Stability Analysis and Fast Algorithms for Triangularization of Toeplitz Matrices

Haesun Park* and Lars Eldén†

August 4, 1997

Abstract

A new $O(mn)$ algorithm for triangularizing an $m \times n$ Toeplitz matrix is presented. The algorithm is based on the previously developed recursive algorithms that exploit the Toeplitz structure and compute each row of the triangular factor via updating and downdating steps. A forward error analysis for this existing recursive algorithm is presented, which allows us to monitor the conditioning of the problem, and use the method of corrected semi-normal equations to obtain higher accuracy for certain ill-conditioned matrices. Numerical experiments show that the new algorithm improves the accuracy significantly while the computational complexity stays in $O(mn)$.

Keywords: Cholesky decomposition, corrected seminormal equations, downdating, forward error analysis, QR decomposition, stabilized hyperbolic transformation, Toeplitz matrix, updating

*Computer Science Department, University of Minnesota, Minneapolis, MN 55455, U.S.A. (e-mail: hpark@cs.umn.edu). The work of this author was supported in part by the National Science Foundation grant CCR-9209726. This paper has appeared in Numer. Math. 1997, 76:383-402

†Department of Mathematics, Linköping University, S–581 83 Linköping, Sweden, (e-mail: laeld@math.liu.se).

1 Introduction

Let $T \in \mathbf{R}^{m\times n}$ be a Toeplitz matrix with $m \geq n$,

$$T = \begin{pmatrix} t_0 & t_{-1} & & \cdots & t_{-n+1} \\ t_1 & t_0 & t_{-1} & & \vdots \\ \vdots & t_1 & t_0 & \ddots \vdots & \\ \vdots & & t_1 & \ddots & t_{-1} \\ \vdots & & & \ddots & t_0 \\ \vdots & & & & t_1 \\ \vdots & & & & \vdots \\ t_{m-1} & & & & t_{m-n} \end{pmatrix}, \tag{1}$$

and assume that its QR decomposition is

$$T = QR, \tag{2}$$

where $Q \in \mathbf{R}^{m\times n}$ has orthonormal columns and $R \in \mathbf{R}^{n\times n}$ is upper triangular. It is well known that computing the QR decomposition of a general $m \times n$ matrix requires $O(mn^2)$ floating point operations (flops) [15]. There are several algorithms developed for computing the QR decomposition of a Toeplitz matrix using its special structure that all the elements on each diagonal are the same. These algorithms require $O(mn + n^2)$ flops, and they are referred to as *fast* algorithms for Toeplitz QR decomposition [6, 9, 10, 11, 22, 23, 30]. However, the stability properties of these fast algorithms are not well understood [7], and most of the fast algorithms suffer from loss of accuracy when they are applied to certain classes of matrices. The accuracy we need to obtain may differ depending on the applications: in least squares problems, we want to obtain accurate solution vectors, and in some other applications we may want the accurate upper triangular factor R. For instance, an accurate R factor is needed in the preprocessing QR decomposition for computing the singular value decomposition, and in some least squares linear prediction problems, where the augmented matrix formed by attaching the right hand side vector to the data matrix preserves the Toeplitz structure.

In this paper, we present a new algorithm that produces a more accurate R factor in the QR decomposition of a Toeplitz matrix than other existing fast methods in $O(mn+n^2)$ flops. The new algorithm is based on the fast algorithms introduced by Bojanczyk, Brent, and de Hoog [6], and also Chun, Kailath, and Lev-Ari [9]. In fact, it is shown in [9] that these two algorithms are essentially equivalent and they are generalizations of a classical algorithm by Schur [25, 18]. In this generalized Schur algorithm, each row of the triangular factor is computed recursively via updating and downdating steps. We present a forward error analysis for the algorithm and show that the accuracy of the upper triangular factor depends on the square of the condition number of the given Toeplitz matrix. The downdating steps in this generalized Schur algorithm are mostly responsible for the loss of accuracy. In our fast algorithm for the triangularization of Toeplitz matrices, we adopt our recent results on Cholesky and QR decomposition downdating problems [14]. As a result, the new algorithm produces more accurate triangular factors than other existing fast algorithms, especially for ill-conditioned problems. Numerical examples presented in Section 5 support this claim.

2 Toeplitz Matrix Triangularization

2.1 A Recursive Algorithm

We first review the derivation of the recursive algorithm for Toeplitz matrix triangularization due to [6]. In the next subsection, an alternative derivation of the same algorithm based on the concept of displacement structure [9, 18] is presented.

Using the Toeplitz structure, T can be partitioned into

$$T = \begin{pmatrix} t_0 & f_r^T \\ f_c & T_0 \end{pmatrix} = \begin{pmatrix} T_0 & l_c \\ l_r^T & t_{m-n} \end{pmatrix}, \tag{3}$$

where $T_0 \in \mathbf{R}^{(m-1)\times(n-1)}$ is a submatrix of $T \in \mathbf{R}^{m\times n}$, f_r, $l_r \in \mathbf{R}^{(n-1)\times 1}$, and f_c, $l_c \in \mathbf{R}^{(m-1)\times 1}$. Also, the upper triangular factor R in the QR decomposition (2) of T, which we denote as

$$R = qr(T),$$

is partitioned into

$$R = \begin{pmatrix} r_{11} & r_r^T \\ 0 & R_b \end{pmatrix} = \begin{pmatrix} R_t & r_c \\ 0 & r_{nn} \end{pmatrix}, \tag{4}$$

where r_r, $r_c \in \mathbf{R}^{(n-1)\times 1}$, and r_{11} and r_{nn} are scalars. From (3) and (4), $R^TR = T^TT$ becomes

$$\begin{pmatrix} r_{11}^2 & r_{11}r_r^T \\ r_{11}r_r & r_rr_r^T + R_b^TR_b \end{pmatrix} = \begin{pmatrix} t_0^2 + f_c^Tf_c & t_0f_r^T + f_c^TT_0 \\ t_0f_r + T_0^Tf_c & f_rf_r^T + T_0^TT_0 \end{pmatrix}, \tag{5}$$

and

$$\begin{pmatrix} R_t^TR_t & R_t^Tr_c \\ r_c^TR_t & r_c^Tr_c + r_{nn}^2 \end{pmatrix} = \begin{pmatrix} T_0^TT_0 + l_rl_r^T & T_o^Tl_c + t_{m-n}l_r \\ l_c^TT_0 + t_{m-n}l_r^T & l_c^Tl_c + t_{m-n}^2 \end{pmatrix}. \tag{6}$$

From (5) and (6), one can obtain

$$R_b^TR_b = R_t^TR_t + f_rf_r^T - l_rl_r^T - r_rr_r^T,$$

which we separate into two parts

$$\begin{aligned} R_1^TR_1 &= R_t^TR_t + f_rf_r^T, &(7)\\ R_b^TR_b &= R_1^TR_1 - (l_rl_r^T + r_rr_r^T). &(8) \end{aligned}$$

A recursive algorithm, where the rows of R_b are computed one by one using Givens and hyperbolic transformations, can be derived from (7) and (8). Extensive derivations are given in [6, 22]. For Givens and hyperbolic transformations, see [15] and also [1, 6, 9].

Before stating the algorithm, we briefly discuss *stabilized hyperbolic transformations.* Given the vector $(x\ y)^T$, where $|x| > |y|$, a hyperbolic cosine and hyperbolic sine pair $[ch,\ sh]$ that annihilates the second component of the vector as

$$\begin{pmatrix} ch & -sh \\ -sh & ch \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sqrt{x^2 - y^2} \\ 0 \end{pmatrix} \tag{9}$$

can be obtained from $ch = x/\sqrt{x^2 - y^2}$ and $sh = y/\sqrt{x^2 - y^2}$. This hyperbolic transformation can be factored into two transformations as

$$\begin{pmatrix} ch & -sh \\ -sh & ch \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -th & 1/ch \end{pmatrix} \begin{pmatrix} ch & -sh \\ 0 & 1 \end{pmatrix}, \tag{10}$$

where $th = sh/ch$. Since $th^2 + (1/ch)^2 = 1$, $|th| \leq 1$ and $|1/ch| \leq 1$, we can set $th = \cos\eta$ and $1/ch = \sin\eta$ for some angle η. In fact, when $(x\ y)^T$ is given, we can compute

$$c = \cos\eta = y/x \qquad \text{and} \qquad s = \sin\eta = \sqrt{x^2 - y^2}/x. \tag{11}$$

Then using (10) and (11), the updating of v_1 and v_2 in

$$\begin{pmatrix} v_1' \\ v_2' \end{pmatrix} := \begin{pmatrix} ch & -sh \\ -sh & ch \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \tag{12}$$

can be performed as

$$v_1' := v_1 * ch - v_2 * sh = v_1/\sin\eta - v_2 * \cos\eta/\sin\eta, \qquad v_2' := -v_1' * \cos\eta + v_2 * \sin\eta. \tag{13}$$

The new v_2' is like the result of applying a plane rotation to the vector $\begin{pmatrix} v_1' & v_2 \end{pmatrix}^T$. This is called a stabilized hyperbolic transformation and for better stability [1, 5], it is important to use (13) instead of the straight forward updating as

$$v_1' := v_1 * ch - v_2 * sh, \qquad v_2' := -v_1 * sh + v_2 * ch. \tag{14}$$

If the new first component v_1' is to be computed like a result of applying a plane rotation, the hyperbolic transformation can be factored into

$$\begin{pmatrix} ch & -sh \\ -sh & ch \end{pmatrix} = \begin{pmatrix} 1/ch & -th \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -sh & ch \end{pmatrix}. \tag{15}$$

In Chambers' downdating algorithm [8], the stabilized hyperbolic transformation is used as a transformation to 'undo' the updating. In [27], it is shown that the downdating of the upper triangular factor via hyperbolic transformation as in (14) is not forward nor backward stable, whereas transformation with (13) is *relationally stable.* In [1, 5], similar

Algorithm A:
Given a Toeplitz matrix $T \in \mathbf{R}^{m\times n}$ as in (1) which is partitioned as in (3), this algorithm computes the upper triangular factor R for T.

1. $R(1,1) := (t_0^2 + f_c^T f_c)^{1/2}$; $\quad r_r := (t_0 f_r + T_0^T f_c)/R(1,1)$; $\quad R(1,2:n) := r_r^T$
2. **for** $k = 1 : n-1$
   (a) $r_t(k:n-1) := R(k, k:n-1)$
   (b) $[r_1(k),\ c,\ s] := \mathrm{givens}(r_t(k), f_r(k))$
   $$\begin{pmatrix} r_1^T(k+1:n-1) \\ f_r^T(k+1:n-1) \end{pmatrix} := \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} r_t^T(k+1:n-1) \\ f_r^T(k+1:n-1) \end{pmatrix}$$
   (c) $[l_r(k),\ c,\ s] := \mathrm{givens}(l_r(k), r_r(k))$
   $$\begin{pmatrix} l_r^T(k+1:n-1) \\ r_r^T(k+1:n-1) \end{pmatrix} := \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} l_r^T(k+1:n-1) \\ r_r^T(k+1:n-1) \end{pmatrix}$$
   (d) $[r_b(k),\ c,\ s] := \mathrm{shyp}(r_1(k), l_r(k))$
   $$\begin{pmatrix} r_b^T(k+1:n-1) \\ l_r^T(k+1:n-1) \end{pmatrix} := \begin{pmatrix} 1 & 0 \\ -c & s \end{pmatrix} \begin{pmatrix} 1/s & -c/s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} r_1^T(k+1:n-1) \\ l_r^T(k+1:n-1) \end{pmatrix}$$
   (e) $R(k+1, k+1:n) := r_b(k:n-1)$

   end

results are presented. It is also important to use the stabilized hyperbolic transformation (10) in downdating steps in the recursive Toeplitz matrix triangularization algorithm for better stability.

The recursive algorithm is summarized in Algorithm A. Here, given a vector with two components $(x\ y)^T$, the function $givens(x,\ y)$ returns $\sqrt{x^2+y^2}$ and the cosine and sine pair $[c,\ s]$, such that
$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sqrt{x^2+y^2} \\ 0 \end{pmatrix},$$
and the function $shyp(x,\ y)$ returns $\sqrt{x^2-y^2}$ and the cosine and sine pair $[c,\ s]$ of the stabilized hyperbolic transformation as in (11), and we apply the transformation as in (13). Algorithm A requires $mn + 6n^2 + O(n)$ flops (1 flop $\approx$ 1 multiplication and 1 addition) to compute R. Unlike the algorithm in [6], we use only one hyperbolic transformation in each step instead of two, cf. [9].

The row vectors r_1 computed in Step 2b, are the rows of the matrix R_1. This will be used in Section 4.2 in improving the accuracy of Algorithm A.

2.2 Alternative Representation based on Displacement Structure

The error analysis of Algorithm A that we will present in the next section is based on the formulation for computing the upper triangular factor R of the matrix T as a block updating and downdating problem. For a given row vector $w = (\, w_1 \quad w_2 \quad w_3 \quad \cdots \quad w_n \,) \in \mathbf{R}^{1\times n}$, we define $\mathrm{up}(w) \in \mathbf{R}^{n\times n}$ as the upper triangular Toeplitz matrix where its first row is w, i.e.,

$$\mathrm{up}(w) = \begin{pmatrix} w_1 & w_2 & w_3 & \dots & w_n \\ & w_1 & w_2 & \dots & w_{n-1} \\ & & w_1 & \dots & w_{n-2} \\ & & & \ddots & \vdots \\ & & & & w_1 \end{pmatrix}.$$

Now we present the result that relates Algorithm A to a block updating and downdating problem.

Proposition 2.1 *Let $T \in \mathbf{R}^{m\times n}$ be a Toeplitz matrix with $m \geq n$. Then T has the displacement representation*

$$T^T T = U_1^T U_1 + U_2^T U_2 - U_3^T U_3 - U_4^T U_4, \tag{16}$$

where

$$U_1 = \mathrm{up}([r_{11}\ r_r^T]), \quad U_2 = \mathrm{up}([0\ f_r^T]), \quad U_3 = \mathrm{up}([0\ l_r^T]), \quad U_4 = \mathrm{up}([0\ r_r^T]).$$

The proof of the proposition can be found in [9, Lemmas 1 and 2].

We next describe an alternative formulation of Algorithm A [9, 22] since this formulation allows us to use the existing results in error analysis more readily. In Algorithm A, the first row of R is computed in Step 1. This gives the elements of the matrices U_1 and U_4. The matrices U_2 and U_3 are defined by parts of the first and last rows of T, respectively. Then the Cholesky factor R for $T^T T$ can be computed by transforming

$$U \equiv \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} \in \mathbf{R}^{4n\times 4n}, \tag{17}$$

to an upper triangular matrix by updating U_2 and downdating U_3 and U_4 to U_1.

For example, when $m = 8$ and $n = 4$, $U \in \mathbf{R}^{16\times 4}$ consists of four Toeplitz upper triangular matrices $U_i \in \mathbf{R}^{4\times 4}$, $1 \leq i \leq 4$. First, we apply Givens rotations G_1 and G_2 in the (2,5) and the (9,13) planes, that annihilate t_{-1} and r_{12}, respectively. Due to the Toeplitz structure of each U_i, the same rotations G_1 and G_2 annihilate the entire first

$$U = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ & r_{11} & r_{12} & r_{13} \\ & & r_{11} & r_{12} \\ & & & r_{11} \\ 0 & t_{-1} & t_{-2} & t_{-3} \\ & 0 & t_{-1} & t_{-2} \\ & & 0 & t_{-1} \\ & & & 0 \\ 0 & t_7 & t_6 & t_5 \\ & 0 & t_7 & t_6 \\ & & 0 & t_7 \\ & & & 0 \\ 0 & r_{12} & r_{13} & r_{14} \\ & 0 & r_{12} & r_{13} \\ & & 0 & r_{12} \\ & & & 0 \end{pmatrix} \to U' = \begin{pmatrix} U_1' \\ U_2' \\ U_3' \\ U_4' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ & r_{11}' & r_{12}' & r_{13}' \\ & & r_{11}' & r_{12}' \\ & & & r_{11}' \\ & 0 & t_{-2}' & t_{-3}' \\ & & 0 & t_{-2}' \\ & & & 0 \\ & & & 0 \\ & 0 & t_6' & t_5' \\ & & 0 & t_6' \\ & & & 0 \\ & & & 0 \\ & 0 & r_{13}' & r_{14}' \\ & & 0 & r_{13}' \\ & & & 0 \\ & & & 0 \end{pmatrix}$$

Figure 1: The first superdiagonals in U_2, U_3 and U_4, are annihilated.

superdiagonals in U_2 and U_4, respectively, when applied in appropriate planes, without any extra computation. Next, by a stabilized hyperbolic transformation H_1 in the (2,9) plane, we can annihilate the modified element in the location of t_7, and annihilate the entire first superdiagonal of the third block without any extra computation. After these steps, we obtain U' as shown in Figure 1. Due to the structures of U_2', U_3', and U_4', the second row of U_1' will not be modified any further and this becomes the second row of the final matrix R. So far, we have performed the same transformations as in the first iteration of the loop in Step 2 of Algorithm A. Since all U_i' have the necessary Toeplitz structure, we can continue the same procedure, now working with the 2×2 submatrices.

In general, after $n-1$ such steps we have

$$\Theta U = \begin{pmatrix} R \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{18}$$

where $\Theta \in \mathbf{R}^{4n \times 4n}$ is a product of all the Givens and stabilized hyperbolic transformations involved. Also, since Θ satisfies

$$\Theta^T \Lambda \Theta = \Lambda, \qquad \Lambda = \begin{pmatrix} I_{2n} & 0 \\ 0 & -I_{2n} \end{pmatrix},$$

where I_{2n} denotes the $2n \times 2n$ identity matrix, we have $R^T R = T^T T$. This formulation is a way of presenting Algorithm A as a non-recursive algorithm. Since the two algorithms are equivalent, the computational complexities are the same and the error analyses will be essentially the same.

3 Error Analysis

From Section 2.2, we see that Algorithm A consists of block updating of U_2 and block downdating of U_3 and U_4 to the matrix U_1, i.e.,

$$R^T R = U_1^T U_1 + U_2^T U_2 - U_3^T U_3 - U_4^T U_4. \tag{19}$$

Our error analysis is based on the recent result of Stewart [27] on the stability of sequential updates and downdates.

Consider for the moment the Cholesky block updating and downdating,

$$S^T S = S_1^T S_1 + S_2^T S_2 - S_3^T S_3,$$

where the matrices S_2 and $S_3 \in \mathbf{R}^{n\times n}$, and the triangular matrix $S_1 \in \mathbf{R}^{n\times n}$ are given. We assume that orthogonal transformations are used to perform the block updating of S_2 and *stabilized* hyperbolic transformations are used for the block downdating of S_3. Let $\bar{S}$ denote the computed value for S in floating point arithmetic with the unit round off μ. Then, according to [27] (cf. also [5]), there exists an orthogonal matrix J such that

$$J \begin{pmatrix} S_1 \\ 0 \\ S_2 \end{pmatrix} = \begin{pmatrix} \bar{S} \\ S_3 \\ 0 \end{pmatrix} + E, \qquad \|E\|_F \le C\mu\rho_1,$$

where the norm is the Frobenius norm, C is a low degree polynomial in n, and ρ_1 is bounded by

$$\rho_1 \le \| \begin{pmatrix} S_1 & S_2 \end{pmatrix} \|_F.$$

Throughout this paper, we use C to denote any low degree polynomial in n. Each occurrence of C may represent a different such value. Applying [27, Eq. (9)] or the results in [14], we obtain the following forward error estimate

$$\frac{\|\bar{S} - S\|_F}{\|S\|_F} \le C\mu\rho_1^2 \|S^{-1}\|_F^2 + O(\mu^2). \tag{20}$$

From (20) we see that *if S is well-conditioned and ρ_1 is not large, then the forward error in the computed $\bar{S}$ is small.* The perturbation theory that is the basis of this error estimate can be refined, and tighter bounds can be obtained using the results in [28] and [14]. In our error analysis for Algorithm A, we will not try to obtain the sharpest possible error estimate, but rather expose the causes of numerical instability, with the aim of modifying the algorithm for enhanced stability. Therefore, instead of analyzing the algorithm based on the recursive formulation in Section 2.1, we will apply the analysis of Stewart to the block updating and downdating (19).

In the first step of Algorithm A, we compute r_{11} and r_r from

$$r_{11} := (t_0^2 + f_c^T f_c)^{1/2}, \qquad r_r := \frac{1}{r_{11}} \begin{pmatrix} T_0^T & f_r \end{pmatrix} \begin{pmatrix} f_c \\ t_0 \end{pmatrix}.$$

The computed $\bar{r}_{11}$ satisfies (see e.g. [20, p. 85])

$$\bar{r}_{11} = r_{11}(1+\alpha), \qquad |\alpha| \le C\mu + O(\mu^2). \tag{21}$$

Similarly, the computed $\bar{r}_r$ satisfies [15, p. 66],

$$\bar{r}_r = r_r + e_r, \tag{22}$$

where

$$\|e_r\|_F \le \frac{C\mu}{|r_{11}|} \| \begin{pmatrix} T_0^T & f_r \end{pmatrix} \|_F \, \| \begin{pmatrix} f_c \\ t_0 \end{pmatrix} \|_F + O(\mu^2) \le C\mu \| \begin{pmatrix} T_0^T & f_r \end{pmatrix} \|_F + O(\mu^2), \tag{23}$$

since $\| \begin{pmatrix} f_c \\ t_0 \end{pmatrix} \|_F / |r_{11}| = 1$. Furthermore, since the upper triangular Toeplitz matrices U_1 and U_4 are defined as

$$U_1 = \mathrm{up}([r_{11}\ \ r_r^T]), \qquad U_4 = \mathrm{up}([0\ \ r_r^T]),$$

from (21)–(23), the computed $\bar{U}_1$ and $\bar{U}_4$ satisfy

$$\bar{U}_1 = U_1 + E_1, \qquad \bar{U}_4 = U_4 + E_4, \qquad \|E_i\|_F \le C\mu\|T\|_F + O(\mu^2), \quad i = 1, 4. \tag{24}$$

Let $\bar{R}$ denote R computed using Algorithm A in floating point arithmetic. Combining the results presented in this section, we see that there exists an orthogonal matrix J such that

$$J \begin{pmatrix} U_1 + E_1 \\ U_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \bar{R} \\ 0 \\ U_3 \\ U_4 + E_4 \end{pmatrix} + E_2, \qquad \|E_2\|_F \le C\mu\rho_2,$$

where $\rho_2 \le \| \begin{pmatrix} U_1 + E_1 & U_2 \end{pmatrix} \|_F$. This can be rewritten as

$$J \begin{pmatrix} U_1 \\ U_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \bar{R} \\ 0 \\ U_3 \\ U_4 \end{pmatrix} + E_3, \qquad \|E_3\|_F \le C\mu\rho_3,$$

where $\rho_3 \le \max\{\| \begin{pmatrix} U_1 & U_2 \end{pmatrix} \|_F, \|T\|_F\}$. Since the elements of U_1 and U_2 are from R and T, respectively, and since $\|R\|_F = \|T\|_F$, we can estimate

$$\| \begin{pmatrix} U_1 & U_2 \end{pmatrix} \|_F \le C\|T\|_F. \tag{25}$$

Thus, we have

$$\|E_3\|_F \leq C\mu\rho_4, \qquad \rho_4 \leq \|T\|_F.$$

Using the technique in [27], we get the following bound for the forward relative error in $\bar{R}$:

$$\frac{\|\bar{R}-R\|_F}{\|R\|_F} \leq C\mu\rho_4^2\|R^{-1}\|_F^2 + O(\mu^2) \leq C\mu\kappa^2(T) + O(\mu^2), \tag{26}$$

where $\kappa(T) = \|R\|_F\|R^{-1}\|_F$. From (26) we see that the forward relative error in $\bar{R}$ is bounded by $C\mu\kappa(T^TT)$. This is comparable to the bound for the forward error for the computation of the Cholesky factor of T^TT (which can be obtained by combining the error analysis for the Cholesky decomposition algorithm with the perturbation theory for the decomposition, see e.g. [16, Chap. 10] and [29]). In contrast, it can be shown (using the results in [20, Chap. 15] and [26, Theorem 3.1] or [29, Theorem 1.6]; see also [16, Chap. 18]) that the forward error for computing R using orthogonal transformations, is bounded essentially by $C\mu\kappa(T)$.

Thus, even if the upper triangular factor R is computed in Algorithm A without forming T^TT explicitly, *the error in the computed $\bar{R}$ may be as large as if it had been computed as the Cholesky factor of T^TT*. From this point of view, Algorithm A computes the Cholesky factor of T^TT rather than the upper triangular factor from the QR decomposition of T.

From (26) we can conclude that if T is well-conditioned then R is accurately computed by Algorithm A. This is the basis of our improvements of Algorithm A, which is presented in Section 4.2: *instead of computing an ill-conditioned R directly, we first compute an intermediate upper triangular matrix, which is more well-conditioned. Then, we perform the downdates by a more accurate downdating method.*

4 Improving the Accuracy of R

4.1 An Improved Downdate

It is well-known that when a matrix is triangularized using orthogonal transformations applied from the left, the norms of the column vectors of the matrix stay constant during the procedure. This is computationally not the case in Algorithm A, as is demonstrated below. Consider the computation of the hyperbolic transformation (9) using the function shyp when $|x|$ is only slightly larger than $|y|$. Then cancellation occurs in the computation of $\sqrt{x^2-y^2}$ and the computed sh may have large relative error, which will lead to large errors in R. We now show a simple example where this happens and demonstrate how it can be avoided using an alternative way of computing the downdating in Step 2(d) of Algorithm A.

Example. Let

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \tau & 0 & 0 \end{pmatrix},$$

where τ is large. The R factor in the QR decomposition of T is

$$R = \begin{pmatrix} \sqrt{\tau^2+1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

From the displacement representation (16), the matrix U for T can be written as

$$U = \begin{pmatrix} U_1 \\ U_3 \end{pmatrix} = \begin{pmatrix} \sqrt{\tau^2+1} & 0 & 0 \\ & \sqrt{\tau^2+1} & 0 \\ & & \sqrt{\tau^2+1} \\ 0 & \tau & 0 \\ & 0 & \tau \\ & & 0 \end{pmatrix},$$

where U_2 and U_4 are equal to zero due to the structure of the matrix T, and U_3 is to be downdated from U_1 by stabilized hyperbolic transformations. Thus, using the function shyp we compute the diagonal element r_{22} in R from

$$r_{22} := \sqrt{(\tau^2+1) - \tau^2},$$

which may lead to severe cancellation and large relative error. Also the sine value computed by shyp will have a large relative error.

On the other hand, when we are to compute the element r_{22}, we already know r_{12}. Furthermore, we can compute the norm of the second column, $T_{\cdot 2}$ of T directly from T. Therefore, alternatively we can compute r_{22} from

$$r_{22} := \sqrt{\|T_{\cdot 2}\|_2^2 - r_{12}^2}\,.$$

In this example, we get $r_{22} := \sqrt{1-0}$, which involves no cancellation.

The large element in the first column may cause large errors in the rest of the columns. By the procedure of this example, we circumvent this difficulty by basing the second hyperbolic transformation on the second column.

In general, at the kth step of Algorithm A, before the hyperbolic transformation is computed, we have already obtained the first k rows of R, i.e., $R(1:k, 1:n)$. The $(k+1)$th row of R is computed by the stabilized hyperbolic transformation in Step 2(d) and the diagonal element of R is computed from

$$r_{k+1,k+1} := \sqrt{r_1(k)^2 - l_r(k)^2}. \tag{27}$$

---

function $[a,\ c,\ s] = \text{modhyp}(x_1, y_1, x_2, y_2)$

$c := y_1/x_1; \qquad temp := y_2/x_2$

if $|c| < |temp|$ **then**

$a := \sqrt{x_1^2 - y_1^2};\ s := a/x_1;$

else

$a := \sqrt{x_2^2 - y_2^2};\ s := a/x_1;$

end

On the other hand, since orthogonal transformations preserve Euclidean length, we have

$$\|T_{\cdot(k+1)}\|_2 = \|R_{\cdot(k+1)}\|_2,$$

where $T_{\cdot(k+1)}$ and $R_{\cdot(k+1)}$ denote the $(k+1)$th columns of T and R, respectively. Therefore, the diagonal element can also be computed from

$$r_{k+1,k+1} := \sqrt{\|T_{\cdot(k+1)}\|_2^2 - \|R_{\cdot(k+1)}(1:k)\|_2^2}\,. \tag{28}$$

We refer to the algorithm, where $\text{shyp}(r_1(k), l_r(k))$ in Step 2(d) of Algorithm A is replaced by

$$[r_b(k),\ c,\ s] := \text{modhyp}(r_1(k), l_r(k), \|T_{\cdot(k+1)}\|_2, \|R_{\cdot(k+1)}(1:k)\|_2), \tag{29}$$

as Algorithm B. This allows us to choose the expression with less cancellation between (27) and (28). The computational complexity for Algorithm B is $2mn + 6.5n^2$ flops.

4.2 Refining R using Corrected SemiNormal Equations

When R is ill-conditioned, we can expect a large error in the computed matrix $\bar{R}$ by Algorithm A according to the error estimate (26). We now show how to improve the accuracy of R by applying an accurate block downdating algorithm [14] based on Corrected SemiNormal Equations (CSNE) [4]. From (3) and (4), we know that R_t is the upper triangular factor for the first $n-1$ columns of the matrix T, i.e.,

$$R_t = qr(T(:, 1:n-1)).$$

The matrix R_t is computed by the same procedure as R in Algorithm A and Algorithm B except that the last column of T is not involved. Then R_1 is the result of updating f_r to R_t as shown in (7). Thus, applying the result presented in Section 3 (also see [27]), we know that the relative accuracy in the computed R_1 satisfies

$$\frac{\|\bar{R}_1 - R_1\|_F}{\|R_1\|_F} \le C\mu \|R_1\|_F^2 \|R_1^{-1}\|_F^2 + O(\mu^2). \tag{30}$$

Since
$$\sigma_{n-1}(R_1) \geq \sigma_{n-1}(R_t) \geq \sigma_n(R),$$
where $\sigma_i(X)$ denotes the ith singular value of a matrix X, and
$$\|R_1\|_F^2 = \| \begin{pmatrix} R_t \\ f_r^T \end{pmatrix} \|_F^2 \leq 2\|R\|_F^2,$$
we have
$$\|R_1\|_F \|R_1^{-1}\|_F \leq C\|R\|_F \|R^{-1}\|_F.$$
Therefore, if R_1 is considerably more well-conditioned than R, then we can expect that the computed R_1 is more accurate than the computed R in Algorithms A or B. If we perform the block downdating of (8) from this more accurate R_1 by using the CSNE downdating algorithm that we describe next, then we will achieve improved accuracy in the computed value for R.

Now, we first give a brief description of the CSNE method for block downdating, and then show how it can be applied to the downdating problem (8) to refine the R matrix obtained from Algorithm B. In [3, 12, 13], we showed how the method of CSNE [4] can be used to compute accurate solutions for downdating problems.

In block downdating of a QR decomposition, we need to find the QR decomposition of $\tilde{X}$,
$$\tilde{X} = \tilde{Q} \begin{pmatrix} \tilde{R}_X \\ 0 \end{pmatrix},$$
where the matrix $X \in \mathbf{R}^{m\times n}$, $m \geq n$, and its QR decomposition
$$X = \begin{pmatrix} Z \\ \tilde{X} \end{pmatrix} = Q \begin{pmatrix} R_X \\ 0 \end{pmatrix},$$
are given, and $Z \in \mathbf{R}^{k\times n}$ is the first k rows of X to be deleted. If Q were known explicitly, then we could compute $\tilde{R}_X$ by determining an orthogonal matrix $P \in \mathbf{R}^{m\times m}$, such that
$$\begin{pmatrix} I_k & Z \\ 0 & \tilde{R}_X \\ 0 & 0 \end{pmatrix} := P \begin{pmatrix} Q_1 & R_X \\ Q_2 & 0 \end{pmatrix}, \tag{31}$$
where $({Q_1}^T \; {Q_2}^T) \in \mathbf{R}^{k\times n}$ are the first k rows of Q, I_k is the $k \times k$ identity matrix, and $\tilde{R}_X \in \mathbf{R}^{n\times n}$ is upper triangular [12]. When Q is not known, which is often the case in order to avoid the computing cost and storage for Q, we can recover the information about the first k rows of Q that is necessary in determining the downdating transformation P shown in (31). This is due to the fact that Q_1 can be computed by solving the equation $R_X^T Q_1 = Z^T$, and also Q_2 can be replaced by the Cholesky factor Γ of $I_k - {Q_1}^T Q_1$, which we represent as
$$\Gamma = chol(I_k - {Q_1}^T Q_1).$$

For details, see [12]. This is a generalization of the LINPACK algorithm for downdating a single row [24].

The gist of the CSNE algorithm for downdating is that by using the *data matrix* in one step of iterative refinement, we can recover the information that was lost in computing R_X and accordingly, can recover Q_1 and Γ more accurately. Thus, it is important to identify which data matrix accounts for the upper triangular factor in applying the CSNE method. The block CSNE downdating algorithm is based on the fact that if V is the solution to

$$\min_V \|E_k - XV\|_F, \qquad E_k = \begin{pmatrix} I_k \\ 0 \end{pmatrix}, \tag{32}$$

then

$$\begin{pmatrix} R_X & Q_1 \\ 0 & \Gamma \end{pmatrix} = qr((\, X \quad E_k \,)), \tag{33}$$

where $\Gamma = qr(E_k - XV) \in \mathbf{R}^{k\times k}$ and V satisfies the equation $Q_1 = R_X V$. Applying the CSNE method for least squares problems [4] to (32), we obtain

$$\begin{aligned} R_X^T Q_1 &= (E_k^T X)^T = Z^T, \qquad R_X V = Q_1, \qquad F := E_k - XV, && (34) \\ R_X^T \delta Q_1 &= X^T F, \qquad R_X \delta V = \delta Q_1, && (35) \\ Q_1 &:= Q_1 + \delta Q_1, \qquad \Gamma = qr(F - X\delta V). && (36) \end{aligned}$$

Now, we show how the block CSNE method can be used to refine the matrix R obtained from Algorithm B. For this purpose, we rewrite (7)–(8) as

$$\begin{aligned} R_1^T R_1 &= R_t^T R_t + f_r f_r^T, && (37) \\ R_b^T R_b &= R_1^T R_1 - Z^T Z, && (38) \end{aligned}$$

where $Z^T = (\, l_r \quad r_r \,) \in \mathbf{R}^{n\times 2}$. Suppose we have R_1 from Algorithm B. From (3), (4), and (7), we have

$$R_1 = qr(X), \qquad X = \begin{pmatrix} f_r^T \\ T_0 \\ l_r^T \end{pmatrix} \in \mathbf{R}^{(m+1)\times(n-1)},$$

and the matrix R_b is obtained by downdating the block Z from R_1. Since r_r^T is not necessarily a row of X, X is not the data matrix from which the block Z is to be downdated to give the resulting factor R_b. However, note that r_r^T is part of the first row in the upper triangular factor R of the matrix T. We can find the product of a sequence of plane rotations, $\hat{W}$, which transforms the first column of T into $\|T_{\cdot 1}\| e_1$ and, accordingly, the first row f_r^T into r_r^T. Then Z consists of the first two rows of the matrix defined as

$$Y = WX = \begin{pmatrix} 0 & 1 \\ \hat{W} & 0 \end{pmatrix} \begin{pmatrix} f_r^T \\ T_0 \\ l_r^T \end{pmatrix}.$$

Now, we can apply the CSNE block downdating algorithm considering Y as the data matrix for the triangular factor R_1, which transforms the problem into

$$\min_V \|E_2 - YV\|_F, \qquad E_2 = \begin{pmatrix} I_2 \\ 0 \end{pmatrix}. \tag{39}$$

This problem is equivalent to

$$\min_V \|W_2^T - XV\|_F, \qquad W_2 = E_2^T W, \tag{40}$$

since $\|E_2 - YV\|_F = \|E_2 - WXV\|_F = \|W^T E_2 - XV\|_F$. Note that the first row of W is $e_{m+1}^T \in \mathbf{R}^{1\times(m+1)}$ which is the $(m+1)$th unit vector.

The above derivations are summarized in Algorithm C, which is based on Algorithm B, with CSNE refinement of R if necessary. As we remarked in Section 2.1, the matrix R_1 can be obtained as a byproduct of Algorithm A and Algorithm B. Furthermore, in order to find out whether CSNE refinement can give improved accuracy, we estimate the condition numbers $\kappa(R)$ and $\kappa(R_1)$ at the extra cost of $4n^2$ flops, using a condition estimator [15, p. 128], [17].

To demonstrate the enhanced accuracy given by Algorithm C, we let $\bar{R}_1$ denote R_1 computed by Algorithm B. Then let $\hat{R}_b$ be the *exact* Cholesky factor of $\bar{R}_1^T \bar{R}_1 - Z^T Z$ (cf. (8) and (38)), and $\bar{R}_b$ the computed value by Algorithm C. Then from [14] we have the forward error estimate

$$\frac{\|\bar{R}_b - \hat{R}_b\|}{\|\hat{R}_b\|} \le C\mu\, \frac{\|\bar{R}_1\|}{\|\hat{R}_b\|}\kappa(\bar{R}_1) + O(\mu^2).$$

Combining this with (30) and comparing to (26), we see that in Algorithm C it is not $\kappa(R)$ but $\kappa(R_1)$ that determines the accuracy.

The CSNE refinement requires $7mn + 8n^2$ flops, and the total operation count for Algorithm C is $9mn + 18.5n^2$. Since Toeplitz matrix-vector multiplications can be performed using the Fast Fourier Transform (see e.g. [31]), the operation count can be reduced to $mn + 18.5n^2 + O(m \log n)$ flops (the computation of the norms of the column vectors in Algorithm B cannot be replaced by matrix-vector multiplications). In applications where m and n are large (see, e.g., [21]), this will give a significant reduction of the execution time. When the CSNE refinement is not made, the operation count is $2mn + 10.5n^2$ flops (or $mn + 10.5n^2 + O(m \log n)$ when FFT is used).

5 Numerical Tests

We have performed numerical tests in MATLAB with IEEE double precision floating point arithmetic with unit round off $\mu \approx 10^{-16}$. The accuracy of results from Algorithms

Algorithm C:
Given a Toeplitz matrix $T \in \mathbf{R}^{m \times n}$, and $0 \leq tol1 \leq tol2$, this algorithm computes the upper triangular factor R in the QR decomposition of T.

1. (a) Compute R and R_1, using Algorithm B, and estimates $\kappa(R)$ and $\kappa(R_1)$.
   If $\kappa(R) < tol1$, then **stop** (CSNE refinement is not needed).
   If $\kappa(R_1) > tol2$, then **stop** (R cannot be improved much by CSNE).

2. Compute the second row y^T of W; $W_2 = \begin{pmatrix} e_{m+1}^T \\ y^T \end{pmatrix}$

3. $X := \begin{pmatrix} T(1, 2:n) \\ T(1:m, 1:n-1) \end{pmatrix}$

4. Compute Q_1, V and F from

$$(a)\ R_1^T Q_1 = (W_2 X)^T \qquad (b)\ R_1 V = Q_1 \qquad (c)\ F := W_2^T - XV$$

5. (a) Update Q_1, V, and F:

$$\begin{aligned} R_1^T \delta Q_1 &= X^T F, \qquad Q_1 := Q_1 + \delta Q_1 \\ R_1 \delta V &= \delta Q_1, \qquad F := F - X \delta V \end{aligned}$$

   (b) Compute the R factor of F to determine Γ:

$$\Gamma = qr(F)$$

6. Determine an orthogonal matrix P as a product of Givens rotations such that

$$\begin{pmatrix} I_2 & Z \\ 0 & R_b \end{pmatrix} := P \begin{pmatrix} Q_1 & R_1 \\ \Gamma & 0 \end{pmatrix}$$

7. $R(2:n,\ 2:n) := R_b$

A, B, and C were compared. The tolerances in Algorithm C were chosen so that the CSNE refinement was done in all cases.

In order to estimate the forward error, we compared the computed matrix $\bar{R}$ to R_H which was computed by Householder QR decomposition algorithm on the Toeplitz matrix T and present the relative error

$$error := \frac{\|\bar{R} - R_H\|_F}{\|R_H\|_F}. \tag{41}$$

Given the computed matrix $\bar{R}$, we also computed an estimate of the backward error as suggested in [2], by solving the orthogonal Procrustes problem [15, p. 582]

$$\min_Q \|Q \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix} - T\|_F.$$

The solution of this problem can be expressed in terms of the SVD of $T \begin{pmatrix} \bar{R}^T & 0 \end{pmatrix}$,

$$T \begin{pmatrix} \bar{R}^T & 0 \end{pmatrix} = U\Sigma V^T,$$

and the minimum value is

$$\eta = \|UV^T \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix} - T\|_F. \tag{42}$$

For Tests I–III, we computed the minimal backward error η from (42) using the variable precision extension of MATLAB, with an accuracy of 40 decimal digits[1]. In Tests IV and V, where the matrices have larger dimension, we had to use ordinary IEEE double precision arithmetic. We present relative backward errors, $\eta/\|T\|_F$ for the three algorithms. For a backward stable algorithm, this quantity should be of the order of the unit round off.

We also plot the condition numbers $\kappa(T)$, and $\kappa(R_1)$.

Test I. Three well-conditioned Toeplitz matrices from [22, Examples 1-3] were triangularized (see also [23]). Since all three algorithms produced accurate results (errors of the order of magnitude of the unit round off as predicted by theory), we do not present details.

Test II. We consider the matrix [22, 23]:

$$T = \frac{1}{27} \begin{pmatrix} 27 & 9 & 3 & -23+\epsilon \\ 9 & 27 & 9 & 3 \\ 3 & 9 & 27 & 9 \\ -23+\epsilon & 3 & 9 & 27 \end{pmatrix},$$

for $\epsilon = 10^{-1-i}$, $i = 1, 2, \ldots, 7$, chosen so that the matrix becomes more and more ill-conditioned. The results are shown in Figure 2 for the seven test cases. In this test, Algorithm C is about as accurate as the Householder algorithm.

[1]Since we subtract two matrices to compute η, which in some cases is of the order of magnitude of the unit round off μ, we need extra precision to avoid cancellation.

Test III. The matrix T [23] is

$$T = \begin{pmatrix} 1 & 2 & 3 & 1/\epsilon \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 1/\epsilon & 3 & 2 & 1 \end{pmatrix},$$

and ϵ is chosen as in Test II. The results are presented in Figure 3. Here Algorithm B is considerably more accurate than Algorithm A. Again, Algorithm C is about as accurate as the Householder algorithm.

Test IV. In the following artificial example, where

$$T = \begin{pmatrix} 1 & t & t & t & \cdots & t \\ 0 & 1 & t & t & \cdots & t \\ 0 & 0 & 1 & t & \cdots & t \\ \vdots & & & \ddots & \ddots & \vdots \\ & & & & 1 & t \\ 0 & \cdots & & & & 1 \end{pmatrix} \in \mathbf{R}^{50\times 50},$$

we applied the algorithms presented in this paper ignoring the fact that the matrix is already triangular. The parameter t was chosen as $t = 0.05s + 1.95$, $s = 1, 2, \ldots, 7$. Note that when $t = 2.3$, the matrix is quite ill-conditioned. The results are shown in Figure 4.

Since here we have the exact R factor (equal to T), we can compute the true forward errors. A comparison of the computed R factors, shows that in Algorithm A the error grows gradually from the top row of the matrix to the bottom. The matrix produced by Algorithm C shows the same very low error level for all elements.

Test V. This test matrix is a modification of an example from [19]. A set of data was generated as

$$x_n = x(n\Delta t) = \sum_{j=1}^{9} A_j \exp(-\alpha n\Delta t)\cos(\omega_j n\Delta t) + \tau r_n,$$

where

$$\begin{aligned} A_j &= j/10, \qquad j = 1, \ldots, 9, \\ \alpha &= \pi/50, \qquad \Delta t = 0.5, \\ \omega_j &= \begin{cases} j\pi/5, & j = 1,2,3,4,5,6,8,9, \\ 7.95\pi/5 & j = 7, \end{cases} \end{aligned}$$

and r_n is taken from a normal distribution, $r_n \in N(0,1)$, and $\tau = 10^{-2}, 10^{-3}, \ldots, 10^{-8}$ in the different test cases. The matrix $T \in \mathbf{R}^{59\times 19}$ was constructed from $T_{ij} = x_{i-j+19}$. For small values of τ, it has one singular value considerably smaller than the rest. The results are shown in Figure 5.

6 Remarks and Conclusions

We have presented a forward error analysis for a fast Toeplitz triangularization algorithm which is a generalization of the classical algorithm due to Schur. The analysis shows that the relative accuracy of the computed upper triangular factor is bounded essentially by the condition number square of the given Toeplitz matrix, like in the Cholesky decomposition. Based on the analysis and CSNE block downdating algorithm, we developed a new fast $O(mn+n^2)$ algorithm for computing the R factor in the QR decomposition of a Toeplitz matrix. For the ill-conditioned problems that we have tested, the new algorithm is considerably more accurate than Algorithm A. This CSNE based algorithm is further modified so that the more accurate but expensive CSNE computation is used only for ill-conditioned problems, by monitoring the condition numbers of the matrices involved.

If the matrix R_1 in (7) is too ill-conditioned (essentially, if $\kappa(R_1) > 1/\sqrt{\mu}$, where μ is the unit round off of the floating point system; this will happen e.g. if T has three or more very small singular values), then this method may not work as well [4]. Nevertheless, the new algorithm has significantly extended the range of problems for which fast methods can produce accurate results.

In [22] it is shown how Algorithm A can be generalized to deal with problems, where T has the structure

$$T = \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_p \end{pmatrix},$$

and each T_i is a Toeplitz matrix, see [21]. The update (37) is replaced by a block update, where the block has dimension $p \times n$, and, similarly, the downdate (38) becomes a block downdate of dimension $(p+1) \times n$. Therefore, the error analysis and the CSNE refinement procedure can easily be generalized for this case. In order to more fully investigate the potential of the fast solvers for Toeplitz problems, the presented results need to be extended to handle a general right hand side vector for least squares problems. In a future report, we will present our results on this topic and extensive experimental results from application areas.

References

[1] S. T. Alexander, C.-T. Pan, and R. J. Plemmons. Analysis of a recursive least squares hyperbolic rotation algorithm for signal processing. *Lin. Alg. Appl.*, 98:3–40, 1988.

[2] Å. Björck. Error analysis of least squares algorithms. In G. H. Golub and P. Van Dooren, editors, *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*, NATO ASI Series, pages 41–73, Berlin, 1991. Springer-Verlag.

[3] Å. Björck, H. Park, and L. Eldén. Accurate downdating of least squares solutions. *SIAM J. Matrix Anal. Appl.*, 15:549–568, 1994.

[4] Å. Björck. Stability analysis of the method of semi-normal equations for least squares problems. *Lin. Alg. Appl.*, 88/89:31–48, 1987.

[5] A. W. Bojanczyk, R. P. Brent, P. van Dooren, and F. R. de Hoog. A note on downdating the Cholesky factorization. *SIAM J. Sci. Stat. Comput.*, 8:210–221, 1987.

[6] A.W. Bojanczyk, R.P. Brent, and F.R. de Hoog. QR factorization of Toeplitz matrices. *Numer. Math.*, 49:81–94, 1986.

[7] R.P. Brent. Parallel algorithms for Toeplitz systems. In G.H. Golub and P. Van Dooren, editors, *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*. Springer, 1990.

[8] J.M. Chambers. Regression updating. *J. Amer. Stat. Assoc.*, 66:744–748, 1971.

[9] J. Chun, T. Kailath, and H. Lev-Ari. Fast parallel algorithms for QR and triangular factorization. *SIAM J. Sci. Stat. Comput.*, 8:899–913, 1987.

[10] G. Cybenko. A general orthogonalization technique with applications to time series analysis and signal processing. *Math. Comp.*, 40:323–336, 1983.

[11] G. Cybenko. Fast Toeplitz orthogonalization using inner products. *SIAM J. Sci. Stat. Comput.*, 8:734–740, 1987.

[12] L. Eldén and H. Park. Block downdating of least squares solutions. *SIAM J. Matrix Anal. Appl.*, 15:1018–1034, 1994.

[13] L. Eldén and H. Park. Perturbation analysis for block downdating of a Cholesky decomposition. *Numerische Mathematik*, 68:457–467, 1994.

[14] L. Eldén and H. Park. Perturbation and error analyses for block downdating of a Cholesky decomposition. *BIT*, 36:239–255, 1996.

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations. 2nd ed.* Johns Hopkins Press, Baltimore, MD., 1989.

[16] N. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, 1996.

[17] N. J. Higham. A survey of condition number estimators. *SIAM Review*, 29:575–596, 1987.

[18] T. Kailath and J. Chun. Generalized displacement structure for block-Toeplitz, Toeplitz-block, and Toeplitz-derived matrices. *SIAM J. Matrix Anal. Appl.*, 15:114–128, 1994.

[19] W. Kolbel and H. Schafer. Improvement and automation of the LPSVD algorithm by continuous regularization of the singular values. *J. Magnetic Resonance*, 100:598–603, 1992.

[20] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* SIAM, Classics in Applied Mathematics, Philadelphia, 1995.

[21] J.J. Led and H. Gesmar. Application of the linear prediction method to NMR spectroscopy. *Chem. Rev.*, 91:1413–1426, 1991.

[22] J.G. Nagy. Fast inverse QR factorization for Toeplitz matrices. *SIAM J. Sci. Stat. Comput.*, 14:1174–1183, 1993.

[23] S. Qiao. Hybrid algorithm for fast Toeplitz orthogonalization. *Numer. Math.*, 53:351–366, 1988.

[24] M. A. Saunders. Large-scale linear programming using the Cholesky factorization. Technical Report Technical Report CS252, Computer Science Department, Stanford University, 1972.

[25] I. Schur. Über Potenzreihen, die im Inneren des Einheitskreises beschränkt sind. I. *Journal für die reine und angewandte Mathematik*, 147:205–232, 1917. (English translation: I. Schur. *On power series which are bounded in the interior of the unit circle. I.*, pages 31–59. In I. Gohberg, editor, Operator Theory: Advances and Applications. Birkhäuser Verlag, 1986.)

[26] G. W. Stewart. Perturbation bounds for the QR factorization of a matrix. *SIAM J. Numer. Anal.*, 14:509–518, 1977.

[27] G.W. Stewart. On the stability of sequential updates and downdates. *IEEE Transactions on Signal Processing*, 43:2642–2648, 1995.

[28] J. Sun. Perturbation analysis of the Cholesky downdating and QR updating problems. *SIAM J. Matrix Anal. Appl.*, 16:760–775, 1995.

[29] J.-G. Sun. Perturbation bounds for the Cholesky and QR factorizations. *BIT*, 31:341–352, 1991.

[30] D.R. Sweet. Fast Toeplitz orthogonalization. *Numer. Math.*, 43:1–21, 1984.

[31] C. F. Van Loan. *Computational frameworks for the Fast Fourier Transform.* SIAM, Philadelphia, 1992.

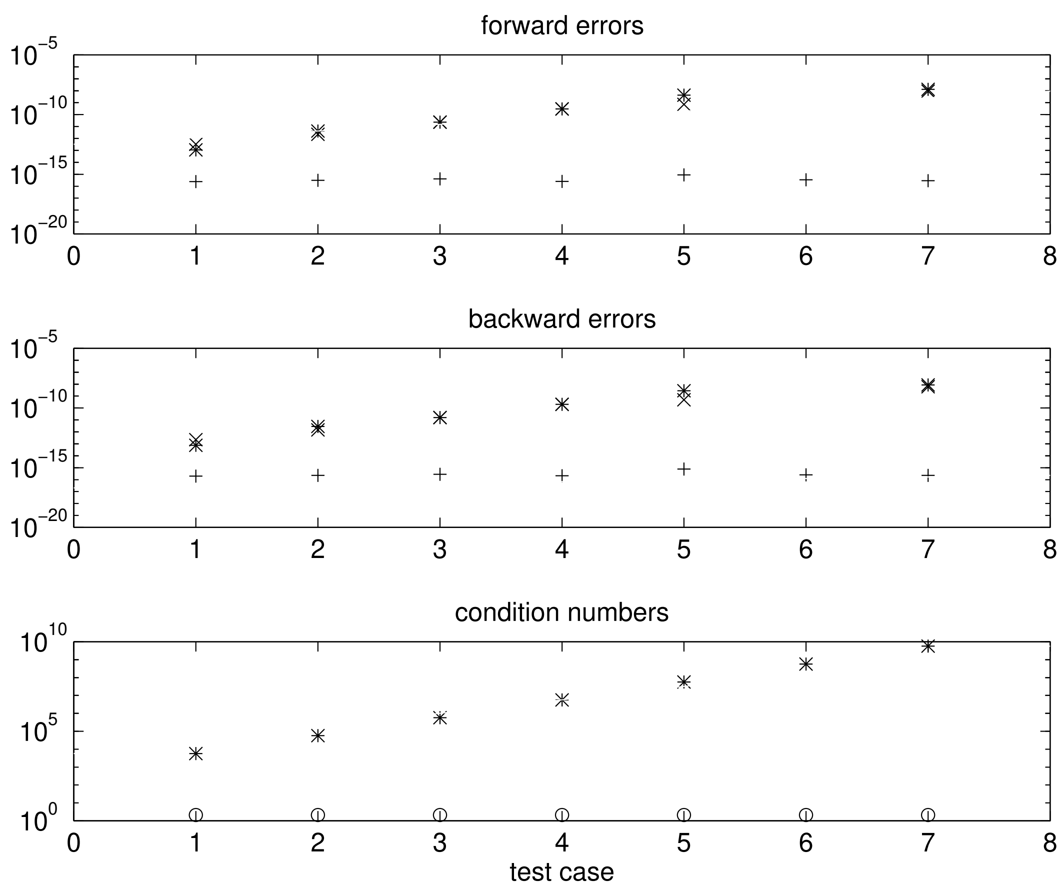

Figure 2: Test II. The upper graph shows the "forward" relative error in R computed by Algorithm A ($\times$), Algorithm B ($*$), and Algorithm C ($+$). Note that there is no value for Algorithm A and Algorithm B in one test case, since the algorithms failed (th was greater than one in one of the hyperbolic transformations). The middle graph shows the estimates of the backward error, $\eta/\|T\|_F$. The lower graph shows the condition numbers $\kappa(T)$ ($*$) and $\kappa(R_1)$ (o).

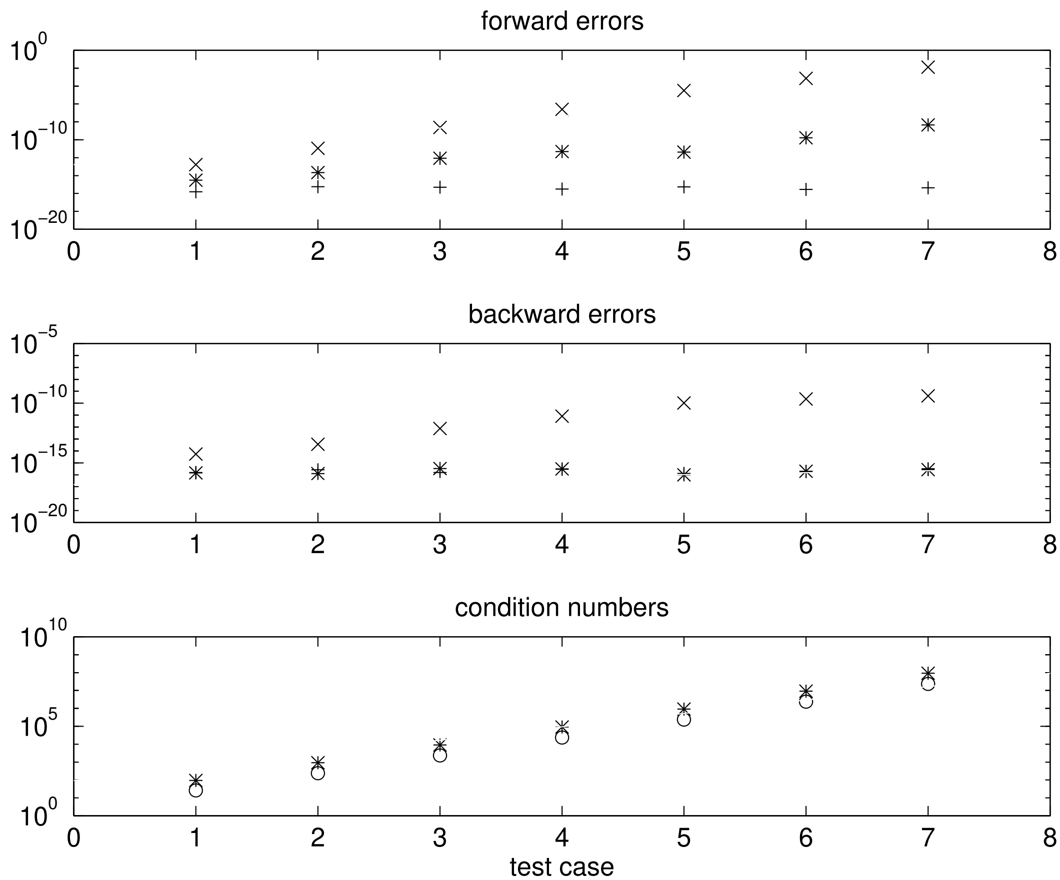

Figure 3: Test III. Estimated forward and backward relative errors in R, computed by Algorithm A ($\times$), Algorithm B ($*$), and Algorithm C ($+$), and the condition numbers $\kappa(T)$ ($*$) and $\kappa(R_1)$ (o).

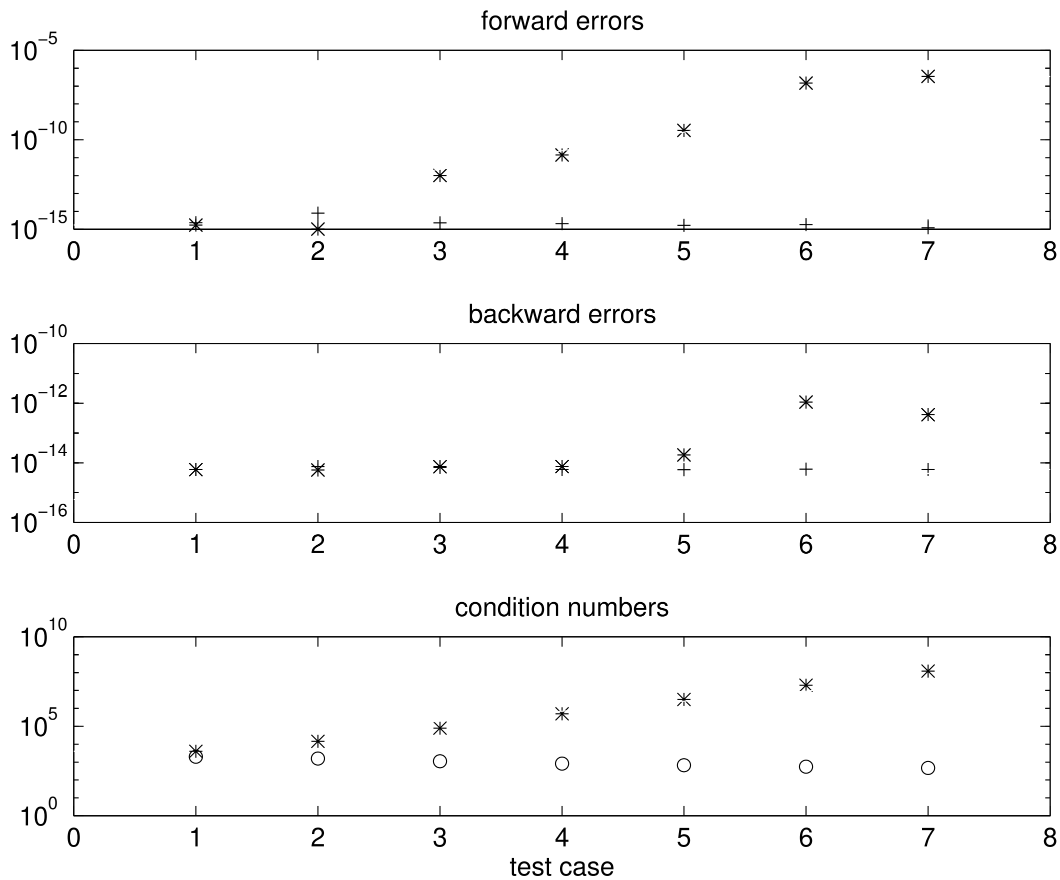

Figure 4: Test IV. True forward and estimated backward relative errors in R, computed by Algorithm A ($\times$), Algorithm B ($*$), and Algorithm C ($+$), and the condition numbers $\kappa(T)$ ($*$) and $\kappa(R_1)$ (o).

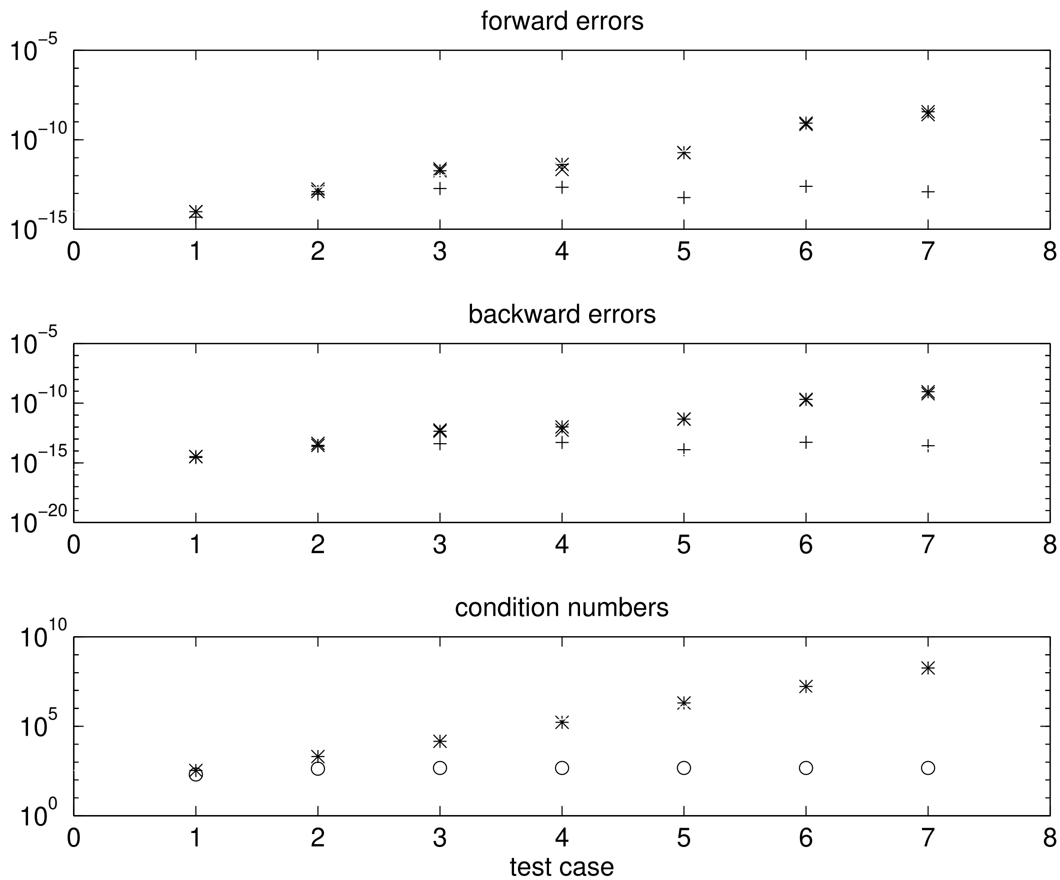

Figure 5: Test V. Estimated forward and backward relative errors in R, computed by Algorithm A ($\times$), Algorithm B ($*$), and Algorithm C ($+$), and the condition numbers $\kappa(T)$ ($*$) and $\kappa(R_1)$ (o).