ ); compression.
14:   $\mathbf{X}_{P3} = DeltaProd(\mathbf{X}_{P2}, \mathbf{X}_{P1})$ ; rapidly compute the product
    $X_{k-1}MX_{k-1}MX_{k-1}$ .
15:  FREE  $X_{P1}$ ; crucial operation for large matrices; at this point  $X_{P1}$ 
   can be deleted freeing a large portion of memory.
16:   $ODG(X_{P3})$ ; reorthogonalization.
17:  CUT( $\mathbf{X}_{P3}$ ,  $rank + over_{rank}$ ); compression.
18:   $DeltaScalIP(C, X_{k-1})$ ; scaling in place.
19:   $DeltaScalIP((3 - 2C), X_{P2})$ ; scaling in place.
20:   $DeltaScalIP((C - 2), X_{P3})$ ; scaling in place.
21:   $X_k = DeltaSum(X_{P3}, DeltaSum(X_{P2}, X_{k-1}))$ 
22:   $X_k = ODG(X_k)$ ; reorthogonalization for a fast norm estimation.
23:   $X_k = CUT(X_k, rank)$ ;
24:   $Res = DeltaSum(-I, DeltaProd(X_k, M))$ ;
25: end while
26: OUTPUT: the approximated inverse  $X_k$  of the normalized matrix  $M$ .

```

Algorithm 2: Newton's iteration based on a cubic polynomial

Table 4.1: Euclidean condition number $O(n^4)$.

Residual	Iterations	Dimension	Condition Number
C = 2, The Newton iteration			
3.20814e-06	14	2^5	2.7026e+03
1.52751e-05	17	2^6	2.9174e+05
6.12537e-04	20	2^7	2.9174e+05
3.64023e-04	24	2^8	3.9709e+06
3.79358e-02	27	2^9	5.8560e+07
C = 3			
1.14001e-10	9	2^5	2.7026e+03
1.18725e-08	11	2^6	2.9174e+05
1.72215e-06	13	2^7	2.9174e+05
1.23804e-04	15	2^8	3.9709e+06
4.33197e-02	17	2^9	5.8560e+07
C = 4			
5.06532e-08	8	2^5	2.7026e+03
1.38793e-07	9	2^6	2.9174e+05
4.54479e-06	11	2^7	2.9174e+05
2.49081e-03	12	2^8	3.9709e+06
6.3061e-02	13	2^9	5.8560e+07
C = 5			
1.41586e-08	8	2^5	2.7026e+03
4.782e-05	9	2^6	2.9174e+05
4.22898e-06	10	2^7	2.9174e+05
2.3277e-03	11	2^8	3.9709e+06
7.64821e-02	12	2^9	5.8560e+07

5 Conclusion

In this paper, we give an overview of several iterative methods based on Newton's iteration for approximating the inverse of general matrices and then focus on some classes of structured matrices. For structured matrices, we analyze and extend the techniques of the least-squares compression/refinement introduced in [7] and propose a new cubic iteration formula. The numerical experiments indicate that when compression is done at each iteration step,

Table 4.2: Euclidean condition number $O(n^2)$.

Residual	Iterations	Dimension	Condition Number
C = 5			
8.59137e-10	6	2^7	7.7852e+01
8.3459e-10	6	2^8	2.8664e+02
7.4425e-11	8	2^9	1.1010e+03
2.03686e-09	9	2^{10}	4.3169e+03
2.12262e-08	10	2^{11}	1.7097e+04
2.96647e-07	11	2^{12}	$\approx 7e+04$
5.4985e-06	11	2^{13}	$\approx 3e+05$
8.57205e-05	12	2^{14}	$\approx 1e+06$
1.27038e-03	13	2^{15}	$\approx 4e+06$
2.78462e-02	13	2^{16}	$\approx 2e+07$
The Classical Newton Iteration			
4.41563e-10	10	2^7	7.7852e+01
8.31367e-11	12	2^8	2.8664e+02
3.73467e-11	14	2^9	1.1010e+03
2.48426e-11	16	2^{10}	4.3169e+03
1.89551e-11	18	2^{11}	1.7097e+04
3.09097e-11	20	2^{12}	$\approx 7e+04$
6.57986e-11	22	2^{13}	$\approx 3e+05$
3.74132e-05	23	2^{14}	$\approx 1e+06$
7.46543e-05	25	2^{15}	$\approx 4e+06$
1.51784e-04	27	2^{16}	$\approx 2e+07$

the cubic iteration method outperforms the scaled Newton iteration which is superior when no compression is involved. This superior behaviour is due to the autocorrection of the compression of the cubic iteration (which was observed earlier for modified Newton's iteration), i.e., even when in some iteration step the residual norm exceeds one, some additional iteration steps with compression bring it down below one, and the method converges. A theoretical explanation of this phenomenon is under investigation.

References

- [1] G. S. Ammar and P. Gader. New decompositions of the inverse of a Toeplitz matrix. In M. A. Kaashoek, J. H. van Schuppen, and A. C. N. Ran, editors, *Signal Processing, Scattering and Operator Theory, and Numerical Methods*, pages 421–428. Birkhäuser, 1990.
- [2] A. Ben-Israel. A note on iterative method for generalized inversion of matrices. *Mathematics of Computation*, 20:439–440, 1966.
- [3] A. Ben-Israel, D. Cohen, On iterative computation of generalized inverses and associated projections. *SIAM Journal on Numerical Analysis*, 3:410–419, 1966.
- [4] D. A. Bini, G. Codevico and M. Van Barel Solving Toeplitz least square problems by means of Newton’s iterations. *Numerical Algorithms*, Kluwer, 33:63–103, 2003.
- [5] D.A. Bini and B. Meini Solving block banded block Toeplitz systems with banded Toeplitz blocks. F.T. Luk, *Proceedings of SPIE*, 3807:300–311, 1999.
- [6] D. A. Bini and V. Y. Pan. *Polynomial and Matrix Computations, vol. 1: Fundamental Algorithms*. Birkhäuser, Boston, 1994.
- [7] G. Codevico, V. Pan, M. Van Barel, X. Wang. *Iterative Inversion of Structured Matrices*. Report TW351, Department of Computer Science, Katholieke Universiteit Leuven, 2002 (accepted for a special issue of Theoretical Computer Science on Algebraic and Numerical Algorithms).
- [8] I. Gohberg and V. Olshevsky. Circulants, displacements and decompositions of matrices. *Integral Equations and Operator Theory*, 15 (1992), 730–743.
- [9] I. Gohberg and A. Semencul. On the inversion of finite Toeplitz matrices and their continuous analogs. *Matematicheskie Issledovaniia*, 2:187–224, 1972.
- [10] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, Baltimore, Maryland, 1996.

- [11] G. Heinig. *Beitrage zur spektraltheorie von Operatorbushec und zur algebraischen Theoriei von Toeplitzmatrizen*. PhD thesis, TH Karl-Marx-Stadt, 1979.
- [12] G. Heinig and K. Rost. *Algebraic Methods for Toeplitz-like Matrices and Operators*. Akademie-Verlag, Berlin, and Birkhäuser, Basel/Stuttgart, 1984.
- [13] T. Kailath, S.-Y. Kung, and M. Morf. Displacement ranks of matrices and linear equations. *Journal of Mathematical Analysis and Its Applications*, 68:395–407, 1979.
- [14] T. Kailath, S. Kung, and M. Morf. Displacement ranks of a matrix. *Bulletin of the American Mathematical Society*, 1:769–773, 1979.
- [15] T. Kailath and A. Sayed. Displacement structure: Theory and applications. *SIAM Review*, 37(3):297–386, 1995.
- [16] T. Kailath and A. H. Sayed (editors). *Fast Reliable Algorithms for Matrices with Structure*. SIAM Publications, Philadelphia, 1999.
- [17] T. Kailath, A. Vieira, and M. Morf. Inverses of Toeplitz operators, innovations and orthogonal polynomials. *SIAM Review*, 20:106–119, 1978.
- [18] M. Morf. *Fast Algorithms for Multivariable Systems*. PhD thesis, Department of Electrical Engineering, Stanford University, Stanford, CA, 1974.
- [19] V. Y. Pan. Nearly Optimal Computations with Structured Matrices. *Proceedings of 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2000)*, 953–962, ACM Press, New York, and SIAM Publications, Philadelphia, 2000.
- [20] V. Y. Pan. *Structured Matrices and Polynomials: Unified Superfast Algorithms*. Birkhäuser/Springer, Boston/New York, 2001.
- [21] V. Y. Pan, M. Kunin, R. E. Rosholt, and H. Cebecioğlu. Residual correction algorithms for general and structured matrices, 2002. Preprint.

- [22] G. Pólya, G. Szegő, *Problems and Theorems in Analysis II: Theory of Functions, Zeros, Polynomials, Determinants, Number Theory, Geometry*, Springer Verlag, New York, 1976.
- [23] V. Y. Pan and R. Schreiber. An improved Newton iteration for the generalized inverse of a matrix, with applications. *SIAM Journal on Scientific and Statistical Computing*, 12(5):1109–1131, 1991.
- [24] V. Y. Pan and X. Wang. Inversion of displacement operators. *SIAM Journal on Matrix Analysis and Applications*, 24(3):660–677, 2003.
- [25] G. Schultz, Iterative Berechnung der Reciproken Matrix, *Z. Angew. Meth. Mech.*, 13:57–59, 1933.
- [26] T. Söderström and G. W. Stewart. On the numerical properties of an iterative method for computing the Moore–Penrose generalized inverse. *SIAM Journal on Numerical Analysis*, 11:61–74, 1974.
- [27] M. Van Barel and G. Codevico. *An adaptation of the Newton iteration method to solve symmetric positive definite Toeplitz systems*. Report TW349, Department of Computer Science, Katholieke Universiteit Leuven, 2002.