

**An adaptation of the Newton iteration method
to solve symmetric positive definite Toeplitz
systems**

Marc Van Barel Gianni Codevico

Report TW 349, November 2002



Katholieke Universiteit Leuven
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

An adaptation of the Newton iteration method to solve symmetric positive definite Toeplitz systems

Marc Van Barel Gianni Codevico

Report TW 349, November 2002

Department of Computer Science, K.U.Leuven

Abstract

The classical Newton iteration method for matrices can be modified into an efficient algorithm when structured matrices are involved. The difficulty, however, is the importance of the choice of the starting matrix. In this paper, we propose a new initial iteration step which makes the choice of the starting matrix less critical. The validity of the approach is illustrated by numerical experiments.

Keywords : Toeplitz, Toeplitz-like, displacement rank, Newton iteration, starting value
AMS(MOS) Classification : Primary : 65F10, Secondary : 65F20.

An adaptation of the Newton iteration method to solve symmetric positive definite Toeplitz systems ^{*}

Marc Van Barel ^{*}, Gianni Codevico

*Katholieke Universiteit Leuven, Department of Computer Science,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium*

Abstract

The classical Newton iteration method for matrices can be modified into an efficient algorithm when structured matrices are involved. The difficulty, however, is the importance of the choice of the starting matrix. In this paper, we propose a new initial iteration step which makes the choice of the starting matrix less critical. The validity of the approach is illustrated by numerical experiments.

Key words: Toeplitz, Toeplitz-like, displacement rank, Newton iteration, starting value.

^{*} The research 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), and G.0176.02 (Asymptotic analysis of the convergence behavior of iterative methods in numerical linear algebra), 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.

^{*} Corresponding author.

URL: www.cs.kuleuven.ac.be/~marc (Marc Van Barel).

1 Introduction

Let $T = [t_{ij}]$ be an $n \times n$ matrix. We say that T is a Toeplitz matrix if $t_{ij} = t_{i-j}$, i.e.,

$$T = \begin{bmatrix} t_0 & t_{-1} & \dots & t_{-n+1} \\ t_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \dots & t_1 & t_0 \end{bmatrix}.$$

In this paper we consider the problem of solving a linear system $Tx = b$ where the matrix $T \in \mathbb{R}^{n \times n}$ is a symmetric positive definite (possibly ill-conditioned) Toeplitz matrix. We do this by computing an approximation of T^{-1} . The method presented is a variation of the Newton iteration method for matrices (see, e.g., [22] and a more recent paper [20]). This method generalizes the scalar Newton method as explained, e.g., in [1], to matrices.

For a general square matrix A , the Newton iteration method starts with an approximation X_0 for the inverse of A . Using this starting point, we can construct the sequence of matrices X_0, X_1, \dots as follows

$$X_k = 2X_{k-1} - X_{k-1}AX_{k-1}, \quad k = 1, 2, \dots$$

The (left) residual matrix R_k is defined as $R_k = I - X_kA$. It is easy to show that $R_{k+1} = R_k^2$. Hence, if the starting point X_0 is chosen such that the norm of $R_0 = I - X_0A$ is less than 1, the sequence of matrices X_0, X_1, X_2, \dots converges quadratically to A^{-1} when using infinite precision. It is however difficult to choose a good starting point in an efficient way. Especially when the matrix A is ill-conditioned and using finite precision arithmetic, this is not a trivial task. The choice of the starting point is crucial in the initial iteration steps of the method. The purpose of this paper is to replace the first iteration step by another one such that the convergence behaviour depends less critically on the choice of the starting point X_0 when using finite precision and when using the modified Newton iteration.

Let us explain how this *modified* Newton iteration works. In general, each Newton iteration step requires $\mathcal{O}(n^3)$ flops. When we apply the Newton iteration to a structured matrix, e.g., a Toeplitz matrix, we can make use of the structure as follows. Such an $n \times n$ structured matrix can be represented using $\mathcal{O}(rn)$ parameters where r is called the displacement rank. If we take a starting point X_0 having a similar structure as the matrix A , it turns out that the number of parameters to represent X_{k+1} is 3 times the number of

parameters needed to represent X_k . Hence, after a few iteration steps, the number of parameters $\mathcal{O}(3^k rn)$ has become so large that it is not interesting anymore to take into consideration the structure of the matrices X_k . That's why after each iteration step, we need to compress, i.e., we need to approximate the matrix X_{k+1} by a matrix which can be represented again by a limited number of parameters, say, e.g., $\mathcal{O}(ln)$ parameters where l is the maximum displacement rank we allow for the matrices X_k . Hence, the modified Newton iteration looks as follows:

$$\tilde{X}_{k+1} = 2X_k - X_k A X_k, \quad X_{k+1} = \text{compress}(\tilde{X}_{k+1}).$$

When the number of parameters stays limited to $\mathcal{O}(ln)$ during the iteration, each iteration step can be performed using $\mathcal{O}(ln \log(n))$ flops when the matrix A is Toeplitz or, more general, Toeplitz-like. However, because the Newton iteration is not performed exactly, the choice of the starting point is even more critical to obtain convergence. The techniques for the Newton iteration method for structured matrices were developed in [16,17,14,13,21]. For a detailed overview of the method applied to structured matrices, we refer the interested reader to [19], [18, Chapter 6] and the references therein. There is also an algebraic version of Newton's iteration, see, e.g., [15,4] and [5, p. 189-190]. The Newton iteration method can also be used to solve structured least squares problems, see, e.g., [2].

In Section 2, we will recall the notions of (circulant) displacement rank and ϵ -displacement rank. Section 3 contains the main result of the paper and explains an alternative first iteration step leading to a more robust algorithm, i.e., an algorithm where the choice of the starting point X_0 is less critical. In Section 5, some numerical experiments show the effectiveness of the new approach. In our implementation, we used the circulant displacement representation. An important component of this implementation is the norm estimation of the residual matrix. This is explained in Section 4.

2 Circulant displacement rank and ϵ -displacement rank

We recall here the concept of circulant displacement rank and of ϵ -displacement rank. Displacement operators and displacement rank, introduced in [12,8,9], and elaborated in [7,5,10,11], are powerful tools to deal with structured matrices. Let $Z_n = [z_{i,j}]$ be the $n \times n$ lower shift matrix such that $z_{i,j} = 1$ for $i = j + 1$ and $z_{i,j} = 0$ elsewhere, and introduce the displacement operator $\Delta_n(A) = Z_n A - A Z_n$ for an $n \times n$ matrix A . For the sake of notational simplicity we write $\Delta(A)$ in place of $\Delta_n(A)$ when the dimension of the matrix A is clear from the context.

Define the displacement rank of A , associated with the displacement operator Δ , as $\text{drk}(A) = \text{rank}(\Delta(A))$. When the displacement rank k is small compared to the dimension of the matrix, we can work with the displacement $\Delta(A)$ instead of the matrix A because this displacement can be efficiently represented as

$$\Delta(A) = \sum_{i=1}^k \mathbf{u}_i \mathbf{v}_i^T$$

where $\mathbf{u}_i, \mathbf{v}_i, i = 1, \dots, k$, are n -dimensional vectors. These vectors are called the generators of the matrix A with respect to the displacement operator Δ . It is easy to check that the displacement rank of a Toeplitz matrix is 2. We call a matrix Toeplitz-like if its displacement rank is k with $k \ll n$ where n is the dimension of the matrix.

Let $L(\mathbf{a})$ be the $n \times n$ lower triangular Toeplitz matrix defined by its first column \mathbf{a} . If we know the displacement of A , we can easily retrieve A as follows

$$\Delta(A) = \sum_{i=1}^k \mathbf{u}_i \mathbf{v}_i^T \quad \Leftrightarrow \quad A = L(A\mathbf{e}_1) + \sum_{i=1}^k L(\mathbf{u}_i)L^T(Z\mathbf{v}_i), \quad (2.1)$$

where \mathbf{e}_1 denotes the first column of the identity matrix. The expression in the right-hand side of (2.1) is called a displacement representation of A . Equivalent representations can be given in terms of different displacement operators.

Each Newton iteration step can be performed efficiently using the displacement representation of the matrices involved. Multiplying a Toeplitz-like matrix and a vector by means of the previously defined displacement representation of length k involves $4k + 3$ FFTs and $2k + 1$ convolutions of length $2n$.

It is possible to reduce the lengths of FFTs and convolutions by a factor of 2 by means of a different displacement representation. For completeness, we summarize here the main results of [3, Section 2]. This displacement representation is a representation only in terms of generators, without the knowledge of the first column of the matrix. This leads to a considerable reduction in the memory space needed to store a Toeplitz-like matrix. Therefore, we will use from now on this representation for Toeplitz(-like) matrices, called the circulant displacement representation which is introduced as follows (see also [6] and the book [5]). Let us define the matrices C^+ and C^- as $C^+ = Z + \mathbf{e}_1 \mathbf{e}_n^T$, $C^- = Z - \mathbf{e}_1 \mathbf{e}_n^T$, and set $C^+(\mathbf{x}) = \sum_{i=1}^n x_i (C^+)^{i-1}$, $C^-(\mathbf{x}) = \sum_{i=1}^n x_i (C^-)^{i-1}$ the (+1)-circulant and (-1)-circulant matrices having \mathbf{x} as their first column.

We have that:

$$\begin{aligned}
C^+(\mathbf{x}) &= F \text{diag}(\mathbf{y}) F^H, \\
C^-(\mathbf{x}) &= DF \text{diag}(\hat{\mathbf{y}}) F^H D^H, \\
\mathbf{y} &= \frac{1}{n} F^H \mathbf{x}, \hat{\mathbf{y}} = \frac{1}{n} F^H D^H \mathbf{x}, \\
F &= (\omega^{(i-1)(j-1)})_{i,j=1,\dots,n}, \quad \omega = \cos(2\pi/n) + \mathbf{i} \sin(2\pi/n) \\
D &= \text{diag}(1, \theta, \theta^2, \dots, \theta^{n-1}), \quad \theta = \cos(\pi/n) + \mathbf{i} \sin(\pi/n),
\end{aligned} \tag{2.2}$$

where \mathbf{i} is the complex number such that $\mathbf{i}^2 = -1$. Introduce the (invertible) circulant displacement operators

$$\begin{aligned}
\Delta^+(A) &= C^+ A - A C^- \\
\Delta^-(A) &= C^- A - A C^+
\end{aligned}$$

such that

$$\begin{aligned}
\Delta^+(A) &= -\Delta(A) + \mathbf{e}_1 \mathbf{e}_n^T A + A \mathbf{e}_1 \mathbf{e}_n^T \\
\Delta^-(A) &= -\Delta(A) - \mathbf{e}_1 \mathbf{e}_n^T A - A \mathbf{e}_1 \mathbf{e}_n^T.
\end{aligned} \tag{2.3}$$

We have the following properties.

Theorem 1 *It holds that*

$$\begin{aligned}
\Delta^+(A) &= \sum_{i=1}^k \mathbf{u}_i \mathbf{v}_i^T \Leftrightarrow A = \frac{1}{2} \sum_{i=1}^k C^+(\mathbf{u}_i) C^-(J \mathbf{v}_i), \\
\Delta^-(A) &= \sum_{i=1}^k \mathbf{u}_i \mathbf{v}_i^T \Leftrightarrow A = -\frac{1}{2} \sum_{i=1}^k C^-(\mathbf{u}_i) C^+(J \mathbf{v}_i), \\
\Delta^-(A^{-1}) &= -A^{-1} \Delta^+(A) A^{-1}, \\
\Delta^+(AB) &= \Delta^+(A) B + A \Delta^+(B) - 2A \mathbf{e}_1 \mathbf{e}_n^T B, \\
\Delta^-(AB) &= \Delta^-(A) B + A \Delta^-(B) + 2A \mathbf{e}_1 \mathbf{e}_n^T B,
\end{aligned}$$

where J is the permutation matrix having 1 on the anti-diagonal. In particular, if $\Delta^+(A) = \sum_{i=1}^k \mathbf{u}_i \mathbf{v}_i^T$ and $\det A \neq 0$, we obtain

$$A^{-1} = \frac{1}{2} \sum_{i=1}^k C^-(A^{-1} \mathbf{u}_i) C^+(J A^{-T} \mathbf{v}_i).$$

PROOF. For the proof see [3, Th. 2.5]. \square

The displacement representation given in the above theorem allows one to compute the product $A\mathbf{x}$ by means of $4k+2$ FFTs, $2k$ convolutions and $2k+1$ scalings of length n . It is also possible to apply the concept of *orthogonal displacement representation* (ODR) and the concept of the ϵ -displacement

rank to the representation obtained through the operators Δ^+ and Δ^- as follows. Due to the non-singularity of Δ^+ and Δ^- the ODR associated with Δ^- and Δ^+ is simply defined by the triple $(U, \boldsymbol{\sigma}, V)$ such that $U\Sigma V$ is the SVD of $\Delta^+(A)$ or of $\Delta^-(A)$, respectively, where $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_k)$. Consider the operator $\text{trunc}_\epsilon(\cdot)$, defined on the sets of orthogonal displacement generators in the following way:

$$\begin{aligned} \text{trunc}_\epsilon((U, \boldsymbol{\sigma}, V)) &= (\hat{U}, \hat{\boldsymbol{\sigma}}, \hat{V}), \\ \hat{\boldsymbol{\sigma}} &= (\sigma_1, \dots, \sigma_{\hat{k}}), \hat{U} = [\mathbf{u}_1, \dots, \mathbf{u}_{\hat{k}}], \hat{V} = [\mathbf{v}_1, \dots, \mathbf{v}_{\hat{k}}] \in \mathbb{R}^{n \times \hat{k}} \\ \sigma_{\hat{k}} &> \sigma_1 \epsilon \geq \sigma_{\hat{k}+1}, \end{aligned} \quad (2.4)$$

We can extend this definition to the set of matrices as $A_\epsilon = \text{trunc}_\epsilon(A)$. Equation (2.4) provides a means for computing an ϵ -displacement representation A_ϵ of A whenever an ODR is available. For further details concerning the ϵ -displacement rank, we refer the interested reader to [3].

3 Another initial iteration step

The displacement representation allows one to compute the product between a matrix with displacement rank k and a vector in $O(kn \log n)$ operations by means of FFT. The modified Newton iteration described in [3], is based on the idea of generating sequences of matrices with a displacement rank that grows exponentially while their ϵ -displacement rank remains bounded but still guaranteeing the convergence properties of the original algorithm, and at the same time, dramatically reducing the complexity of the computations.

The difficulty when compressing or cutting is choosing the value of ϵ or the value of \hat{k} in (2.4). Several tests have indicated that a *brute force* heuristic, restarting the method once divergence was verified, gives the best results.

In this section, we present a new version of the modified Newton iteration, introducing an initial iteration step which makes the choice of the first matrix less critical.

3.1 The initial iteration step

The biggest problem using the Newton iteration is connected with the local convergence of the method. The choice of the first approximation of the inverse is critical. A wrong choice leads to divergence. Consider a positive definite Toeplitz matrix T . Hence, it can be decomposed in its eigenvalue decomposition $U\Lambda U^H$. Using this decomposition it is possible to investigate the

behaviour of the Newton iteration by looking at the behaviour of the scalar Newton iteration method for each eigenvalue separately. In fact:

$$\begin{cases} T = T^H \Rightarrow T = U\Lambda U^H & \text{with } U^H U = I \\ X_0 = U\Lambda_0 U^H \\ X_{i+1} = 2X_i - X_i T X_i \end{cases} \Rightarrow \quad (3.1)$$

$$X_{i+1} = 2U\Lambda_i U^H - U\Lambda_i U^H U\Lambda U^H U\Lambda_i U^H = U(2\Lambda_i - \Lambda_i \Lambda \Lambda_i)U^H.$$

Note that we have chosen the starting point X_0 such that it has the same eigenvectors as T . A possible choice for X_0 which can be constructed relatively easy is $X_0 = \frac{T}{\eta^2}$ with $\eta \geq \|T\|_2$. In this case, the residual matrix R_0 is equal to $R_0 = I - X_0 T = U D U^T$ with $D = I - \frac{\Lambda^2}{\eta^2}$. Therefore, $\|R_0\|_2 = 1 - \frac{\min \lambda_i^2}{\eta^2}$. Here, λ_i are the eigenvalues of the original Toeplitz matrix T , i.e., the diagonal elements of Λ . This also shows that $\|R_0\|_2$ will be very near to 1 if the matrix T is ill-conditioned. Because we need an overestimation η of the 2-norm of T , once we have this number, we can compute the inverse of T/η instead of T . This has the advantage that the eigenvalues of T/η are within the interval $(0, 1]$. In Section 4, an efficient method is given to compute an overestimation of the 2-norm of a Toeplitz-like matrix based on its circulant displacement representation.

From (3.1), it follows that Newton's iteration computes implicitly the inverse of the eigenvalues by using a scalar Newton method (see Fig. 1). The critical

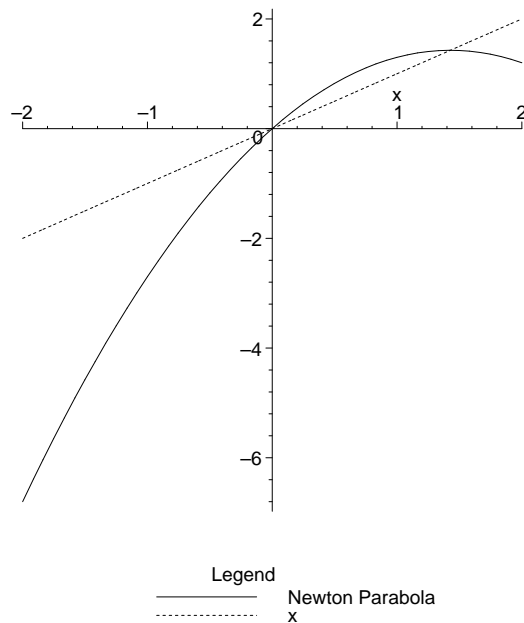


Figure 1. Newton parabola and bisectrice

point is when an eigenvalue lies in the neighbourhood of 0. When the starting

point X_0 has an eigenvalue in the neighbourhood of 0, this starting value for the implicit scalar Newton iteration can be updated into a negative value due to round-off errors and/or due to compressing, leading to divergence.

The idea is to modify the iteration such that in the first iteration step the critical zone is left (see Fig. 2). This is possible moving the iteration function away from the origin and at the same time keeping a high rate of convergence.

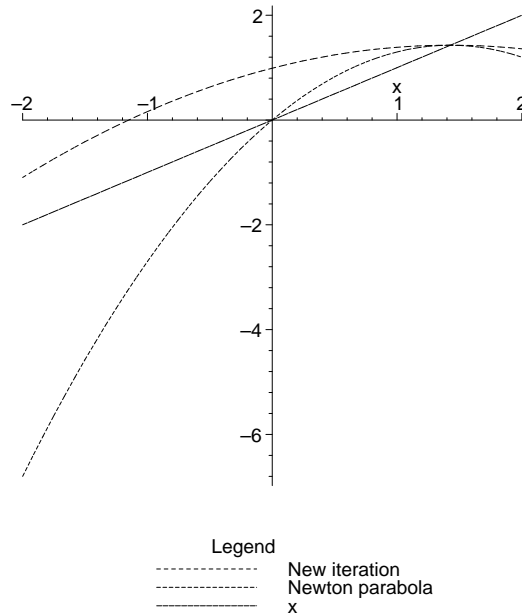


Figure 2. Shifted Iteration, Newton parabola and bisectrice

3.2 Computation of the coefficients

We want to show now the procedure that we followed to obtain the coefficients of the new iteration step. First of all we want to move the parabola described by the new iteration away from the origin and maintaining at the same time a high rate of convergence. This can be done as follows. Let $F(x) = ax^2 + bx + c$ and so $F'(x) = 2ax + b$.

We want:

$$\begin{cases} F(\frac{1}{\lambda}) = \frac{1}{\lambda} & \text{fixed point} \\ F'(\frac{1}{\lambda}) = \alpha & \text{very small} \Rightarrow \text{high convergence speed.} \end{cases} \Rightarrow$$

$$\begin{aligned}
 F'(\frac{1}{\lambda}) &= 2a\frac{1}{\lambda} + b = \alpha \\
 &\equiv \{\alpha = 10^{-4}\} \leftarrow \begin{array}{|l|} \hline \text{One chooses} \\ \hline \text{a small value} \\ \hline \end{array} \\
 a &= \frac{1}{2}\lambda(10^{-4} - b) \\
 &\Rightarrow \{F(x)|_{a=\frac{1}{2}\lambda(10^{-4}-b)}\} \leftarrow \begin{array}{|l|} \hline \text{Substitute the} \\ \text{occurrence of } a \text{ in } F(x) \\ \hline \end{array} \\
 F(x) &= \frac{1}{2}\lambda(10^{-4} - b)x^2 + bx + c \\
 &\Rightarrow \{F(\frac{1}{\lambda}) = \frac{1}{\lambda}\} \leftarrow \begin{array}{|l|} \hline \text{The fixed point relation} \\ \text{gives us } b \\ \hline \end{array} \\
 b &= -2c\lambda + (2 - 10^{-4}) \\
 &\Rightarrow \dots \\
 \alpha &= (10^{-4} - 1)\lambda + c\lambda^2 \\
 &\Rightarrow \dots \\
 F(x) &= [(10^{-4} - 1)\lambda + c\lambda^2]x^2 + [-2c\lambda + (2 - 10^{-4})]x + c.
 \end{aligned}$$

The coefficient c has to be positive, hence:

$$0 > x_1x_2 = \frac{c}{(10^{-4} - 1)\lambda + \lambda^2} \Rightarrow c < \frac{1 - 10^{-4}}{\lambda}. \quad (3.2)$$

Normalizing the matrix, one gets:

$$c < 1 - 10^{-4}.$$

If we choose $c = 99/100$, the final form of the iteration function is:

$$F(x) = -\frac{9999}{10000}\lambda x^2 + \frac{99}{100}\lambda^2 x^2 - \frac{99}{50}\lambda x + \frac{19999}{10000}x + \frac{99}{100}.$$

Now, to obtain the iteration in matrix form it is enough to re-apply the eigen-decomposition as in (3.1) but in the inverse direction, obtaining the final matrix form:

$$\begin{aligned}
 X_{i+1} &= aX_iT X_i + bX_iT^2 X_i + cX_iT + dX_i + eI \\
 a &= -\frac{9999}{10000}, b = \frac{99}{100}, c = -\frac{99}{50}, d = \frac{19999}{10000}, e = \frac{99}{100}.
 \end{aligned} \quad (3.3)$$

Let us give two examples to show how big the improvement is using the new initial iteration step, especially in ill-conditioned cases. Fig. 3 shows the case for “easy” eigenvalues, i.e., for eigenvalues near 1. For the “difficult” eigenvalues, i.e., the eigenvalues in the neighbourhood of zero, the new initial iteration step moves the initial value away from zero as can be observed in Fig. 4. Looking carefully in the origin you can see how many iterations should

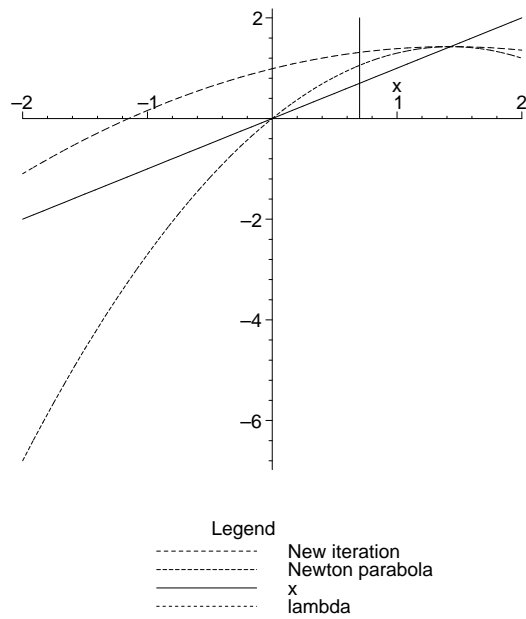


Figure 3. Shifted Iteration, Newton parabola, bisectrice and λ

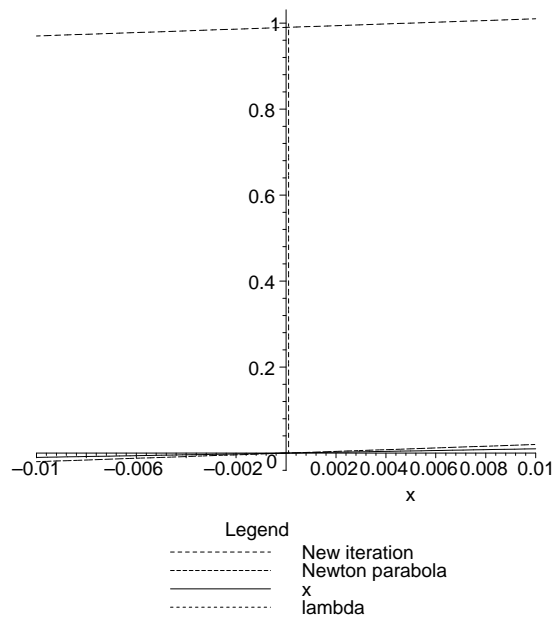


Figure 4. Shifted Iteration, Newton parabola, bisectrice and λ

be necessary for the classical Newton iteration to converge (or even diverge). But applying the new iteration one step is enough to run away from the critical zone and thereafter iterate with the modified Newton iteration.

Several experiments, as explained in Section 5, demonstrate that with this

new initial iteration the method is much more robust and the behaviour of the method in terms of number of iterations as a function of the condition number is very predictable. This consideration defines a complete scheme which can be applied also for Toeplitz-like matrices. Just estimating the condition number it's possible to have a very good prediction of the number of iterations necessary to compute a good approximation of the generators of the inverse. Once we have this approximation, we can compute an approximation of the solution of the linear system by multiplying the right-hand side vector by the approximation of the inverse using its (circulant) displacement representation. This can be done in $\mathcal{O}(kn \log(n))$ flops when the displacement rank of the inverse is k .

3.3 The non-symmetric case

In the non-symmetric case, it is not possible to transform the scalar iteration in a corresponding matrix form due to the fact that the decompositions (i.e. the SVD) are obviously non-symmetric. It is possible to extend our iteration to the non-symmetric case, by modifying the starting point of the Newton iteration, following the next procedure.

Let A be an (ill-conditioned) non-singular Toeplitz matrix, then apply the new iteration to the following matrix:

$$T = \frac{A^H A}{\|A^H A\|}.$$

The matrix T is Toeplitz-like, symmetric and positive definite. The new initial iteration step becomes:

$$X_0 = aTAT + bTA^2T + cTA + dT + eI.$$

Then after applying the following transformation it is possible to continue the computation with the classical modified Newton iteration:

$$X_1 = X_0 A^H.$$

4 Norm estimation

Before introducing the numerical tests it is crucial to explain how we estimated the 2-norm of a low-displacement rank matrix. Computing the 2-norm in the classical way requires a lot of computational effort, so it is necessary to have an estimation of it. In the Newton method, this estimation becomes crucial

due to the necessity of normalization of the matrix in our new iteration. With the circulant displacement representation it has been possible to define an efficient and robust 2-norm overestimation:

Theorem 2 *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 and D are defined in (2.2).

PROOF. The matrix T can be written as $\frac{1}{2} \sum_{i=1}^k C^+(\mathbf{u}_i)C^-(J\mathbf{v}_i)$ (see Th. 1). Hence, from (2.2), we derive that

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

□

5 Numerical tests

5.1 Implementation

The kernel of the implementation is the computation of the displacement representation of the product or the sum of two matrices when each of their displacement representations is known. We shall give the results only for the Δ^+ operator because they are very similar for the Δ^- operator. From Th. 1, we know that

$$\Delta^+(AB) = \Delta^+(A)B + A\Delta^+(B) - 2A\mathbf{e}_1\mathbf{e}_n^T B.$$

Hence, we can easily prove the following result.

Proposition 3 *Let $\Delta^+(A) = U_A \Sigma_A V_A^H$ and $\Delta^+(B) = U_B \Sigma_B V_B^H$ be the singular value decompositions of $\Delta^+(A)$ and $\Delta^+(B)$, respectively, with $\text{drk}(A) =$*

k_1 and $\text{drk}(B) = k_2$, then the singular value decomposition of $\Delta^+(AB) = U_{AB}\Sigma_{AB}V_{AB}^H$ can be computed in $\mathcal{O}(k_1k_2n \log n)$ ops.

PROOF. We know that $\Delta^+(AB) = \Delta^+(A)B + A\Delta^+(B) - 2Ae_1e_n^TB$ so $\Delta^+(AB) = U_A\Sigma_A V_A^H B + AU_B\Sigma_B V_B^H - 2Ae_1e_n^TB$. Hence, it can be rewritten in matrix form:

$$\Delta^+(AB) = \begin{bmatrix} U_A & AU_B & Ae_1 \end{bmatrix} \begin{bmatrix} \Sigma_A & & \\ & \Sigma_B & \\ & & -2 \end{bmatrix} \begin{bmatrix} V_A^H B \\ V_B^H \\ e_n^H B \end{bmatrix}.$$

The multiplication AU_B costs $\mathcal{O}(k_1k_2n \log n)$ ops and the same for $V_A^H B$. Apply now the following steps in order to obtain the thesis:

First: $Q_l R_l = \text{QR} \left(\begin{bmatrix} U_A & AU_B & Ae_1 \end{bmatrix} \right)$ **cost** $\mathcal{O}((k_1 + k_2 + 1)^2 n)$

Second: $Q_r R_r = \text{QR} \left(\begin{bmatrix} V_A^H B \\ V_B^H \\ e_n^H B \end{bmatrix} \right)$ **cost** $\mathcal{O}((k_1 + k_2 + 1)^2 n)$

Third: $L\Sigma_{AB}R^H = \text{SVD} \left(R_l \begin{bmatrix} \Sigma_A & & \\ & \Sigma_B & \\ & & -2 \end{bmatrix} R_r^H \right)$ **cost** $\mathcal{O}((k_1 + k_2 + 1)^3)$

Fourth: $U_{AB} = Q_l * L; V_{AB}^H = R^H * Q_r^H;$ **cost** $\mathcal{O}((k_1 + k_2 + 1)^2 n)$

Total asymptotic cost $\mathcal{O}(k_1k_2n \log n)$

Here we have used QR to denote that the QR-factorization is computed. Similarly, SVD denotes the computation of the (economical) form of the singular value decomposition. \square

Similarly, we have the following result for the sum.

Proposition 4 Let $\Delta^+(A) = U_A\Sigma_A V_A^H$ and $\Delta^+(B) = U_B\Sigma_B V_B^H$ with $\text{drk}(A) = k_1$ and $\text{drk}(B) = k_2$, then the $\Delta^+(A+B) = U_{A+B}\Sigma_{A+B}V_{A+B}^H$ can be computed in $\mathcal{O}(k_1k_2n \log n)$ ops.

We have designed a database of testing matrices defined in the following way: take a random vector which represents the first column of a symmetric Toeplitz matrix, approximate the biggest and the smallest eigenvalues, using for example the Arnoldi method. Then, shift the first element of the vector, and so

the diagonal of the matrix, so that the matrix becomes positive-definite, and parametrize it to have the possibility to obtain an arbitrary condition number. These matrices were generated using Matlab¹. Then, we have defined a library based on the *ATLAS* package², the *FFTW* package³, and the *LAPACK* package⁴. Everything is written in C++ and Object Oriented. We modeled a Toeplitz-like matrix with an object. Using the two operators described in Prop. 3 and 4 we designed the new initial iteration step and the modified Newton iteration.

5.2 Numerical experiments

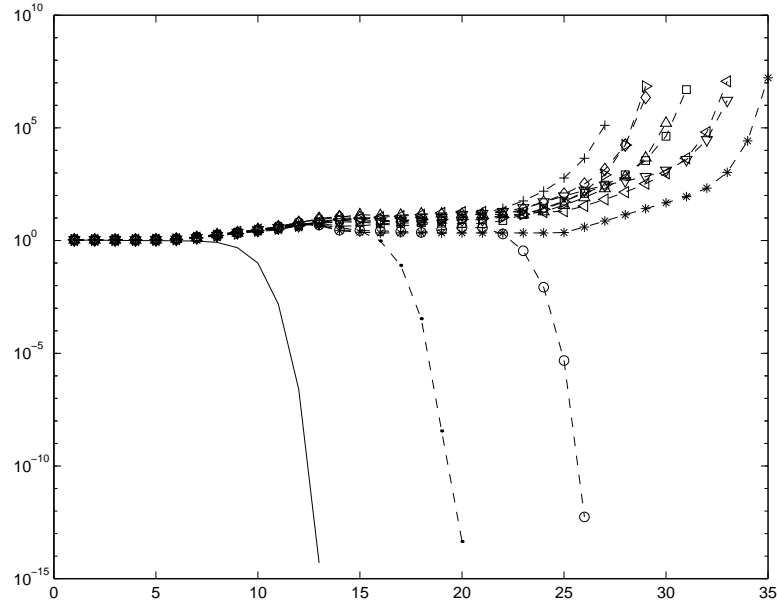


Figure 5. The norm of the residual matrix in function of the iteration step without the new initial iteration step

We want to illustrate here the improvement of the result when using the new initial iteration step compared to the initial step from the modified Newton iteration. In Fig. 5 you see the behaviour of the residual norm for a positive definite random Toeplitz when the modified Newton iteration is applied without the new initial step. The graph clearly shows that, when the condition number of the matrix grows, the method starts to diverge. When the new initial step is used, however, the convergence is preserved as is shown in Figure 6. It is also interesting to observe that all the samples tested with the same condition numbers, but having a different dimension are computed using

¹ Matlab is a registered trademark of The MathWorks.

² url: <http://math-atlas.sourceforge.net/>

³ url: <http://www.fftw.org/>

⁴ url: <http://www.netlib.org/lapack/>

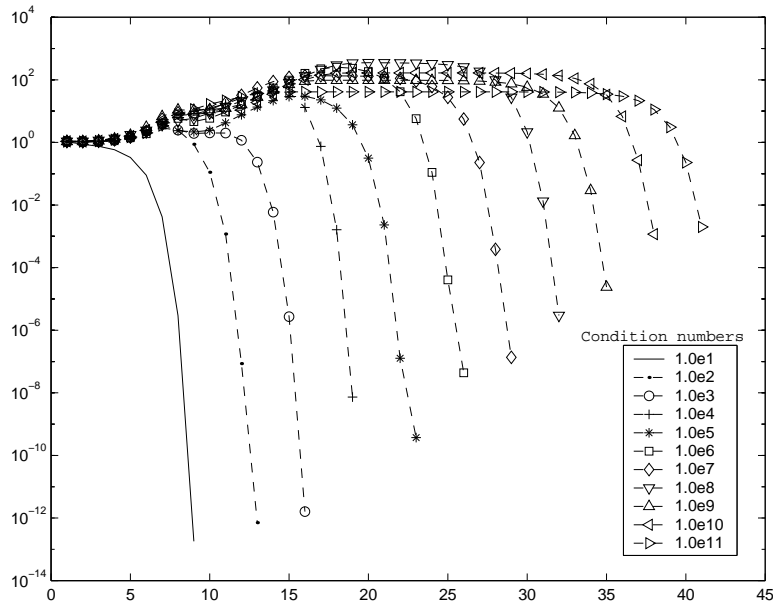


Figure 6. The norm of the residual matrix in function of the iteration step using the new initial iteration step

the same number of iteration steps. This allows to make a good prediction of the number of iteration steps when a good estimate is known for the condition number of the matrix that you want to invert.

The last point that we want to focus on, is the impact of the new iteration on the execution time of the whole method. Fig. 7 compares the execution time of one classical iteration step to the time of the new initial step and one classical iteration step.

6 Conclusion

In this paper, we have designed an alternative initial iteration step for the modified Newton method resulting in a method which is less critically dependent on the choice of the starting matrix. We have illustrated the validity of the adapted method by some numerical experiments.

Acknowledgments

The authors want to thank Prof. Dario Bini and Prof. Victor Pan for the helpful discussions concerning the modified Newton iteration method for structured matrices.

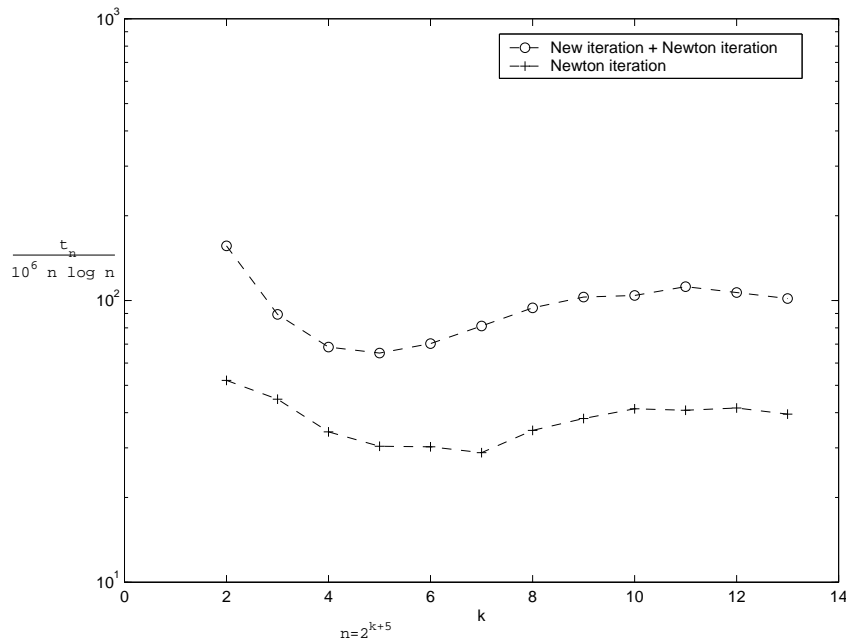


Figure 7. Timing comparison of the iterations tested on several dimension

References

- [1] K.E. Atkinson. *An introduction to numerical analysis*. John Wiley, New York, 1978.
- [2] D.A. Bini, G. Codevico, and M. Van Barel. Solving Toeplitz least squares problems by means of Newton's iteration. Report TW 333, Department of Computer Science, K.U.Leuven, Leuven, Belgium, December 2001.
- [3] D.A. Bini and B. Meini. Approximate displacement rank and applications. In V. Olshevsky, editor, *Structured matrices in mathematics, computer science, and engineering II*, volume 281 of *Contemporary Mathematics*, pages 215–232. American Mathematical Society, Providence, Rhode Island, 2001.
- [4] D.A. Bini and V. Pan. Improved parallel computations with Toeplitz-like and Hankel matrices. *Linear Algebra and its Applications*, 188,189:3–29, 1993.
- [5] D.A. Bini and V. Pan. *Matrix and polynomial computations, vol. 1: Fundamental algorithms*. Birkäuser, Boston, 1994.
- [6] I. Gohberg and V. Olshevsky. Circulant displacement and decomposition of matrices. *Integral Equations Operator Theory*, 15:730–743, 1992.
- [7] G. Heinig and K. Rost. Algebraic methods for Toeplitz-like matrices and operators. *Akademie-Verlag, Berlin, and Birkhäuser, Basel/Stuttgart*, 1984.
- [8] T. Kailath, S. Kung, and M. Morf. Displacement ranks of a matrix. *Bulletin of the American Mathematical Society*, 1:769–773, 1979.

- [9] T. Kailath, S. Kung, and M. Morf. Displacement ranks of matrices and linear equations. *J. Math. Anal. Appl.*, 68:395–407, 1979.
- [10] T. Kailath and A. Sayed. Displacement structure: Theory and applications. *SIAM Review*, 37(3):297–386, September 1995.
- [11] T. Kailath and A. Sayed, editors. *Fast reliable algorithms for matrices with structure*. SIAM, 1999.
- [12] T. Kailath, A. Vieira, and M. Morf. Inverses of Toeplitz operators, innovations and orthogonal polynomials. *SIAM Rev.*, 20:106–119, 1978.
- [13] V.Y. Pan. Computations with dense structured matrices. *Math. Comp.*, 55:179–190, 1990.
- [14] V.Y. Pan. Parallel solution of Toeplitz-like linear systems. *J. Complexity*, 8:1–21, 1992.
- [15] V.Y. Pan. Parametrization of Newton’s iteration for computation with structured matrices and applications. *Comput. Math.*, 24:61–75, 1992.
- [16] V.Y. Pan. Concurrent iterative algorithms for Toeplitz-like linear systems. *IEEE Trans. Parallel Distributive Systems*, 4(5):592–600, 1993.
- [17] V.Y. Pan. Decreasing the displacement rank of a matrix. *SIAM J. Matrix Anal. Appl.*, 14:118–121, 1993.
- [18] V.Y. Pan. *Structured matrices and polynomials*. Birkhäuser Springer, 2001.
- [19] V.Y. Pan, S. Branham, R.E. Rosholt, and Ai-Long Zheng. Newton’s iteration for structured matrices. In T. Kailath and A. Sayed, editors, *Fast reliable algorithms for matrices with structure*, pages 189–210. SIAM, 1999.
- [20] V.Y. Pan and R. Schreiber. An Improved Newton iteration for the generalized inverse of a matrix with applications. *SIAM J. Sci. Statist. Comput.*, 12:1109–1131, 1991.
- [21] V.Y. Pan, A.L. Zheng, X.H. Huang, and O. Dias. Newton’s iteration for inversion of Cauchy-like and other structured matrices. *J. Complexity*, 13:108–124, 1997.
- [22] G. Schultz. Iterative Berechnung dar Reziproken Matrix. *Z. Angew. Math. Mech.*, 13:57–59, 1933.