VERIFIED ERROR BOUNDS FOR MATRIX DECOMPOSITIONS*

SIEGFRIED M. RUMP† AND TAKESHI OGITA‡

Abstract. In this note we consider common matrix factorizations such as LU decomposition of a square and rectangular matrix, Cholesky and QR decomposition, singular value decomposition for square and rectangular matrices, eigen-, Schur and Takagi decomposition. We first note that well-conditioned factors tend to be sensitive to perturbations of the input matrix, while ill-conditioned factors tend to be insensitive. It seems that this behaviour has not been recognized in numerical analysis. We develop a formula for the relation between condition number of the factor and its sensitivity with respect to input perturbations, and give reasons for that.

Our main focus is to describe verification methods for the factors of the mentioned decompositions. That means to prove existence of the factorization together with rigorous entrywise error bounds for the factors. Our goal is to develop algorithms requiring $\mathcal{O}(Pp^2)$ operations for an $m \times n$ matrix with $P := \max(m,n)$ and $p := \min(m,n)$. Moreover, bounds of high quality are aimed for, often not far from maximal accuracy. A main tool to achieve that is accurate dot products based on error-free transformations. Since preconditioning based on approximate inverses is used, our methods are restricted to full matrices.

Key words. Sensitivity of matrix factors, verified inclusions, error-free transformations, LU decomposition, Cholesky decomposition, QR decomposition, singular value decomposition, eigendecomposition, Schur decomposition, polar decomposition, Takagi decomposition, INTLAB

MSC codes. 65G20, 65F99

1. Introduction.

Verification methods are mathematical theorems the assumptions of which can be verified on a digital computer. The assumptions are verified with mathematical rigor including all procedural, rounding and other sources of error, thus the assertions are true with mathematical rigor. The error bounds are computed together with the proof of existence and often uniqueness of the solution. Problems cover systems of linear and nonlinear equations, eigenproblems or ordinary and partial differential equations. For the theoretical foundation and algorithms see [26, 32, 28].

Verification methods aim to formulate the assumptions in such a way that they can be rigorously verified on a computer, and that it is likely that they are satisfied for not too ill-conditioned problems. The computing time should be of the same order as that of a standard numerical algorithm. The bounds should be narrow.

There is a general limit to verification methods, namely, they are not applicable to ill-posed problems. That is the price we have to pay by using floating-point operations combined with error estimates rather than computing exactly like in computer algebra. For example, it is possible to verify that a matrix is nonsingular, even for very large condition numbers. However, it is not possible to verify that a matrix is singular because that problem is ill-posed in the sense of Tikhonov [40, 41]: An arbitrarily small change of the input data may change the answer. Similarly, even for a symmetric matrix it is not possible to compute narrow error bounds for an individual eigenvector to a double eigenvalue, see (6.1). However, verified bounds are possible for a basis of the invariant subspace.

In this note we are interested in fast verification methods for the factors of stan-

*Submitted to the editors DATE.

Funding: This work was partially supported by JSPS KAKENHI Grant Number JP23H03410.

†Institute for Reliable Computing, Hamburg University of Technology, Am Schwarzenberg-Campus 3, Hamburg 21073, Germany, and Faculty of Science and Engineering, Waseda University, 3–4–1 Okubo, Shinjuku-ku, Tokyo 169–8555, Japan (rump@tuhh.de).

‡Faculty of Science and Engineering, Waseda University, 3–4–1 Okubo, Shinjuku-ku, Tokyo 169–8555, Japan (t.ogita@waseda.jp).

dard matrix decompositions with emphasis on the complete matrix decomposition. For example, methods are known [18, 4] to compute error bounds for a single eigenpair. For an $n \times n$ matrix those methods might be applied to each individual eigenpair, however, resulting in totally $\mathcal{O}(n^4)$ operations. In contrast, [25, 24] give methods to compute error bounds for the complete eigendecomposition in $\mathcal{O}(n^3)$ operations including the treatment of clustered and/or multiple eigenvalues.

In [34] fast methods are described to compute error bounds for the complete singular value decomposition of an $m \times n$ matrix with special emphasis on clustered and/or multiple singular values. Here "fast" means $\mathcal{O}(Pp^2)$ operations with $P := \max(m,n)$ and $p := \min(m,n)$.

However, no verification methods are known for other standard matrix decompositions. We close this gap by giving fast algorithms for the LU, Cholesky, QR, and Schur decomposition. In addition to "fast" in terms of $\mathcal{O}(Pp^2)$ operations we aim on inclusions being accurate for all solution components, i.e., the entrywise relative error between lower and upper bound should be close to the relative rounding error unit $\mathbf{u}$ of the floating-point arithmetic in use.

Let $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. We use the notation $M_{m,n}$ for the set of matrices in $\mathbb{K}^{m \times n}$, and shortly M_n if $m = n$. For $A \in M_{m,n}$ we denote by $A_k \in M_k$ the upper left $k \times k$ principal submatrix of A. We adopt the convention that inverses are assumed to exist if used. The $n \times n$ identity matrix is denoted by I_n, where the index is omitted if clear from the context. Moreover, $I_{m.n} \in M_{m,n}$ is the matrix with I_p for $p := \min(m,n)$ in the upper left corner and zero elsewhere.

Any method to compute rigorous error bounds for scalar, vector and matrix operations is suitable for the algorithms to be presented. We use interval arithmetic [26] because it is simple and intuitive to use, in particular in INTLAB [31], the MATLAB/Octave toolbox for reliable computing. We use the interval notation [15], where in particular boldface letters indicate interval quantities.

Not much knowledge about verification methods and/or interval arithmetic is necessary to follow this note, basically familiarity with MATLAB notation. The representation of intervals like infimum-supremum or midpoint-radius is not important: throughout this note we only use the *inclusion property*, namely, that interval operations $\circ \in \{+, -, \cdot, /\}$ are defined such that for compatible interval quantities $\mathbf{A}, \mathbf{B}$

$$\forall A \in \mathbf{A} \; \forall B \in \mathbf{B}\colon \qquad A \circ B \in \mathbf{A} \circ \mathbf{B} \tag{1.1}$$

is satisfied. For details see [26, 32, 28]. For $M \in M_n(\mathbb{K})$ and non-negative $R \in M_n(\mathbb{R})$ the command `midrad(M,R)` is a superset of $\{A \in M_n(\mathbb{K})\colon |A - M| \leqslant R\}$ with entrywise comparison and absolute value. Moreover, $\mathbf{X} = f(\mathbf{A})$ for an interval quantity $\mathbf{A}$ and the induced function f implies that $f(A) \in \mathbf{X}$ for all $A \in \mathbf{A}$.

For an interval $\mathbf{X}$, the magnitude is defined by $\mathrm{mag}(\mathbf{X}) := \max\{|x| : x \in \mathbf{X}\} \geqslant 0$. The definition applies entrywise to vectors and matrices, so that $B = \mathrm{mag}(\mathbf{A})$ satisfies $|A_{ij}| \leqslant B_{ij}$ for all i, j, cf. [26]. The result B is a non-negative vector/matrix.

Throughout this note we use the new definition [35] of the relative error of an interval quantity $\mathbf{X}$, which is basically $\mathrm{diam}(\mathbf{X})/\mathrm{mag}(\mathbf{X})$ for $\mathrm{diam}(\mathbf{X})$ denoting the diameter of $\mathbf{X}$. The definition applies to vectors and matrices entrywise.

We use some notations in MATLAB-style, in particular

$A^{[\ell]}$	the strictly lower triangular part of A
$A^{[u]}$	the upper triangular part of A
`max`(A)	the row vector of columnwise maxima of A
`sum`$(A,2)$	the column vector of rowwise sums of A

for square A. For the maximum and sum we use the typewriter font to avoid confusion with the mathematical terms. The MATLAB notation for $A^{[\ell]}$ and $A^{[u]}$ is `tril(A,-1)` and `triu(A)`, respectively. We introduce this short notation because we use them frequently when developing the algorithms for the LU decomposition. The maximum and sum apply to $A \in M_{m,n}$ as well, and also the triangular parts apply to $A \in M_{m,n}$ in the sense that, for example, $B = \mathrm{tril}(A)$ is the lower triangular part of A. That means for matrices of any dimension $A = A^{[\ell]} + A^{[u]}$.

For $A, B, C \in M_n$ we note entrywise upper bounds for $|ABC|$. Denote by $\mu := \mathtt{max}(|C|)$ the row vector of columnwise maxima of $|C|$, and by $\sigma := \mathtt{sum}(|A|,2)$ the column vector of rowwise sums of $|A|$, i.e., $\mu_\ell = \max_{1 \leqslant k \leqslant n} |C_{k\ell}|$ and $\sigma_i = \sum_{j=1}^n |A_{ij}|$. Then

$$\begin{aligned} |ABC|_{i\ell} &= \textstyle\sum_{j=1}^n \sum_{k=1}^n |A_{ij} B_{jk} C_{k\ell}| \leqslant \sum_{j=1}^n \sum_{k=1}^n |A_{ij} B_{jk} \mu_\ell| \\ &\leqslant \|B\|_\infty \mu_\ell \textstyle\sum_{j=1}^n |A_{ij}| \leqslant \sigma_i \mu_\ell \|B\|_\infty \ , \end{aligned} \tag{1.2}$$

so that entrywise upper bounds

$$|ABC| \leqslant \mathtt{sum}(|A|,2)\ \mathtt{max}(|C|)\|B\|_\infty \quad \text{and} \quad |AB| \leqslant \mathtt{sum}(|A|,2)\ \mathtt{max}(|B|) \tag{1.3}$$

by outer products follow. Note that the computational cost is $\mathcal{O}(n^2)$. With the same complexity the bounds

$$|ABC| \leqslant \mathtt{sum}(|A|,2)\ \mathtt{max}(|B|) \cdot |C| \quad \text{and} \quad |ABC| \leqslant |A| \cdot \mathtt{sum}(|B|,2)\ \mathtt{max}(|C|)$$

follow; however, in our applications $B = (I + F)^{-1}$ is a perturbation of the identity matrix, so that the entries of B are not available but an upper bound for $\|B\|_\infty$ is. These estimates are true for real and complex matrices, as well as for any compatible matrix dimensions of A, B, and C.

All computational results use MATLAB [22] and double precision (binary64), i.e., some 53 bits in the mantissa with the relative rounding error unit $\mathbf{u} = 2^{-53} \approx 10^{-16}$. We use directed rounding which is part of the IEEE 754 arithmetic standard [1]. The statement `setround(-1)` implies that from now until the next call of `setround` the rounding mode is downwards, i.e., towards $-\infty$. As a consequence the result of floating-point operations is the largest floating-poing number being less than or equal to the true real result. Similarly, `setround(1)` switches the rounding upwards so that the smallest floating-poing number being greater than or equal to the true real result is computed. These statements are true for vector and matrix operations as well. As a consequence, the code sequence

```
setround(-1), Cinf = A*B;
setround(+1), Csup = A*B;
flpt = isequal(Cinf,Csup)
```

produces the result $flpt = true$ if, and only if, all entries of the product AB are exactly representable in floating-point.

We aim on producing accurate bounds, i.e., the lower and upper bound often differ by few bits. Our main tool to achieve this are accurate dot products, either purely approximate or with error bound. To that end there are many techniques. Some of the early references are [42, 23, 21]. Later so-called "error-free transformations" [16, 7] were used to transform a pair (a, b) of floating-point numbers into a new pair (x, y) such that, e.g., x is the floating-point product ab and y is the error in the sense $ab = x + y$ and similarly for sum, quotient, and square root. That technique was used in [27] to introduce "error-free vector transformations" where a vector v of n floating-point numbers is transformed into a vector w of the same length such that w_n is the

floating-point sum of the v_i and $\sum v_i = \sum w_i$. In that paper the term "error-free transformations" was coined which was the start of a revival of such methods. Using error-free vector transformations, sums and dot products of arbitrarily large condition number can be computed with maximal precision [27].

The mentioned error-free transformations are based on a relative splitting of the input data. Yet a completely different method was introduced in [43] where an absolute splitting of vectors was introduced. That method was analyzed in [36] and is also used for reproducible results [3]. Moreover, this method was used to develop very efficient algorithm for accurate matrix multiplication [29], with or without error bounds. In our note we use such algorithms. They work in ordinary double precision floating-point arithmetic but produce a result "as if" computed with doubled precision, i.e., some 32 decimal digits, or more. That allows to store the result of a matrix product in two terms, a higher and a lower order part. We use that technique occasionally. To that end some double-double arithmetic as in [5] or [19] could be used as well.

To be more precise the function `prodK` computes matrix products in $(1 + \frac{k}{2})$-fold precision and rounds the result into working precision. In fact `prodK` is a very versatile function, but we need only few functionalities. For example, for a given square matrix A the code

```
k = 2;
[L,U,p] = lu(A,'vector');
R = prodK(L,U,-1,A(p,:),k);
```

computes the residual $LU - A(p,:)$ in two-fold precision and rounds the result into `R` in working precision. The call

```
[R,E] = prodK(L,U,-1,A(p,:),k);
```

produces the same `R` but in addition an error matrix `E` such that

$$|LU - A(p,:) - R|_{ij} \leqslant E_{ij}.$$

for all indices i, j. Larger values of k are possible, but not used in this note. When calculating triple products ABC it may be useful to compute the first product in two-fold precision but also store it as an unevaluated sum $P_1 + P_2$. For example,

```
P = prodK(A,B,k,'OutputTerms',2);
X = prodK(P,C,k);
```

stores `P` as a cell array, and the second call of `prodK` computes $\texttt{P}_1 * \texttt{C} + \texttt{P}_2 * \texttt{C}$ in two-fold precision and stores the result in `X` in working precision.

The key to our verification methods will be to transform the problem into the problem for a perturbed identity matrix. In particular in combination with extra-precise dot products that technique turns out to be effective. The transformation uses approximate inverses of approximate factors. These are usually full, also for sparse input matrix. Therefore applying our methods to sparse matrices is prohibitive because of expected fill-in. For some factorizations such as LU and at least the R-factor of QR are usually sparse for sparse input. Verified inclusions for these cases are open problems.

We begin with an investigation of the sensitivity of matrix factors. In particular the fact that, in case of an ill-conditioned input matrix A, well-conditioned factors tend to be sensitive to perturbations of A seems unknown in numerical analysis. In the following sections verification methods for the factors of the LU decomposition of a square and rectangular matrix, Cholesky- and QR decomposition, singular value decomposition for square and rectangular matrices, eigen- and Schur decomposition

are presented, accompanied by numerical results. As an application of the symmetric eigendecomposition we show how to compute inclusions for the Takagi factors.

Throughout the note random matrices $A \in \mathbb{F}^{m\times n}$ with specified condition number $\mathrm{cond}(A) \approx 10^k$ are generated by

```
mn = min(m,n); s = logspace(0,k,mn); S = diag(s(randperm(mn)));
if m~=n, S(m,n)=0; end; A = orth(randn(m)) * S * orth(randn(n));
```

which is for square matrices equivalent to MATLAB's `gallery/randsvd`.

We present numerical evidence that mostly our method compute error bounds with an accuracy close to the relative rounding error unit $\mathbf{u}$ of the floating-point arithmetic in use. All our algorithms are given and implemented in pure MATLAB code, therefore suffering from interpretation overhead. Therefore we restrict timing information to the QR decomposition in Section 5 together with accuracy information of the built-in (approximate) MATLAB routines; the time ratio of other verification methods is similar.

2. Sensitivity of factors in a decomposition. For any of the matrix decompositions under investigation we made a general observation which seems to be known in the literature [13, 12, 8] but not so much in numerical analysis. Some perturbation bounds for LU, Cholesky, and QR decompositions can be found in [37], see also [11], however they overestimate the sensitivity of ill-conditioned factors.

Let X be a factor of some decomposition of a matrix A. Denote by $A + \Delta A$ a small perturbation of A such that $\frac{\|\Delta A\|}{\|A\|} \sim \mathbf{u}$ for some matrix norm, and let $\tilde{X}$ be the corresponding factor of $A + \Delta A$. Then numerical evidence (cf. Tables 1–10) suggests that often the sensitivity of X satisfies

$$\mathrm{sensitivity}(X) := \frac{\|\tilde{X} - X\|}{\|X\|} \sim \mathbf{u}\, \frac{\mathrm{cond}(A)}{\mathrm{cond}(X)}\ , \tag{2.1}$$

where $\mathrm{cond}(B) := \|B\| \cdot \|B^{-1}\|$ for a nonsingular square matrix B.

For example, suppose $A = LU$ with L being unit lower triangular. Then the U-factor of the LU decomposition has usually the same condition number as A. Although the L-factor is usually well conditioned by the widely accepted rule of thumb, numerical evidence (cf. Tables 1–4) suggests that its sensitivity grows with the condition number of A. That can be seen as follows. Let $A + \Delta A = (L + \Delta L)(U + \Delta U)$, then to first order

$$L^{-1} \cdot \Delta A \cdot U^{-1} = L^{-1} \cdot \Delta L + \Delta U \cdot U^{-1}\ . \tag{2.2}$$

The matrices ΔL and $L^{-1} \cdot \Delta L$ are strictly lower triangular, whereas $\Delta U \cdot U^{-1}$ is upper triangular. Thus, taking the strictly lower triangular and upper triangular part from matrices of both sides of (2.2) implies

$$\Delta L = L\left[L^{-1} \cdot \Delta A \cdot U^{-1}\right]^{[\ell]} \quad \text{and} \quad \Delta U = \left[L^{-1} \cdot \Delta A \cdot U^{-1}\right]^{[u]} U\ .$$

Numerical evidence suggests that the elements of each row of (the upper triangular part of) U are often of similar magnitude[1], so that $U \approx DX$ for diagonal D with elements decreasing in magnitude with $\|D^{-1}\| \sim \|A^{-1}\|$ and well-conditioned X with the upper triangular part of entries close to 1 in magnitude. Hence

$$L^{-1} \cdot \Delta A \cdot U^{-1} \approx L^{-1} \cdot \Delta A \cdot X^{-1}D^{-1} =: Y D^{-1}$$

[1]That is true due to our practical experience, and it is also satisfied for matrices generated by `randsvd` from MATLAB's matrix gallery in any of the 5 modes. However, for ill-conditioned matrices generated by `sprand` with densitity 1, i.e., full matrices, it is sometimes not true.

for some Y with entries of the size of those of ΔA, i.e., $\|Y\| \sim \|\Delta A\|$. Then

$$\Delta L = L\left[L^{-1} \cdot \Delta A \cdot U^{-1}\right]^{[\ell]} \approx L\left[YD^{-1}\right]^{[\ell]} = LY^{[\ell]}D^{-1}$$

and

$$\Delta U = \left[L^{-1} \cdot \Delta A \cdot U^{-1}\right]^{[u]} U \approx \left[YD^{-1}\right]^{[u]} DX = Y^{[u]}X \ .$$

Now $\|L\|$ and $\|X\|$ are small because both are usually well conditioned, so that $\|D^{-1}\| \sim \|A^{-1}\|$ and $\|Y\| \sim \|\Delta A\| \sim \mathbf{u}\|A\|$ imply

$$\|\Delta L\| \sim \|\Delta A\| \cdot \|A^{-1}\| \sim \mathbf{u} \cdot \mathrm{cond}(A) \quad \text{and} \quad \|\Delta U\| \sim \|\Delta A\| \sim \mathbf{u}\|A\|$$

and explain (2.1) for the LU decomposition.

For the QR decomposition (2.1) is mentioned in [20]. Let $A = QR$ and $A + \Delta A = (Q + \Delta Q)(R + \Delta R)$, so that to first order

$$M := Q^* \cdot \Delta A \cdot R^{-1} = Q^* \cdot \Delta Q + \Delta R \cdot R^{-1} \ \text{ and } \ [Q^* \cdot \Delta A \cdot R^{-1}]^{[\ell]} = [Q^* \cdot \Delta Q]^{[\ell]} \ .$$

Using $(Q + \Delta Q)^*(Q + \Delta Q) = I$ implies that $C := Q^* \cdot \Delta Q$ is skew-Hermitian, so that $M^{[\ell]} = C^{[\ell]}$ yields

$$C = C^{[\ell]} - \left(C^{[\ell]}\right)^* = M^{[\ell]} - \left(M^{[\ell]}\right)^* \quad \text{and} \quad \Delta Q = Q\left[M^{[\ell]} - \left(M^{[\ell]}\right)^*\right]$$

and explains (2.1) for the Q-factor. The perturbation of the R-factor satisfies

$$\Delta R = [Q^* \cdot \Delta A]^{[u]} - \left[M^{[\ell]}R\right]^{[u]} + \left[\left(M^{[\ell]}\right)^* R\right]^{[u]} \ . \tag{2.3}$$

The first summand support (2.1), i.e., that R is not very sensitive to perturbations of A, the second and third one need some extra consideration. In a similar way to the LU decomposition, numerical evidence suggests that $R \approx DX$ for diagonal D with elements decreasing in magnitude and well-conditioned X with entries close to 1 in magnitude. Hence

$$\left[M^{[\ell]}R\right]^{[u]} \approx \left[\left[Q^* \cdot \Delta A \cdot X^{-1}D^{-1}\right]^{[\ell]} DX\right]^{[u]} = \left[\left[Q^* \cdot \Delta A \cdot X^{-1}\right]^{[\ell]} X\right]^{[u]}$$

which is of the order $\|\Delta A\|$. For the third summand of (2.3) we have

$$\left[\left(M^{[\ell]}\right)^* R\right]^{[u]} \approx \left[\left(\left[Q^* \cdot \Delta A \cdot X^{-1}D^{-1}\right]^{[\ell]}\right)^* DX\right]^{[u]} \approx \left[(D^{-1})^* YDX\right]^{[u]}$$

for some Y with entries of the size of $\|\Delta A\|$. Since the D_{ii} decrease in magnitude, that supports (2.1) for R as well. For the Cholesky decomposition $A = R^T R$ the ansatz

$$R^{-T}\Delta A \cdot R^{-1} = R^{-T}\Delta R^T + \Delta R \cdot R^{-1} \quad \text{implies} \quad \Delta R = [R^{-T}\Delta A \cdot R^{-1}]^{[u]}R$$

and explains (2.1) along the same lines as well. The condition number of the Cholesky factor is $\mathrm{cond}(A)^{1/2}$, and numerical evidence suggests indeed that the sensitivity is always of the order $\mathbf{u}\ \mathrm{cond}(A)^{1/2}$ in accordance with (2.1). Similarly, for the QR decomposition $\mathrm{cond}(Q) = 1$ and $\mathrm{cond}(R) = \mathrm{cond}(A)$, so that (2.1) suggests that Q is sensitive to perturbations of A while R is not. Numerical evidence supporting these statements will be presented in the following sections.

As a consequence and from a numerical standpoint of view to our surprise, we may expect that accurate inclusions are more demanding for well conditioned factors.

3. LU decomposition. If all upper left principal minors $\det(A_k)$ of $A \in M_{m,n}$ are nonzero, then there is a unique LU decomposition of A. That is true for square as well as for rectangular matrices. If the first $m-1$ minors are nonzero but $\det(A_m) = 0$, then the decomposition exists but is not unique [11, Theorem 9.1].

A verification method for computing inclusions of the L- and U-factor of a matrix A asserts, with mathematical certainty, that the decomposition exists and is unique. Thus, a necessary condition is that A has full rank.

Given a matrix $A \in M_{m,n}$, the following MATLAB code in Algorithm 3.1 (`getL`) computes the L-factor of A, cf. [10, p. 35], [11, (9.2a)].

Algorithm 3.1 Computation of the L-factor

```
function L = getL(A)
  [m,n] = size(A);
  mn = min(m,n);
  L = eye(m,mn);
  for k=1:mn-1
    v = 1:k;
    w = k+1:m;
    Bv = inv(A(v,v));     % last column of inv(A(v,v)) needed
    L(w,k) = A(w,v)*Bv(:,end);
  end
```

For square $A \in M_n$, this requires to compute the last column of the inverse of A_k for $1 \leqslant k \leqslant n-1$. To that end we see no other way than to compute the inverses individually at the cost of $\mathcal{O}(k^3)$ operations each, so that totally prohibitive $\mathcal{O}(n^4)$ operations are necessary.

Let $A \in \mathbb{K}^{m\times n}$ be given and denote $P := \max(m,n)$ and $p := \min(m,n)$. Our goal is to compute verified and sharp error bounds for the factors L and U of A with a total computing time of $\mathcal{O}(Pp^2)$ operations. This will be achieved by preconditioners X_L, X_U such that $X_L A X_U$ is a perturbed identity matrix I_E.

We first show how to compute the LU decomposition of a perturbed identity matrix, followed by the cases $m = n$, $m > n$, and $m < n$ for the LU decomposition of a general matrix.

3.1. LU decomposition of a perturbed identity matrix. Let $A \in M_{m,n}$ with $m \geqslant n$ be given, denote $E := A - I_{m,n}$ and assume that E_n is convergent. Fix k with $1 \leqslant k \leqslant n$, let i satisfy $k+1 \leqslant i \leqslant m$, and denote $B := A_k^{-1} = (I_k + E_k)^{-1}$. Then, according to Algorithm 3.1 (`getL`), using $(I+E_k)^{-1} = I - (I+E_k)^{-1}E_k$ and $A_{i\nu} = E_{i\nu}$ for $i > k$ and $\nu \leqslant k$ yields

$$L_{ik} = \sum_{\nu=1}^{k} A_{i\nu}B_{\nu k} = \sum_{\nu=1}^{k} E_{i\nu}[I_k - (I_k + E_k)^{-1}E_k]_{\nu k} \ .$$

Hence, denoting the k-th column of I_k by $e^{(k)}$ and using $i > \nu$ gives

$$L_{ik} - E_{ik} = -\sum_{\nu=1}^{k} (E^{[\ell]})_{i\nu}[(I_k + E_k)^{-1}E_k e^{(k)}]_\nu \ .$$

Using (1.2) it follows

$$|L_{ik} - E_{ik}| \leqslant \sum_{\nu=1}^{k} \left|E^{[\ell]}\right|_{i\nu} \|(I_k + E_k)^{-1}E_k e^{(k)}\|_\infty \leqslant \sum_{\nu=1}^{k} \left|E^{[\ell]}\right|_{i\nu} \frac{\|E_k e^{(k)}\|_\infty}{1 - \|E_k\|_\infty} \ .$$

The k-th component of the row vector $\max(|E_n^{[u]}|)$ is equal to $\|E_k e^{(k)}\|_\infty$, so that the strictly lower triangular part of the difference between L and E is bounded above by the outer product

$$\left| L^{[\ell]} - E^{[\ell]} \right| \leqslant \frac{\left(\mathtt{sum}(|E^{[\ell]}|,2) \cdot \mathtt{max}(|E_n^{[u]}|) \right)^{[\ell]}}{1 - \|E_n\|_\infty} =: \Delta = \Delta^{[\ell]} \ , \tag{3.1}$$

and there exists a strictly lower triangular matrix $C = C^{[\ell]}$ with

$$L = I + E^{[\ell]} + C^{[\ell]} \quad \text{and} \quad |C^{[\ell]}| \leqslant \Delta^{[\ell]} \ . \tag{3.2}$$

In other words, the strictly lower triangular part of L is essentially equal to the strictly lower triangular part of E. The computational cost is $\mathcal{O}(mn)$ operations. For $m < n$ the factor L is square, and along the same lines we deduce

$$\left| L^{[\ell]} - E_m^{[\ell]} \right| \leqslant \frac{\left(\mathtt{sum}(|E_m^{[\ell]}|,2) \cdot \mathtt{max}(|E_m^{[u]}|) \right)^{[\ell]}}{1 - \|E_m\|_\infty} \ . \tag{3.3}$$

If L is square, i.e., $m \leqslant n$, an inclusion of L^{-1} can be obtained using verification methods [32], however, we may proceed directly by using the Neumann expansion

$$(I+F)^{-1} = I - F(I+F)^{-1} = I - (I+F)^{-1}F = I - F + F(I+F)^{-1}F \ .$$

Then (1.3) implies

$$|(I+F)^{-1} - I + F| \leqslant \frac{\mathtt{sum}(|F|,2)\ \mathtt{max}(|F|)}{1 - \|F\|_\infty}$$

provided that $\|F\|_\infty < 1$. Using (3.2), $F := E^{[\ell]} + C^{[\ell]}$ and $G := |E^{[\ell]}| + \Delta^{[\ell]}$ yields

$$L^{-1} = I - E^{[\ell]} + \delta \quad \text{with} \quad |\delta| \leqslant \Delta^{[\ell]} + \left[\frac{\mathtt{sum}(G,2)\ \mathtt{max}(G)}{1 - \|G\|_\infty} \right]^{[\ell]} \ . \tag{3.4}$$

Note that $G = |L - I|$ and $\delta = \delta^{[\ell]}$, and error bounds are only needed for the strictly lower triangular part. The estimate may be improved by using more terms of the Neumann expansion, however, it seems hardly worth the effort.

In order to compute an inclusion of U we may use, regardless whether $m \geqslant n$ or $m < n$, the L-factor of the upper left square matrix of $I + E$, an inclusion of which can be computed as described before. For the case $m \geqslant n$ we have $(I+E)_n = L_n U$, and the uniqueness of the LU decomposition, (3.2) and $(I+F)^{-1}x = x - (I+F)^{-1}Fx$ imply that

$$\begin{aligned} U &= (I_n + E_n^{[\ell]} + C_n^{[\ell]})^{-1}(I_n + E_n) \\ &= (I_n + E_n^{[\ell]} + C_n^{[\ell]})^{-1}(I_n + E_n^{[\ell]} + C_n^{[\ell]} + E_n^{[u]} - C_n^{[\ell]}) \\ &= I_n + (I_n + E_n^{[\ell]} + C_n^{[\ell]})^{-1}(E_n^{[u]} - C_n^{[\ell]}) \\ &= I_n + E_n^{[u]} - C_n^{[\ell]} - (I_n + E_n^{[\ell]} + C_n^{[\ell]})^{-1}(E_n^{[\ell]} + C_n^{[\ell]})(E_n^{[u]} - C_n^{[\ell]}) \ . \end{aligned} \tag{3.5}$$

Note that the rightmost factor $E_n^{[u]} - C_n^{[\ell]}$ is a composition into upper and strictly lower part. Now $(C^{[\ell]})^{[u]} = O$ because $U = U^{[u]}$ is upper triangular, and similar to

the estimate for the L-factor it follows

$$\left| U - (I_n + E_n^{[u]}) \right| \leqslant \frac{\left(\mathtt{sum}(|E_n^{[\ell]} + C_n^{[\ell]}|,2) \cdot \mathtt{max}(|E_n^{[u]} - C_n^{[\ell]}|) \right)^{[u]}}{1 - \|E_n^{[\ell]} + C_n^{[\ell]}\|_\infty} \ . \tag{3.5}$$

For the case $m < n$ we identify the matrix dimensions by adding subscripts. For example, E_m denotes the left upper $m \times m$ principal submatrix of E and indicates the dimension as well. For $A \in M_{m,n}$ we have $A = I_{m,n} + E_{m,n} = L_m U_{m,n}$, so that $C_m^{[\ell]} = C_m$ with (3.2) for L_m, and for $P \in M_{m,n_1}, Q \in M_{m,n_2}$ with matrix block notation $[P,Q] \in M_{m,n_1+n_2}$ it follows

$$\begin{aligned}
U_{m,n} &= (I_m + E_m^{[\ell]} + C_m^{[\ell]})^{-1}(I_{m,n} + E_{m,n}) \\
&= (I_m + E_m^{[\ell]} + C_m^{[\ell]})^{-1}\Big(I_{m,n} + [E_m^{[\ell]} + C_m^{[\ell]} \ , \ O_{m,n-m}] + [E_m^{[u]} - C_m^{[\ell]} \ , \ E_{m,n-m}]\Big) \\
&= I_{mn,n} + (I_m + E_m^{[\ell]} + C_m^{[\ell]})^{-1}[E_m^{[u]} - C_m^{[\ell]} \ , \ E_{m,n-m}] \\
&= I_{mn,n} + [E_m^{[u]} - C_m^{[\ell]} \ , \ E_{m,n-m}] \\
&\qquad + (I_m + E_m^{[\ell]} + C_m^{[\ell]})^{-1}(E_m^{[\ell]} + C_m^{[\ell]})[E_m^{[u]} - C_m^{[\ell]} \ , \ E_{m,n-m}] \ .
\end{aligned}$$

Since $U \in M_{m,n}$ is upper triangular we obtain, similar to the previous estimate,

$$\left| U - \left(I_{m,n} + E_{m,n}^{[u]} \right) \right| \leqslant \frac{\left(\mathtt{sum}(|E_m^{[\ell]} + C_m^{[\ell]}|,2) \cdot \mathtt{max}(B_{m,n}) \right)^{[u]}}{1 - \|E_m^{[\ell]} + C_m^{[\ell]}\|_\infty} =: \Delta$$

using $B_{m,n} := [|E_m^{[u]}| + |C_m^{[\ell]}| \ , \ |E_{m,n-m}|]$ which is $|E|$ with $E_m^{[\ell]}$ replaced by $|C_m^{[\ell]}|$. The computational cost is again $\mathcal{O}(mn)$ operations.

If $m \leqslant n$ we may derive an inclusion of U^{-1} as well. We rewrite (3.5) into

$$U = I + E^{[u]} + C^{[u]} \quad \text{and} \quad |C^{[u]}| \leqslant \Delta^{[u]} = \Delta \tag{3.6}$$

analogously to (3.2), and for $G := |E^{[u]}| + \Delta^{[u]}$, similar to (3.4), it follows

$$U^{-1} = I - E^{[u]} + \delta \quad \text{with} \quad |\delta| \leqslant \Delta^{[u]} + \left[\frac{\mathtt{sum}(G,2) \ \mathtt{max}(G)}{1 - \|G\|_\infty} \right]^{[u]} \ . \tag{3.7}$$

In Figure 1 the norm of the right hand side of (3.1) and (3.5) together with $\sqrt{n}\|E\|^2$ is displayed for dimensions $(m,n) = (100,200)$ and $(m,n) = (200,100)$ for different norms of E. As can be seen the norm of the error of the bounds for L and U grow with $\sqrt{n}\|E\|^2$. Hence, for $\|E\| \lesssim 10^{-8}$ the inclusions of the factors are maximally accurate, with errors of the size of the relative rounding error unit $\mathbf{u} \approx 10^{-16}$. Actually Figure 1 displays results for complex input matrix; the results for real input are similar.

The graph of the norm of the error as in (3.4) and (3.7) of L^{-1} and U^{-1}, respectively, looks exactly like Figure 1, so the error for the inclusions of L^{-1} and U^{-1} grows with $\sqrt{n}\|E\|^2$ as well.

We close this subsection with giving executable MATLAB/INTLAB code in Algorithm 3.2 for the computation of inclusions of the L- and U-factor as well as of their inverses for a perturbed identity matrix $I + E$. Input is the perturbation E, and inclusions are stored as perturbations of the identity matrix as well, i.e., $I + E = LU$ implies $L \in I + \mathtt{LE}$, $U \in I + \mathtt{UE}$, $L^{-1} \in I + \mathtt{LinvE}$, and $U^{-1} \in I + \mathtt{UinvE}$.

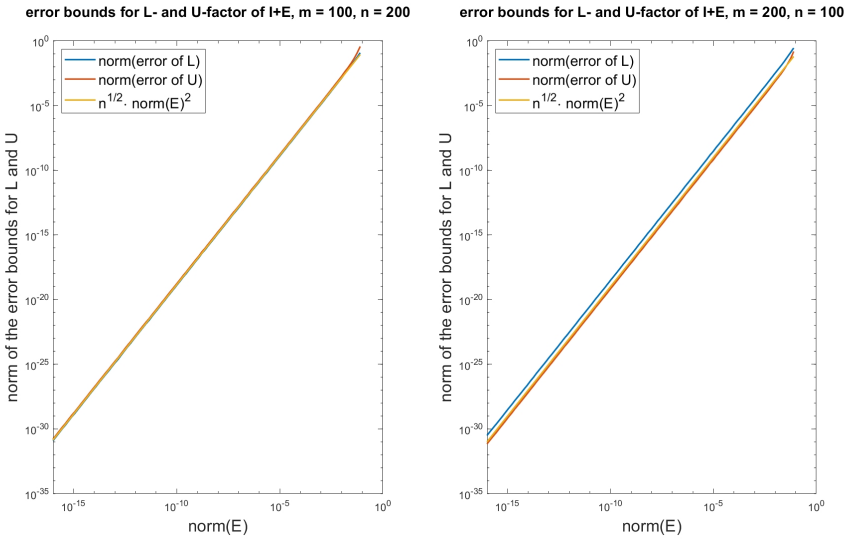

FIG. 1. *Norm of error bound for the L- and U-factor of $I + E$*

Algorithm 3.2 Inclusion of L- and U-factor of $I + E$ and their inverses

```
 1   function [LE,UE,LinvE,UinvE] = LU_E(E)
 2     setround(1)
 3     magE = mag(E);
 4     DeltaL = tril(sum(tril(magE,-1),2)*max(magE),-1);
 5     DeltaL = - ( DeltaL/(norm(magE,inf)-1) );
 6     LE = tril(E,-1) + midrad(0,DeltaL);
 7     GL = mag(LE);
 8     delta = tril(sum(tril(GL,-1),2)*max(GL),-1);
 9     delta = DeltaL - delta/(norm(GL,inf)-1);
10     LinvE = -tril(E,-1) + midrad(0,delta);
11     B = triu(magE) + tril(DeltaL,-1);
12     DeltaU = triu(sum(GL,2)*max(B));
13     DeltaU = - ( DeltaU/(norm(GL,inf)-1) );
14     UE = triu(E) + midrad(0,DeltaU);
15     GU = mag(UE);
16     delta = triu(sum(triu(GU),2)*max(GU));
17     delta = DeltaU - delta/(norm(GU,inf)-1);
18     UinvE = -triu(E) + midrad(0,delta);
19     setround(0)
```

Throughout the code we use from line 2 rounding upwards, i.e., the computed floating-point result is always an upper bound of the true result (see Section 1). That holds true for vector and matrix operations as well. Thus, for example, the sum in the computation of `DeltaL` in line 4 is an upper bound of the row sums of absolute values of the strictly lower triangular part of E. The code is simplified in the sense that in lines $5, 9, 13$, and 17 it is assumed that the upper bounds for the norms are

strictly less than 1. Then the denominator in line 5 is negative and larger or equal to $\|E\|_\infty - 1$, so that the negative of the division produces a correct upper bound `DeltaL`. Similar arguments show the correctness of the code in lines 9, 13, and 17. For an interval matrix $\mathbf{E}$, real or complex, the assertions hold true for every $I + \tilde{E} \in I + \mathbf{E}$.

For given A, the product $U = L^{-1}A$ can be enclosed by $(I+\texttt{LE})\backslash A$ or $A+\texttt{LinvE}\cdot A$, and similarly $L = AU^{-1}$ can be enclosed[2] by $A/(I + \texttt{LU})$ or $A + A \cdot \texttt{LinvU}$. For not too large $\|E\|$ the latter formulation is advantageous. To that end we compare the relative error of the former and the latter method. For different values of ε we choose perturbations $I + E$ with $\|E\| = \varepsilon$ and $A \in M_{100}$ with fixed condition number 10^8. The results are shown in Figure 2 for L and U in the left and right graph, respectively.

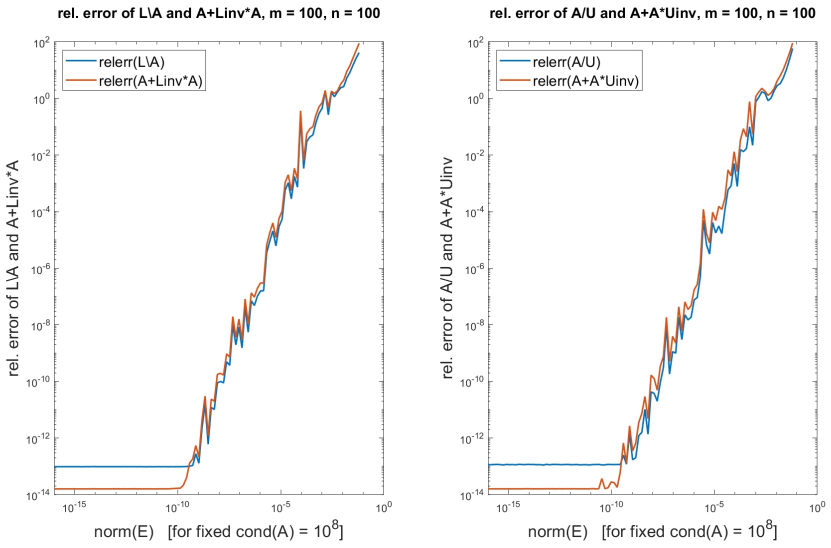

FIG. 2. *Relative error of $L\backslash A$ vs. $A + Linv \cdot A$ and A/U vs. $A + A \cdot Uinv$*

As can be seen both the results for $L^{-1}A$ and AU^{-1} improve for perturbations E of the identity matrix up to 10^{-9}. For larger E there is not much difference.

3.2. LU decomposition of general $A \in M_{m,n}$ with $m = n$. Let $A \in M_n$ be given. We first compute approximate factors $\tilde{L}$ and $\tilde{U}$ by an LU decomposition with partial pivoting and permute the rows of A accordingly.

Let $\tilde{X}_L \approx \tilde{L}^{-1}$ and $\tilde{X}_U \approx \tilde{U}^{-1}$ be approximate preconditioners, so that $I_E := \tilde{X}_L A \tilde{X}_U$ is a perturbed identity matrix. In MATLAB we may compute $\tilde{X}_L$ by `inv(Ls)` as a left inverse, however, use `eye(n)/Us` to compute $\tilde{X}_U$ as a right inverse, cf. [11]. Then the uniqueness of the LU decomposition and $I_E = L_E U_E$ imply

$$L = \tilde{X}_L^{-1} L_E \quad \text{and} \quad U = U_E \tilde{X}_U^{-1} \ .$$

Note that in Algorithm 3.2 (`LU_E`) the offsets `LE` and `UE` of L_E and U_E to the identity matrix are computed, respectively. The computational effort is $\mathcal{O}(n^3)$ operations.

[2]Recall that A/B is the MATLAB notation for AB^{-1}.

TABLE 1
Condition number and sensitivity of the LU-factors for $n = 100$ and different condition numbers

cond(A)	10^2	10^5	10^8	10^{11}	10^{14}
cond(L)	$1.4 \cdot 10^2$	$1.4 \cdot 10^2$	$1.4 \cdot 10^2$	$1.4 \cdot 10^2$	$1.4 \cdot 10^2$
cond(U)	$2.7 \cdot 10^2$	$4.8 \cdot 10^4$	$2.4 \cdot 10^7$	$1.7 \cdot 10^{10}$	$1.3 \cdot 10^{13}$
sensitivity(L)	$5.2 \cdot 10^{-15}$	$7.7 \cdot 10^{-13}$	$3.4 \cdot 10^{-10}$	$2.0 \cdot 10^{-7}$	$1.4 \cdot 10^{-4}$
sensitivity(U)	$3.3 \cdot 10^{-15}$	$5.2 \cdot 10^{-15}$	$6.9 \cdot 10^{-15}$	$8.1 \cdot 10^{-15}$	$9.4 \cdot 10^{-15}$

It is well known that the condition number of A moves into the U-factor, i.e., the factor L will be well conditioned whereas cond$(U) \sim$ cond(A). Thus we might expect the factor U to be sensitive to perturbations in A whereas L is not so sensitive. However, the opposite is true, see Table 1 for square matrices of dimension $n = 100$. As can be seen the condition number of L is small, that of U is of the order of cond(A). For the sensitivity displayed in the last two rows we perturb the matrix A into $\tilde{A}$ by changing each entry of A randomly by 1 bit in the mantissa and display $\|\tilde{L} - L\|/\|L\|$, and for U correspondingly. As can be seen, both L and U are insensitive for small condition number, however, for ill-conditioned A a perturbation of the last bit of A changes L relatively by about 10^{-4}, whereas U changes only about in the last bit. This is in accordance with our rule of thumb (2.1). The numbers are the median of 1000 samples.

The reasoning for this rule of thumb (2.1) in Section 2 relied on the relation of the magnitude of the elements of U. It was also mentioned in a footnote that this relation is often not true for ill-conditioned matrices generated randomly by `sprand` with density 1, i.e., dense matrices. All entries of the factor U of such matrices are often not far from 1 in magnitude except one or two very small entries on the diagonal, often not U_{nn}. We measured the sensitivity of L and U for square matrices of dimension $n = 100$ similar to Table 1 and display the results in Table 2.

TABLE 2
Condition number and sensitivity of the LU-factors for $n = 100$ and matrices generated by `sprand`

cond(A)	10^2	10^5	10^8	10^{11}	10^{14}
cond(L)	$7.0 \cdot 10^1$	$7.2 \cdot 10^1$	$7.3 \cdot 10^1$	$7.4 \cdot 10^1$	$7.6 \cdot 10^1$
cond(U)	$5.4 \cdot 10^2$	$4.1 \cdot 10^5$	$3.9 \cdot 10^8$	$3.8 \cdot 10^{11}$	$4.0 \cdot 10^{14}$
sensitivity(L) median	$2.0 \cdot 10^{-15}$	$1.7 \cdot 10^{-13}$	$2.4 \cdot 10^{-10}$	$3.4 \cdot 10^{-8}$	$2.3 \cdot 10^{-4}$
sensitivity(L) mean	$2.3 \cdot 10^{-15}$	$6.0 \cdot 10^{-13}$	$5.7 \cdot 10^{-10}$	$4.3 \cdot 10^{-7}$	$6.9 \cdot 10^{-4}$
sensitivity(U) median	$1.5 \cdot 10^{-15}$	$3.5 \cdot 10^{-15}$	$5.9 \cdot 10^{-15}$	$7.5 \cdot 10^{-15}$	$7.9 \cdot 10^{-15}$
sensitivity(U) mean	$1.7 \cdot 10^{-15}$	$1.5 \cdot 10^{-13}$	$5.6 \cdot 10^{-11}$	$1.2 \cdot 10^{-7}$	$1.4 \cdot 10^{-4}$

As before the input matrices were perturbed entrywise and randomly by one bit, and the decompositions were performed using the multiple precision package [2] to avoid distortion of the sensitivity by rounding errors. For the median of the sensitivities of L and U there is not too much difference to Table 1. The mean and median of the sensitivity of L are similar[3], so that seems to support (2.1). However, the mean of the sensitivities of U is larger than the median. Thus a few entries of U seem sensitive to perturbations, but the majority is not. So basically the rule of thumb (2.1) seems still applicable, but we don't have an explanation for that behavior.

The quality of the bounds depend on how close I_E is to the identity matrix, i.e., for

[3] All medians and means in Table 1 are all similar, so only the medians are displayed.

$I + E := I_E$ we want $\|E\|$ to be as small as possible. The median of the relative errors of all inclusion components of L and of U is displayed in Figure 3 from left to right, respectively, for condition numbers from 1 to 10^{15}. We first use interval arithmetic for the computation of $I_E := \tilde{X}_L A \tilde{X}_U$ and for L and U and display the relative errors in red. As expected, the error grows with the condition number. For condition numbers close to 10^{15} the inclusions are still accurate to about 8 to 10 decimal figures. The spikes for very large condition number indicate that the verification failed.

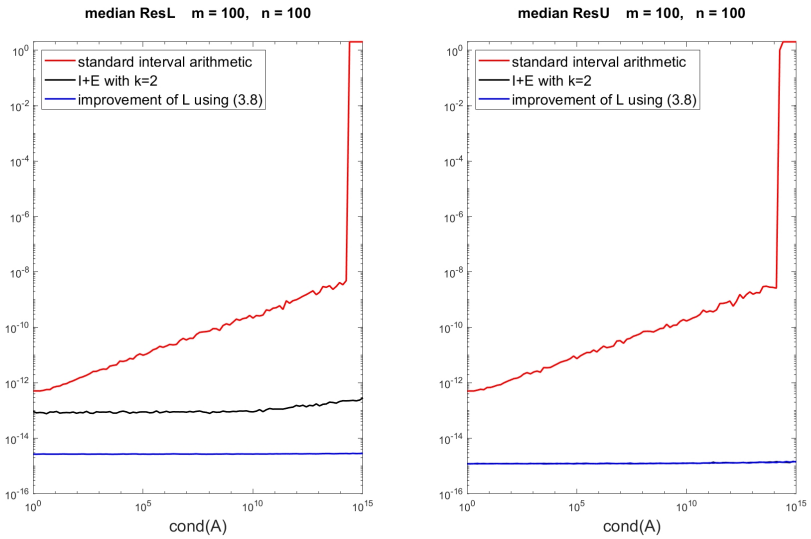

FIG. 3. *Norm of error bounds for the L- and U-factor for different condition numbers*

To achieve more accurate bounds we compute I_E as $\tilde{X}_L(A\tilde{X}_U)$ and use for both products two-fold precision, equivalent to double-double precision. In the legend of Figure 3 this is indicated by $k = 2$. The result is displayed in Figure 3 in black, where in the right picture the black curve is identical to the blue curve to be defined. As can be seen the accuracy of U is now close to maximal precision equivalent to 16 decimal places for all condition numbers, and for L the accuracy is a little bit less. That corresponds to Table 1, i.e., we expect better inclusions for U rather than that of L. In order to improve the accuracy of the L-factor, we use $L = LU\tilde{X}_U U_E^{-1}$ implying

$$L = A\tilde{X}_U U_E^{-1} \quad \text{and} \quad U = U_E \tilde{X}_U^{-1} \tag{3.8}$$

for the square case $m = n$. The product $A\tilde{X}_U$ is computed in doubled precision, the result is displayed as the blue curve in Figure 3. Now for all condition numbers and all entries of the factors L and U the bounds are of almost maximal accuracy, except for $\mathrm{cond}(A) = 10^{15}$ where the verification failed.

3.3. LU decomposition of general $A \in M_{m,n}$ with $m > n$. Let $A \in M_{m,n}$ be given with $m > n$. We first compute an approximate LU decomposition with partial pivoting and permute the rows of A accordingly. Thus we may assume that the upper square block A_n has an LU decomposition. Following the approach discussed at the

beginning of this section

$$A = LU = \begin{pmatrix} A_n \\ A_{\overline{n}} \end{pmatrix} = \begin{pmatrix} L_n \\ L_{\overline{n}} \end{pmatrix} U$$

so that

$$X_L := \begin{pmatrix} L_n^{-1} & O_{n,m-n} \\ -L_{\overline{n}} L_n^{-1} & I_{m-n} \end{pmatrix},\ X_U := U^{-1}$$

implies

$$X_L A X_U = \begin{pmatrix} I_n \\ O_{m-n,n} \end{pmatrix}.$$

We compute approximations $\tilde{X}_L \approx X_L$ and $\tilde{X}_U \approx X_U$ using an approximate LU decomposition $A \approx \tilde{L}\tilde{U}$, so that $I_E := \tilde{X}_L A \tilde{X}_U$ is a perturbed identity matrix. Then $I_E = L_E U_E$ implies

$$\tilde{X}_L := \begin{pmatrix} P & O_{n,m-n} \\ Q & I_{m-n} \end{pmatrix} \quad \Rightarrow \quad L = \begin{pmatrix} P^{-1} & O_{n,m-n} \\ -QP^{-1} & I_{m-n} \end{pmatrix} L_E$$

and $U = U_E \tilde{X}_U^{-1}$. The computational effort is $\mathcal{O}(P^2 p)$ operations for $P = \max(m,n)$ and $p = \min(m,n)$.

TABLE 3
Condition number and sensitivity of the LU-factors for $m = 200$ and $n = 100$

cond(A)	10^2	10^5	10^8	10^{11}	10^{14}
cond(L)	$8.6 \cdot 10^1$	$8.5 \cdot 10^1$	$8.6 \cdot 10^1$	$8.6 \cdot 10^1$	$8.5 \cdot 10^1$
cond(U)	$3.4 \cdot 10^2$	$9.1 \cdot 10^4$	$5.5 \cdot 10^7$	$4.1 \cdot 10^{10}$	$3.5 \cdot 10^{13}$
sensitivity(L_1)	$6.4 \cdot 10^{-15}$	$1.3 \cdot 10^{-12}$	$6.6 \cdot 10^{-10}$	$4.3 \cdot 10^{-7}$	$3.1 \cdot 10^{-4}$
sensitivity(L_2)	$1.3 \cdot 10^{-14}$	$3.6 \cdot 10^{-12}$	$2.3 \cdot 10^{-9}$	$1.8 \cdot 10^{-6}$	$1.5 \cdot 10^{-3}$
sensitivity(U)	$2.7 \cdot 10^{-15}$	$4.5 \cdot 10^{-15}$	$5.9 \cdot 10^{-15}$	$7.1 \cdot 10^{-15}$	$8.5 \cdot 10^{-15}$

In Table 3 we show the sensitivity of the upper square block L_n, the remaining part $L_{\overline{n}}$ of L and of the U-factor, again the median over 1000 samples. Similar to the square case and as predicted by (2.1), with increasing condition number the components of the L-factor are getting sensitive to perturbations in A, while those of U are not. Hence, as for square A, we expect less accurate inclusions of L.

The median of the relative errors of all inclusion components of the upper square part L_n of L, the remaining part $L_{\overline{n}} = L(n+1:m,:)$ and of U is displayed in Figure 4 from left to right, respectively, for condition numbers from 1 to 10^{15}. As before we first use interval arithmetic for the computation of $I_E := \tilde{X}_L A \tilde{X}_U$ and to compute L and U and display the relative errors in red. As expected, the error grows with the condition number. For condition numbers close to 10^{15} the inclusions are still accurate to about 8 to 10 decimal figures.

To achieve more accurate bounds we compute I_E as $\tilde{X}_L(A\tilde{X}_U)$ and use for both products two-fold precision, equivalent to double-double precision. The result is displayed in Figure 4 in black. As can be seen the accuracy of U is now close to maximal precision equivalent to 16 decimal places for all condition numbers, for the upper part of L it improved significantly, and for the lower part of L the accuracy decreases from condition number 10^9.

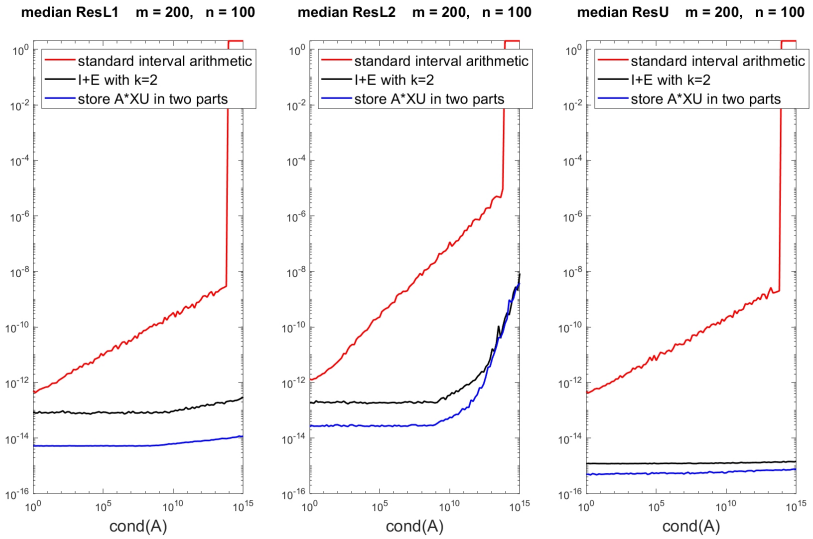

FIG. 4. *First methods: Norm of error bounds for the L- and U-factor based on A*

In order to obtain flat curves in all three pictures, i.e., close to maximal accuracy for all components of the L- and the U-factor, we compute the product $A\tilde{X}_U$ again in doubled precision but store it in two parts $C_1 + C_2$, and then compute $\tilde{X}_L C_1 + \tilde{X}_L C_2$ in doubled precision but store it in one matrix I_E. The result displayed in blue in Figure 4 is better than before, however, there is still a growth of the errors of the lower part of L from condition number 10^9.

The following alternative approach is faster and better. To that end we use an approximate LU decomposition $A_n \approx \tilde{L}_n \tilde{U}$ of the upper left square block of A. For approximations $\tilde{X}_{L_n} \approx \tilde{L}_n^{-1}$ and $\tilde{X}_U \approx \tilde{U}^{-1}$ let $I_E = \tilde{X}_{L_n} A_n \tilde{X}_U = L_E U_E$. Then

$$U = U_E \tilde{X}_U^{-1} \quad \text{and} \quad L = A\tilde{X}_U U_E^{-1} \tag{3.9}$$

using $L = LU\tilde{X}_U U_E^{-1}$. Thus, for $m \geqslant n$ we use the same formula as (3.8) for the square case. The computational effort is $\mathcal{O}(Pp^2)$ operations.

As before display the median of the relative errors of all components of the upper square part L_n of L, the lower part of L and of U in Figure 5. The red curves are the results when using interval arithmetic to compute $I_E := \tilde{X}_L A \tilde{X}_U$, L and U and are similar to those before. Using extra precision to compute $\tilde{X}_L(A\tilde{X}_U)$ is shown in black and is for both parts of L better than before. Finally, for the blue curve we used doubled precision to compute $A\tilde{X}_U$ and stored the result in two parts, which are then multiplied by $\tilde{X}_L$. That last method yields for all condition numbers and all entries of the factors L and U bounds of almost maximal accuracy.

3.4. LU decomposition of general $A \in M_{m,n}$ with $m < n$. Let $A \in M_{m,n}$ with $m < n$ be given. Now the partial pivoting of an approximate LU decomposition may take only the left square block A_m into account. Therefore, we first compute an approximate LU decomposition with partial pivoting of A^T and permute the columns of A accordingly, followed by the computation an approximate LU decomposition of A with partial pivoting and permute the rows of A accordingly. We may assume that

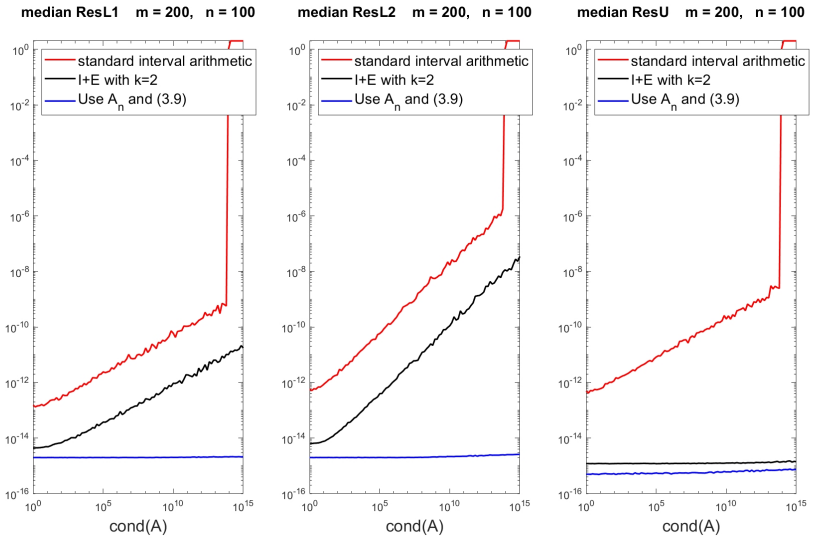

FIG. 5. *Second methods: Norm of error bounds for the L- and U-factor based on A_n*

the left square block A_m has an LU decomposition. Then

$$A = LU = \left(\begin{array}{cc} A_m & A_{\overline{m}} \end{array} \right) = L \left(\begin{array}{cc} U_m & U_{\overline{m}} \end{array} \right)$$

and

$$X_L := L^{-1}, \; X_U = \left(\begin{array}{cc} U_m^{-1} & -U_m^{-1} U_{\overline{m}} \\ O_{n-m,m} & I_{n-m} \end{array} \right)$$

implies $X_L A X_U = \left(\begin{array}{cc} I_m & O_{m,n-m} \end{array} \right)$. We compute approximations $\tilde{X}_L \approx X_L$ and $\tilde{X}_U \approx X_U$ using an approximate LU decomposition $A \approx \tilde{L}\tilde{U}$, so that again $I_E := \tilde{X}_L A \tilde{X}_U$ is a perturbed identity matrix. Then $I_E = L_E U_E$ implies

$$\tilde{X}_U := \left(\begin{array}{cc} P & Q \\ O_{n-m,m} & I_{n-m} \end{array} \right) \quad \Rightarrow \quad L = \tilde{X}_L^{-1} L_E$$

and

$$U = U_E \left(\begin{array}{cc} P^{-1} & -P^{-1}Q \\ O_{n-m,m} & I_{n-m} \end{array} \right) .$$

The computational effort is $\mathcal{O}(P^2 p)$ operations for $P = \max(m,n)$ and $p = \min(m,n)$.

The sensitivity of the L-factor, the left square block U_m and the remaining of the U-factor is displayed in Table 4 and is, as predicted in (2.1), similar to square A or the case $m > n$. Again we expect it to be more difficult to obtain narrow inclusions of L.

Computational results are shown in Figure 6. The median of relative errors of all components of the L-factor, the left square part U_m and the remaining of the U-factor are shown from left to right. The color coding is as in the previous subsections, i.e.,

TABLE 4
Condition number and sensitivity of the LU-factors for $m = 100$ and $n = 200$

cond(A)	10^2	10^5	10^8	10^{11}	10^{14}
cond(L)	$1.4 \cdot 10^2$	$1.4 \cdot 10^2$	$1.4 \cdot 10^2$	$1.4 \cdot 10^2$	$1.4 \cdot 10^2$
cond(U)	$2.7 \cdot 10^2$	$4.8 \cdot 10^4$	$2.4 \cdot 10^7$	$1.7 \cdot 10^{10}$	$1.3 \cdot 10^{13}$
sensitivity(L)	$1.2 \cdot 10^{-14}$	$2.5 \cdot 10^{-12}$	$1.3 \cdot 10^{-9}$	$8.4 \cdot 10^{-7}$	$6.0 \cdot 10^{-4}$
sensitivity(U_1)	$3.8 \cdot 10^{-15}$	$5.4 \cdot 10^{-15}$	$7.0 \cdot 10^{-15}$	$8.3 \cdot 10^{-15}$	$9.6 \cdot 10^{-15}$
sensitivity(U_2)	$9.4 \cdot 10^{-15}$	$1.3 \cdot 10^{-14}$	$1.7 \cdot 10^{-14}$	$1.9 \cdot 10^{-14}$	$2.2 \cdot 10^{-14}$

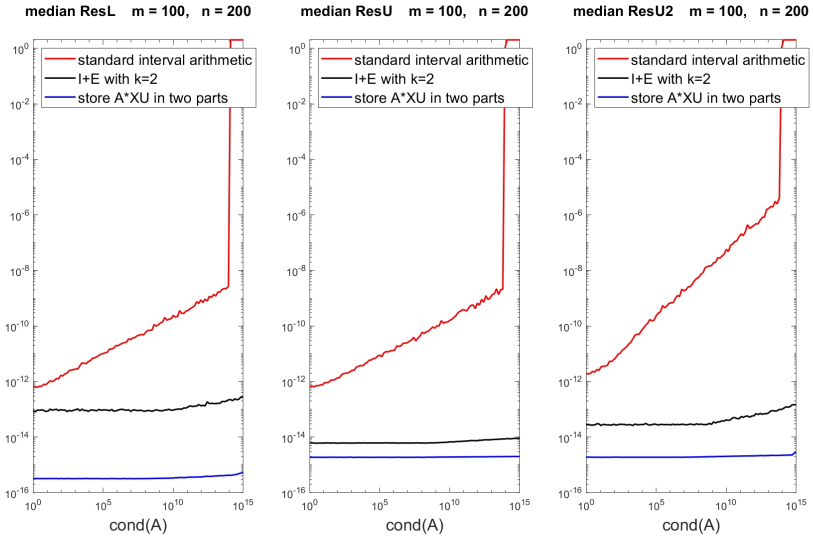

FIG. 6. *First methods: Norm of error bounds for the L- and U-factor based on A*

for the red curve only standard interval arithmetic was used and the expected growth with the condition number can be seen.

For the black curve the two products in $\tilde{X}_L(A\tilde{X}_U)$ are computed in doubled precision. Supposedly, the results are better than for the case $m \geqslant n$ because we performed initially two approximate LU decompositions to identify the permutations of columns and rows.

In order to obtain flat curves close to maximal accuracy for all components of both the L- and the U-factor, we compute $C_1 + C_2 = A\tilde{X}_U$ in doubled precision and use two matrices to store the result, and then compute $\tilde{X}_L C_1 + \tilde{X}_L C_2$ in doubled precision but store the result in one matrix I_E.

Now the results are close to maximal accuracy, shown in blue, but the computing time of $\mathcal{O}(P^2 p)$ operations can be improved into $\mathcal{O}(Pp^2)$ operations. Similar as before we use an approximate LU decomposition $A_m \approx \tilde{L}\tilde{U}_m$ of the left square block of A. For approximations $\tilde{X}_L \approx \tilde{L}^{-1}$ and $\tilde{X}_{U_m} \approx \tilde{U}_m^{-1}$ let $I_E = \tilde{X}_L A_m \tilde{X}_{U_n} = L_E U_E$. Then

$$L = \tilde{X}_L^{-1} L_E \quad \text{and} \quad U = L_E^{-1} \tilde{X}_L A \tag{3.10}$$

using $U = L_E^{-1}\tilde{X}_L LU$ for the latter equality. Now the computational effort is $\mathcal{O}(Pp^2)$ operations.

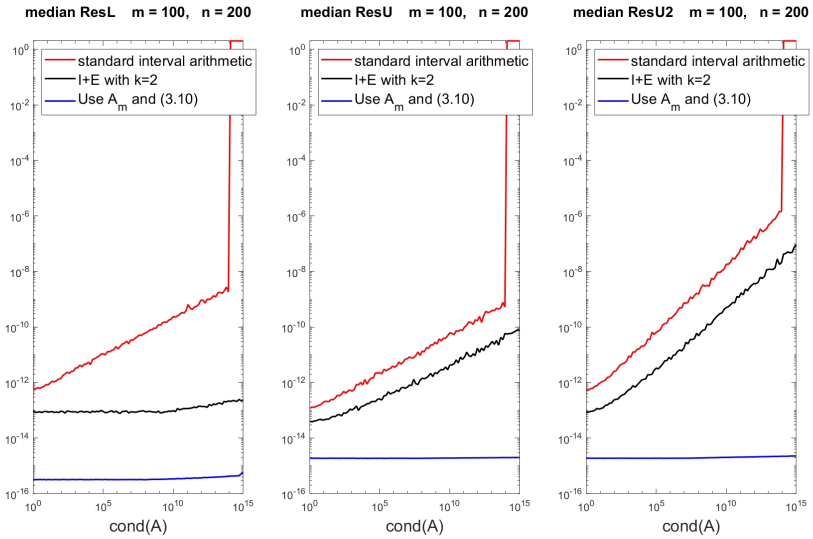

FIG. 7. *Second methods: Norm of error bounds for the L- and U-factor based on* A_m

Computational results are shown in Figure 7. Using the same color coding the main difference is in the black curve, computing the two products $\tilde{X}_L(A\tilde{X}_U)$ in doubled precision but storing either result in one matrix. The results in Figure 6 are better due to the use of the right part $U_{\overline{m}}$ in the computations.

The third and best method, shown in blue in Figure 7, is to store $A\tilde{X}_U$ in two matrices and proceed as before. The resulting curves are flat and close to the relative rounding error unit 10^{-16} for all condition numbers and all components of the L- and the U-factor.

4. Cholesky decomposition. As for the Cholesky decomposition of a symmetric positive definite matrix $A \in M_n$ we have all necessary ingredients. We would like to stress that A being positive definite is not an assumption, because if so, it would have to be verified *before* starting the computation. In contrast, the property is verified *a posteriori*, i.e., if successful the matrix has been proved to be positive definite.

To compute bounds for the Cholesky factor, we first use an approximate Cholesky factor $\tilde{G}$ to precondition A into $I_E := X_G^T A X_G$ for $X_G \approx \tilde{G}^{-1}$. We discussed in Section 3.1 how to obtain verified inclusions for the LU decomposition $I_E = L_E U_E$. That includes in particular the diagonal D of U_E. The uniqueness of the LU and Cholesky decomposition implies that $G_E = D^{1/2} L_E^T$ is the Cholesky factor of I_E. Hence, an inclusion of the Cholesky factor may be computed by

$$G_E^T G_E = I_E = X_G^T A X_G \quad \Rightarrow \quad G = G_E X_G^{-1} = D^{1/2} L_E^T X_G^{-1} \ . \tag{4.1}$$

The computational effort is $\mathcal{O}(n^3)$ operations.

We first show the median of the sensitivity of the Cholesky factor for 1000 samples in Table 5. The condition number of G is, of course, the square root of that of A, and the sensitivity corresponds to that predicted in (2.1). In some way it seems the geometric mean between the sensitivity of the L- and the U-factor.

TABLE 5
Condition number and sensitivity of the Cholesky factor for different condition numbers

cond(A)	10^2	10^5	10^8	10^{11}	10^{14}
cond(G)	$1.0 \cdot 10^1$	$3.2 \cdot 10^2$	$10.0 \cdot 10^3$	$3.2 \cdot 10^5$	$10.0 \cdot 10^6$
sensitivity(G)	$6.8 \cdot 10^{-16}$	$1.8 \cdot 10^{-14}$	$4.9 \cdot 10^{-13}$	$1.4 \cdot 10^{-11}$	$4.0 \cdot 10^{-10}$

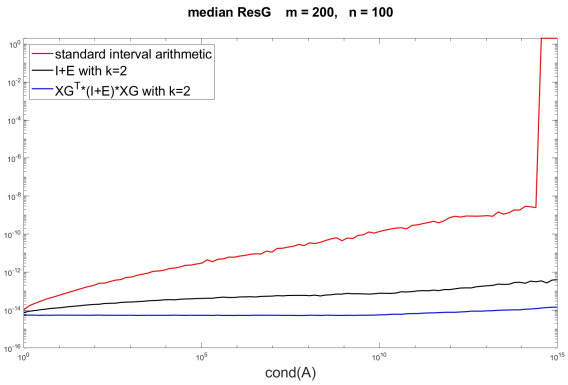

FIG. 8. *Norm of error bounds for the Cholesky factor for different condition numbers*

The computational results for (4.1) are shown in Figure 8. We display the median of the relative errors of all components of the inclusion of the Cholesky factor for different condition numbers. The color coding is similar to the previous sections. For the red curve we use interval arithmetic to compute inclusions of $I_E = X_G^T A X_G$ and for the inclusion of G according to (4.1). As expected, errors grow with the condition number but still guaranteeing some 8 correct decimal figures up to condition number $3 \cdot 10^{14}$ and failure above.

The black curve shows the results for computing $X_G^T A X_G$ in doubled precision. The quality of the inclusion is better and there is no failure.

Finally, we compute $C_1 + C_2 = AX_G$ in doubled precision with two results and $X_G^T C_1 + X_G^T C_2$ again in doubled precision but with one result I_E. Now, shown in blue, for all condition numbers all components of the Cholesky factor are enclosed with almost maximal accuracy.

5. QR decomposition. Assume $A \in M_{m,n}$ with $m \geqslant n$ to be given with full rank. Then there is a unique QR decomposition with orthonormal columns and upper triangular R with non-negative diagonal entries [14, Theorem 2.1.14]. Consider

$$A := \begin{pmatrix} 1 & e \\ 1 & e + \varphi e^2 \end{pmatrix}$$

with

$$Q := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -\varphi \\ 1 & \varphi \end{pmatrix} \quad \text{and} \quad R := \frac{1}{\sqrt{2}} \begin{pmatrix} 2 & (2 + \varphi e)e \\ 0 & e^2 \end{pmatrix}.$$

Then $A = QR$ for $e, \varphi \in \mathbb{R}$, and for $\varphi = -1$ and $\varphi = 1$ this is the unique QR decomposition of $A(\varphi)$ for any $e \in \mathbb{R}$. Hence, the computation of the QR decomposition is ill-posed at $e = 0$ because an arbitrary small perturbation causes a finite change in

Q. As a consequence, a verification method is only applicable to matrices with full rank.

Let $A \approx \tilde{Q}\tilde{R}$ be an approximate "economy-size" QR decomposition, i.e., $\tilde{Q} \in M_{m,n}$ and $\tilde{R} \in M_n$. For $\tilde{X}_R \approx \tilde{R}^{-1}$ we expect $C := A\tilde{X}_R$ to be close to unitary, so that C^TC will be a small perturbation of the identity matrix I_n.

In fact, $C^TC \approx \tilde{X}_R^T A^T A \tilde{X}_R =: I + E$ is the same as preconditioning A^TA by the inverse of the approximate Cholesky factor $\tilde{R}$ of A^TA. However, the formulation $C^TC \approx (\tilde{X}_R^T A^T) \cdot (A\tilde{X}_R)$ avoids to form the matrix A^TA with squared condition number $\mathrm{cond}(A)^2$.

We use the method described in the previous section to compute an inclusion G_E of the Cholesky factor of $I + E$, so that $\tilde{X}_R^T A^T A \tilde{X}_R = I + E = G_E^T G_E$. Hence $R := G_E \tilde{X}_R^{-1}$ is the Cholesky factor of A^TA, which in turn is the R-factor of the QR decomposition of A. An inclusion of the economy-size Q-factor is obtained by $Q_1 = AR^{-1}$ provided that R is non-singular. A second possibility is to use $Q_1 = A\tilde{X}_R G_E^{-1}$.

For the full-size QR-factors note that $Q = (Q_1\ Q_2)$ where Q_2 is the orthogonal complement of Q_1 and, provided that A has full rank, a basis for the null space of A^*. In [17] several methods are discussed to compute an inclusion of a basis of the null space of a rectangular matrix. For $\tilde{Q}_2$ denoting an approximation of the orthogonal complement of Q_1, the solution X of the square linear system

$$\begin{pmatrix} A^* \\ \alpha\tilde{Q}_2^* \end{pmatrix} X = \begin{pmatrix} O_{n,m-n} \\ \alpha I_{m-n} \end{pmatrix} \tag{5.1}$$

does the job [17], i.e., X spans the orthogonal complement Q_2 of Q_1. If $\mathbf{A}$ is an interval matrix, then this is true for every $A \in \mathbf{A}$. We choose α within $[\sigma_n(A), \sigma_1(A)]$ to ensure that the condition number of the system matrix in (5.1) is equal to that of A.

We can expect X to be numerically unitary, but not mathematically. The following lemma from [34] estimates the distance to an orthonormal basis.

LEMMA 5.1. *Let $X, Y \in M_{m,n}$ with $m \geqslant n$ be given. Define $\alpha := \|I - X^*X\|$ and $\delta := \|X - Y\|$. Let $\mathcal{V}$ be an n-dimensional subspace of $\mathbb{K}^m$ that contains all columns of Y. Then there exists $Q \in M_{m,n}$ with $Q^*Q = I$ whose columns span $\mathcal{V}$ and*

$$\|Q - X\| \leqslant \alpha + \sqrt{2}\delta.$$

The bound is sharp. Note that the bound remains true even if $\alpha \geqslant 1$, although that may not be very useful. In our practical application α is of the order of the relative rounding error unit $\mathbf{u} \approx 10^{-16}$.

The application of Lemma 5.1 is as follows. A very good approximate solution to (5.1) is $X := \tilde{Q}_2$, for which also $\alpha := \|I - X^*X\|$ is close to the relative rounding error unit. Define Y to be the true solution of (5.1). An inclusion of Y is computed by verification methods. In fact, the inclusion will be of the form $\tilde{Q}_2 + \Delta$ for an interval matrix Δ with small norm [32], so that $\delta = \|\Delta\|$. It follows that $\tilde{Q}_2 \pm \delta$ is an inclusion of the orthogonal complement Q_2 to Q_1. Hence, $(Q_1\ Q_2)$ is the full Q-factor of the QR decomposition of A, where the full R-factor is $\begin{pmatrix} R \\ O_{m-n,n} \end{pmatrix}$.

For $A \in M_{m,n}$ with $m < n$ we compute inclusions of the (full) QR decomposition of the square matrix A_m, so that $A_m = QR_m$ implies $R = Q^*A$. The computational effort for the inclusion is the same as to compute an approximate decomposition.

To judge the computational results we first check on the median of the sensitivity of R and the two parts of the factor Q of $A \in M_{200,100}$ for 1000 samples. The results are displayed in Table 6. The factor Q is perfectly conditioned, however, sensitive to perturbations in A. The factor R has the same condition number as A, but is insensitive to small perturbations in A in accordance with (2.1). The corresponding data for $m < n$ is completely similar, only Q is sensitive to perturbations in A. Thus we may expect more problems in the computation of narrow bounds of Q.

TABLE 6
Sensitivity of the two parts of the factors Q and R for different condition numbers

cond(A)	10^2	10^5	10^8	10^{11}	10^{14}
sensitivity(Q_1)	$1.2 \cdot 10^{-14}$	$6.0 \cdot 10^{-12}$	$4.5 \cdot 10^{-9}$	$3.7 \cdot 10^{-6}$	$3.3 \cdot 10^{-3}$
sensitivity(Q_2)	$1.7 \cdot 10^{-14}$	$8.2 \cdot 10^{-12}$	$6.1 \cdot 10^{-9}$	$5.0 \cdot 10^{-6}$	$4.5 \cdot 10^{-3}$
sensitivity(R)	$4.8 \cdot 10^{-16}$	$5.6 \cdot 10^{-16}$	$6.2 \cdot 10^{-16}$	$7.0 \cdot 10^{-16}$	$7.8 \cdot 10^{-16}$

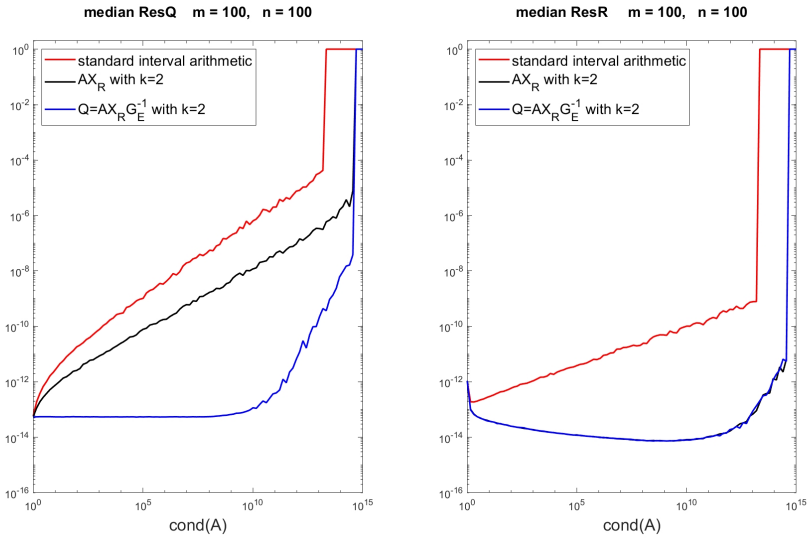

FIG. 9. *Norm of error bounds for the Q- and R-factor for different condition numbers for $m = n$*

Next we show the median of the relative errors of all entries of the inclusions of Q and R. We start with a square matrix $A \in M_n$. We first compute $C = A\tilde{X}_R$ as well the inclusions $G_E\tilde{X}_R^{-1}$ and $Q = AR^{-1}$ in standard interval arithmetic. The result for different condition numbers is the red curve in Figure 9. We observe an increase of the relative errors proportional to the condition number, and as predicted less accurate bounds for Q.

Secondly, we compute an inclusion $C = A\tilde{X}_R$ with doubled precision with one output result. The product C^*C is computed in doubled precision as well, otherwise we use standard interval arithmetic. The result is the black curve in Figure 9. It is much better than before, in particular the inclusion of R.

Finally, we use the second possibility $Q_1 = A\tilde{X}_R G_E^{-1}$ for the inclusion of Q_1, where the first product $Q_1 = A\tilde{X}_R$ is computed in doubled precision. The result is

shown as the blue curve in Figure 9, where the black and blue curves are practically identical for R.

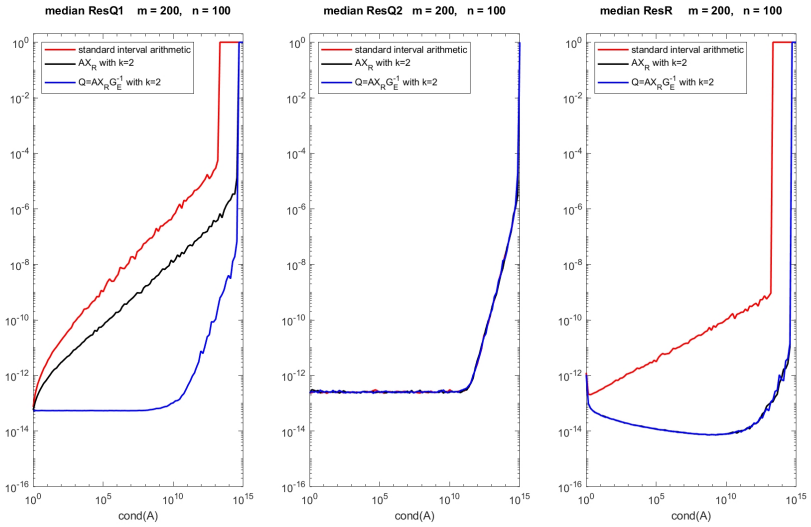

FIG. 10. *Norm of error bounds for the Q- and R-factor for different condition numbers for $m > n$*

The results for $A \in M_{m,n}$ with $m > n$ are shown in Figure 10. They look quite similar to those in Figure 9 for square A. In particular the quality of the two parts of Q shows no surprises. That is also true for the case $m < n$, so we omit to show that graph.

Until now we refrained from giving computing times of our inclusion methods, mainly because those are essentially dominated by the interpretation overhead in MATLAB. At least for one example, the QR decomposition, we show the ratio of computing time compared to the built-in MATLAB routine. For most problems we gave three inclusion methods, as for example in Figure 9. However, the timing is not too different, therefore we give only the time ratio for our best method compared to `qr`. As has been said this ratio is biased by the interpretation overhead, and in particular by the fact that our inclusion methods aim on highly accurate results. Therefore we also show the median relative error of the MATLAB result.

For the following Table 7 we generated real and complex square random matrices with condition number 10^{10}. From left the dimension and ratio of computing time of our best inclusion method together with the median relative error of all components of the floating-point approximation produced by `[Q,R] = qr(A);` are displayed, for real matrices on the left and for complex matrices on the right.

As can be seen the verification method (in pure MATLAB code) is significantly slower than the built-in `qr`, but, according to Figure 9 and Table 7 also more accurate. Note that the accuracy of the MATLAB approximation is different from the sensitivity as displayed in Table 6.

We finally show the same table for rectangular matrices $A \in M_{m,n}$. We set $m := 2n$ and display the results in Table 8.

The median relative error of Q computed by `qr` is slightly weaker than for square matrices, more according to Table 6. Otherwise there is not too much difference in

TABLE 7
Ratio of computing times to MATLAB's `qr` *for different condition numbers*

	real input			complex input		
n	time ratio	Q	R	time ratio	Q	R
100	49.3	$1.2 \cdot 10^{-11}$	$6.8 \cdot 10^{-14}$	47.1	$1.2 \cdot 10^{-11}$	$6.8 \cdot 10^{-14}$
200	33.5	$1.4 \cdot 10^{-11}$	$8.3 \cdot 10^{-14}$	44.2	$1.3 \cdot 10^{-11}$	$7.9 \cdot 10^{-14}$
500	64.6	$6.9 \cdot 10^{-10}$	$4.5 \cdot 10^{-11}$	61.4	$1.2 \cdot 10^{-11}$	$1.1 \cdot 10^{-13}$
1000	58.3	$2.6 \cdot 10^{-9}$	$1.7 \cdot 10^{-8}$	48.2	$1.4 \cdot 10^{-11}$	$1.8 \cdot 10^{-13}$

TABLE 8
Ratio of computing times to MATLAB's `qr` *for different condition numbers*

	real input			complex input		
n	time ratio	Q	R	time ratio	Q	R
100	50.5	$1.7 \cdot 10^{-8}$	$4.4 \cdot 10^{-14}$	63.3	$2.3 \cdot 10^{-8}$	$5.6 \cdot 10^{-14}$
200	45.0	$1.2 \cdot 10^{-8}$	$5.2 \cdot 10^{-14}$	67.3	$1.9 \cdot 10^{-8}$	$6.6 \cdot 10^{-14}$
500	66.9	$9.6 \cdot 10^{-9}$	$3.5 \cdot 10^{-11}$	52.3	$1.8 \cdot 10^{-8}$	$1.0 \cdot 10^{-13}$
1000	52.3	$8.5 \cdot 10^{-9}$	$3.0 \cdot 10^{-8}$	56.6	$2.1 \cdot 10^{-8}$	$1.7 \cdot 10^{-13}$

the ratio of computing times or accuracy.

6. Eigendecomposition. A verified inclusion of an individual eigenvector to a multiple eigenvalue matrix is out of the scope of verification methods because the problem is ill-posed. In case of a non-trivial Jordan block of size k, there may be only one eigenvector which, after an arbitrarily small perturbation, changes into up to k individual eigenvectors.

Hence, the problem of computing a verified error bound for an individual eigenvector is ill-posed, as well as to certify that an eigenvalue is not simple. However, an inclusion of a cluster and/or multiple eigenvalue becomes well posed if it is separated from the remaining spectrum. Then computing a basis for the corresponding invariant subspace is well posed as well.

There are approaches to compute error bounds for one cluster of eigenvalues together with invariant subspace [32, Theorem 13.9], however, the computing time for the complete eigendecomposition by applying that method to each cluster is $\mathcal{O}(n^4)$ operations.

There are papers for computing inclusions of all eigenvalues and -vectors in $\mathcal{O}(n^3)$ operations [25] for the symmetric positive definite case and [24] for general matrices. However, the given practical implementations face some problems. The algorithms in [33] for general and [34] for Hermitian matrices for the complete eigendecomposition also require $\mathcal{O}(n^3)$ operations and are numerically stable. Of course, there are natural limitations for many large clusters. General matrices in [33] are, similar to the methods in this paper, transformed into a perturbed identity matrix, whereas the symmetric and Hermitian case in [34] is treated by generalized perturbation bounds. Both algorithms handle multiple or clustered eigenvalues as follows.

The output of either algorithm is an interval vector $\mathbf{L}$, an interval matrix $\mathbf{X}$ and a cell array μ. If a cell element consists of a single element $\{k\}$, then $\mathbf{L}_k$ is an inclusion of a simple eigenvalue and $\mathbf{X}(:,k)$ an inclusion of a corresponding eigenvector. A challenge for both algorithms is to identify clusters. To that end a threshold κ can be specified accepting eigenvalues with distance below κ to be a cluster. For $\kappa = 0$ the algorithms try, if possible, to produce individual inclusions for all eigenvalues.

If a cell element is a set μ_ℓ of indices, then the $\mathbf{L}_k$ for $k \in \mu_\ell$ are identical and contain exactly $|\mu_\ell|$ eigenvalues, where the set of columns $\mathbf{X}_k$ for $k \in \mu_\ell$ span the corresponding invariant subspace. For symmetric or Hermitian matrix, Lemma 5.1 is used to ensure that $\mathbf{X}$ contains a unitary eigenvector basis.

There is a difference between the results for symmetric/Hermitian and for general matrices. In the former case the matrix is diagonalizable so that there exist $L \in \mathbf{L}$ and $X \in \mathbf{X}$ with $AX = XL$. Thus $\mathbf{L}$ and $\mathbf{X}$ are inclusions of eigenvalues and eigenvectors. A general matrix A may not be diagonalizable. If a cell element μ_ℓ consists of more than one index, i.e., $m := |\mu_\ell| > 1$, then the identical elements $\mathbf{\Lambda} := \mathbf{L}_k$ for $k \in \mu_\ell$ contain m eigenvalues. That may be an m-fold or m distinct eigenvalues or any combination. In any case, the set of columns $\{\mathbf{X}_k : k \in \mu_\ell\}$ contains a basis Y of an invariant subspace of A. That implies existence of a matrix $M \in M_m$ with $AY = YM$, but there may be no diagonal M with this property.

However, the methods in [33] allow to compute a block diagonal interval matrix $\mathbf{D}$ with the property that there exist $D \in \mathbf{D}$ and $X \in \mathbf{X}$ with $AX = XD$. For the Schur decomposition discussed in Section 8 it would be important to include upper triangular $\mathbf{T}$ with the property that there exist $T \in \mathbf{T}$ and $X \in \mathbf{X}$ with $AX = XT$. However, that is not possible as eigenvectors of multiple eigenvalues need not be continuous, even for symmetric matrices. It was shown in [30] that the local behavior of an eigendecomposition of a matrix depending on several parameters may be quite different from the case of one parameter. The following example is adapted from [38]:

$$A(e,f) := \begin{pmatrix} 1+f & e \\ e & 1 \end{pmatrix} . \tag{6.1}$$

The two matrices $A_1 := A(e,e)$ and $A_2 := A(e,2e)$ have eigenvalues $1 + e/2 \pm e\sqrt{5}/2$ and $1 + e \pm e\sqrt{2}$, respectively. So the eigenvalues depend continuously on e at $e = 0$. However, the corresponding orthogonal eigenvectors do not depend on e and are

$$\begin{pmatrix} (1-\sqrt{5})/2 \\ 1 \end{pmatrix}, \begin{pmatrix} (1+\sqrt{5})/2 \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1+\sqrt{2} \\ 1 \end{pmatrix}, \begin{pmatrix} 1-\sqrt{2} \\ 1 \end{pmatrix}$$

for A_1 and A_2, respectively. In other words, the computation of eigenvectors for multiple eigenvalues, even for symmetric matrices, is an ill-posed problem and outside the scope of verification methods.

The inclusion of the eigendecomposition offers a simple way to compute the matrix exponential and other matrix functions, however, only for non-defective matrices.

For detailed computational tests of inclusions for the eigenproblem see [33] and [34]. Here we only mention that error bounds of high quality are computed for a general real or complex, or symmetric or Hermitian matrix. The algorithms are applicable to interval matrices $\mathbf{A}$ as well, where the inclusions are true for every $A \in \mathbf{A}$. The total computational effort is $\mathcal{O}(n^3)$ operations.

7. Singular value and polar decomposition. As for the eigendecomposition, the computation of singular vectors becomes ill-posed for multiple singular values, see example (6.1). Hence, as for the eigenproblem, inclusions for the subspaces spanning the singular vectors corresponding to a multiple or cluster of singular values is computed.

For square matrices the perturbation bounds for symmetric/Hermitian matrices can be adapted without too much difficulty. For $A \in M_{m,n}$ with $m > n$ this is still

true for the right singular vectors. However, the left singular vectors to the $m-n$ extra zero singular values need some special attention. If 0 or a numerical zero is a singular value, the singular vectors cannot be distinguished from those of the extra $m-n$ trivial singular values. They have to be clustered in order to obtain a basis for a singular subspace.

TABLE 9
Sensitivity of the singular value decomposition for different condition numbers

cond(A)	10^2	10^5	10^8	10^{11}	10^{14}
sensitivity(U)	$9.7 \cdot 10^{-14}$	$1.5 \cdot 10^{-11}$	$5.9 \cdot 10^{-9}$	$3.2 \cdot 10^{-6}$	$2.0 \cdot 10^{-3}$
sensitivity(Σ)	$7.5 \cdot 10^{-16}$	$7.2 \cdot 10^{-16}$	$6.5 \cdot 10^{-16}$	$6.0 \cdot 10^{-16}$	$3.6 \cdot 10^{-16}$
sensitivity(V)	$9.7 \cdot 10^{-14}$	$1.5 \cdot 10^{-11}$	$5.9 \cdot 10^{-9}$	$3.2 \cdot 10^{-6}$	$2.0 \cdot 10^{-3}$

As before we verify the rule of thumb (2.1) for the sensitivity of the singular values and -vectors, the results are displayed in Table 9. In the 1000 test cases we generated matrices with separated singular values because otherwise the problem to compute singular vectors becomes ill-posed. Again, the orthogonal/unitary factors become more and more sensitive for increasing condition number, whereas the singular values seem insensitive, even for large condition numbers. So extra attention seems necessary for the singular vectors.

To our knowledge [34] is the only paper for computing verified bounds for the complete singular value decomposition of $A \in M_{m,n}$ in $\mathcal{O}(Pp^2)$ operations for $P := \max(m,n)$ and $p := \min(m,n)$. Detailed computational results can be found in [34]. The quality of the bounds is often close to maximal accuracy, and even for large clusters still some 8 decimal figures can be guaranteed.

Bounds for the factors of the polar decomposition $A = QP$ with unitary Q and positive semidefinite P follow by $Q = UV^*$ and $P = V\Sigma V^*$ using the singular value decomposition $A = U\Sigma V^*$.

8. Schur decomposition. Let $A = XJX^{-1}$ denote a Jordan decomposition of A, and let $X = QR$ be the QR decomposition of X. Then

$$A = QTQ^*, \quad T := RJR^{-1} \tag{8.1}$$

is a Schur decomposition because J and R are upper triangular. The eigenvalues in T are sorted according to the diagonal of J.

The real Schur decomposition $A = Q'UQ'^T$ for orthogonal Q' and block upper triangular U becomes ill-posed for double eigenvalues. Consider

$$A := \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

with double real eigenvalue 1. An arbitrary small perturbation of A_{12} produces two simple real or a pair of complex eigenvalues, thus changing the block size of the factor U of the real Schur decomposition. Hence we restrict our attention to the complex Schur decomposition $A = QTQ^*$.

However, for the symmetric parameterized matrix in (6.1), which is normal, the Schur decomposition is the eigendecomposition with discontinuous eigenvectors. Hence, certified bounds for the Schur decomposition are restricted to matrices with simple eigenvalues – otherwise facing the ill-posed Jordan decomposition.

For diagonalizable A the algorithms discussed in Section 6 yield inclusions of an eigendecomposition $A = XDX^{-1}$. Combining this with the algorithm in Section 5 for an inclusion of a QR decomposition of X yields inclusions for a Schur decomposition according to (8.1). However, only inclusions $\mathbf{X}$ and $\mathbf{D}$ of X and D are available, so verified error bounds for the QR decomposition of $\mathbf{X}$ are to be computed, which include those of the true X. The factor T is equal to the solution of the linear system $TR = RD$. We have to replace R and D by their computed inclusions, introducing an additional source of overestimation. That is also the reason why only for $\mathrm{cond}(A) \lesssim 10^{14}$ verified inclusions of the Schur factors can be calculated. That does not apply to the other decompositions, including the eigendecomposition needed here.

The Schur decomposition offers the possibility to compute an inclusion of the departure from normality of A. To that end, only the inclusion of $T = RDR^{-1}$ is needed.

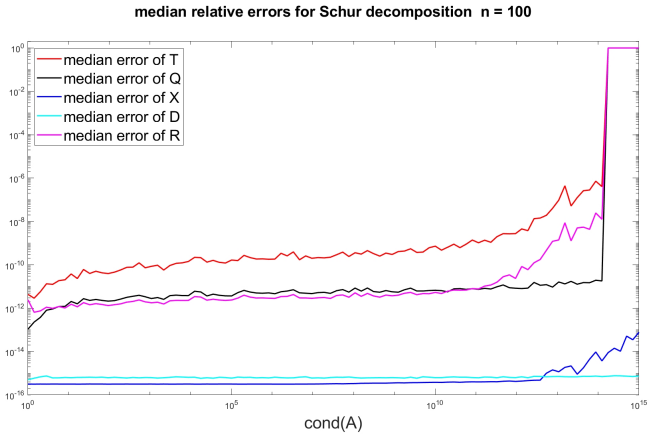

FIG. 11. *Error bounds for the Schur decomposition for different condition numbers*

We show some computational results in Figure 11, where the median relative errors of all components of T is the red line and those of Q the black line. As has been mentioned, the accuracy of the results suffers severely from the fact that only an inclusion $\mathbf{X}$ of X is available.

To see that effect we also show the median relative errors of X and D in blue and cyan, respectively. As can be seen the eigenvalues are enclosed with almost maximal accuracy, the eigenvectors for condition numbers up to 10^{12}, beyond condition number 10^{12} the quality of the eigenvector inclusions decreases slightly. The errors in R, shown in magenta, are close to those of Q for condition numbers up to 10^{12}.

9. Takagi decomposition. We close this note by an application of the inclusion of the factors of a symmetric eigendecomposition. A complex symmetric matrix $A \in M_n(\mathbb{C})$ with $A^T = A$ allows for a Takagi factorization $A = U\Sigma U^T$, also called Autonne-Takagi or symmetric singular value decomposition [14], with unitary U and diagonal Σ with non-negative diagonal elements. The factor Σ comprises of the singular values of A and is unique if the diagonal elements are in nonincreasing order. The factor U may be replaced by US for diagonal S with $S^2 = I$.

Although less known, the Takagi factorization is used in several applications in physics and chemistry, including for example the diagonalization of mass matrices of

Majorana fermions, quadratic fermionic Hamiltonians, the Bloch-Messiah reduction, cf. [6, 39] and the literature cited over there.

It is well known that the factor U is not continuous for singular A. Consider

$$A := \begin{pmatrix} 1 & 0 \\ 0 & e \end{pmatrix}$$

with

$$U := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \Sigma := \begin{pmatrix} 1 & 0 \\ 0 & e \end{pmatrix} \qquad \text{if } e > 0 \ ,$$

and

$$U := \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{-1} \end{pmatrix} \quad \text{and} \quad \Sigma := \begin{pmatrix} 1 & 0 \\ 0 & -e \end{pmatrix} \qquad \text{if } e < 0 \ .$$

The discontinuity is forced by the non-negativity of Σ. Hence, as for the QR decomposition, the computation of the Takagi factorization is ill-posed at $e = 0$. As a consequence, a verification method is only applicable to matrices with full rank.

First, we verify the rule of thumb (2.1) for the sensitivity of the Takagi factors, the results are displayed in Table 10. Here we use `Q = orth(randn(n))` to generate a random complex symmetric matrices of size $n = 100$ with $\mathrm{cond}(A) \approx 10^k$ by

```
D = diag(logspace(0,k,n)); A = Q.'*D*Q; A = A+A.';
```

where the last statement symmetrizes the matrix taking care of rounding errors.

TABLE 10
Sensitivity of the Takagi decomposition for different condition numbers

$\mathrm{cond}(A)$	10^2	10^5	10^8	10^{11}	10^{14}
sensitivity(U)	$2.3 \cdot 10^{-13}$	$2.9 \cdot 10^{-12}$	$9.7 \cdot 10^{-10}$	$5.3 \cdot 10^{-7}$	$3.3 \cdot 10^{-4}$
sensitivity(Σ)	$4.4 \cdot 10^{-15}$	$3.3 \cdot 10^{-15}$	$2.4 \cdot 10^{-15}$	$2.2 \cdot 10^{-15}$	$1.8 \cdot 10^{-15}$

As anticipated, the factor U is sensitive to perturbations of the matrix A, while Σ is not. Of course, the insensitivity of Σ follows by well known perturbation results for singular values.

There are several methods known in the literature to approximate the Takagi factors. For our purposes, the computation of verified bounds, one possibility is the following [9]. Let $A^T = A$ have full rank and denote the singular value decomposition by $A = U\Sigma V^*$. Then $D := U^* A\overline{U}\Sigma^{-1}$ is diagonal, and a computation shows that $UD^{1/2}$ and Σ are the Takagi factors. This is our first method.

One drawback is that the computation of D involves two matrix multiplications. Despite the computational effort this is a source of overestimation because the two factors are interval matrices $\mathbf{U}$ and $\mathbf{V}$ including the true factors U and V, respectively. The inclusions $\mathbf{U}$ and $\mathbf{V}$ may be computed by the methods in the previous section.

Overestimation can be reduced by using $U^* A\overline{U}\Sigma^{-1} = V^*\overline{U}$ which is again diagonal. Now only one multiplication of interval matrices is necessary, and we may expect better results. That is our second method.

We finally transform the problem into a real symmetric eigenproblem, see also [14, 4.4.P2]. Let nonsingular $A^T = A \in M_n(\mathbb{C})$ be given, and denote $A = E + iF$

with $E^T = E, F^T = F$ and $E, F \in M_n(\mathbb{R})$. A direct computation shows that the eigenvalues of the symmetric matrix

$$M := \begin{pmatrix} E & F \\ F & -E \end{pmatrix}$$

come in $\pm$ pairs. If $\begin{pmatrix} x \\ y \end{pmatrix}$ is an eigenvector to $\lambda \in \mathbb{R}$, then $\begin{pmatrix} y \\ -x \end{pmatrix}$ is an eigenvector to $-\lambda$. After suitable renumbering the eigendecomposition of M is

$$M \begin{pmatrix} X & Y \\ Y & -X \end{pmatrix} = \begin{pmatrix} X & Y \\ Y & -X \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix} .$$

A direct computation verifies that $U := X + iY$ and $V := X - iY$ are unitary and $AV = U\Sigma$. Hence the diagonal elements of Σ are the singular values of A. Finally

$$A\overline{U} = (E + iF)(X - iY) = AV = U\Sigma$$

verifies that U and Σ are the factors of the Takagi decomposition. Inclusions of X and Y are computed with the methods for symmetric eigendecomposition presented in Section 6.

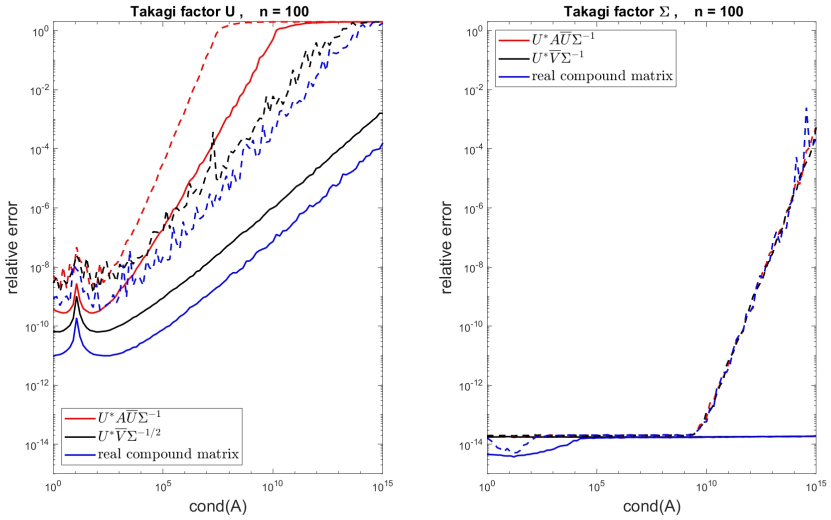

FIG. 12. *The Takagi decomposition, solid line median, dashed line maximum relative error*

We show some computational results in Figure 12. The median relative errors of U in the left and Σ in the right graph computed by the three methods are the solid lines in red, black, and blue, respectively. As for the inclusions of U the third method seems best. In any case, as expected by our rule of thumb (2.1), the relative error increases with the condition number. The reason for the small peak at $\mathrm{cond}(A) \approx 10$ is not clear to us, it may be due to the construction of the test matrices. For Σ all three methods compute bounds of almost maximal accuracy.

However, the numbers are slightly misleading because, for example, most elements of U are enclosed with high accuracy, and only few corresponding to the smallest singular values are weaker. Thus the median reflects mostly the relative error of the better inclusions. Therefore, we display for both U and Σ the maximum relative errors as well, the dashed lines. Figure 12 shows that for the first method and condition number beyond about 10^9 some inclusions have relative error close to 1, for the second method beyond 10^{14}, where for the third method even for $\mathrm{cond}(A) \lesssim 10^{15}$ the inclusions seem to contain some information.

For the singular values Σ below condition number 10^9 all bounds are of maximal accuracy, with some deterioration for larger condition numbers. The results of all three methods are of similar quality.

Acknowledgment. The authors wish to express their heartful thanks to the two anonymous referees for their thorough reading and most constructive and helpful comments.

REFERENCES

[1] IEEE Standard for Floating-point Arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pages 1–84, 2019.
[2] Advanpix: Multiprecision Computing Toolbox for MATLAB, 2024. Code and documentation available at `http://www.advanpix.com/`.
[3] P. Ahrens, J. Demmel, H. D. Nguyen: Algorithms for efficient reproducible floating-point summation. *ACM Trans. Math. Software*, 46: 1–49, 2020.
[4] G. Alefeld, H. Spreuer: Iterative improvement of componentwise error bounds for invariant subspaces belonging to a double or nearly double eigenvalue. *Computing*, 36: 321–334, 1986.
[5] D. H. Bailey: A Fortran-90 double-double precision library. `https://www.davidhbailey.com/dhbsoftware/`.
[6] G. Cariolaro and G. Pierobon. Bloch-messiah reduction of gaussian unitaries by Takagi factorization. *Phys. Rev. A*, 94: Article 062109, 2016.
[7] T. J. Dekker: A floating-point technique for extending the available precision. *Numer. Math.*, 18: 224–242, 1971.
[8] L. Dieci, T. Eirola: On smooth decompositions of matrices. *SIAM J. Matrix Anal. Appl.*, 20(3): 800–819, 1999.
[9] L. Dieci, A. Papini, A. Pugliese: Takagi factorization of matrices depending on parameters and locating degeneracies of singular values. *SIAM J. Matrix Anal. Appl.*, 43(3): 1148–1161, 2022.
[10] F. R. Gantmacher: *The Theory of Matrices*, volume 1, Chelsea, New York, 1959.
[11] N. J. Higham: *Accuracy and Stability of Numerical Algorithms*, SIAM Publications, Philadelphia, 2nd edition, 2002.
[12] N. J. Higham: private communication, 2023
[13] F. R. de Hoog, R. S. Anderssen, M. A. Lukas: Differentiation of matrix functionals using triangular factorization. *Math. Comp.*, 80(275): 1585–1600, 2011.
[14] R. A. Horn, C. R. Johnson: *Matrix Analysis*, second edition. Cambridge University Press, 2013.
[15] R. B. Kearfott, M. T. Nakao, A. Neumaier, S. M. Rump, S. P. Shary, P. van Hentenryck: Standardized notation in interval analysis, *Computational Technologies*, 15(1): 7–13, 2010.
[16] D. E. Knuth: *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison Wesley, Reading, Massachusetts, 1969.
[17] R. Kobayashi, M. Lange, A. Minamihata, S. M. Rump: Verified inclusion of a basis of the null space. *Reliable Computing*, 27: 26–41, 2020.
[18] R. Krawczyk: Fehlerabschätzung reeller Eigenwerte und Eigenvektoren von Matrizen. *Computing*, 4: 281–293, 1969.
[19] M. Lange, S. M. Rump: Faithfully rounded floating-point computations. *ACM Trans. Math. Softw.*, 46(3): Article 21, 2020.
[20] J. de Leeuw: Differentiating the QR decomposition, Preprint, 2023 (DOI: 10.13140/RG.2.2.19201.12640).

[21] M. Malcolm: On accurate floating-point summation. *Comm. ACM*, 14(11): 731–736, 1971.
[22] MATLAB. User's Guide, Version 2023b, the MathWorks Inc., 2023.
[23] O. Møller: Quasi double precision in floating-point arithmetic. *BIT Numerical Mathematics*, 5: 37–50, 1965.
[24] S. Miyajima: Fast enclosure for all eigenvalues and invariant subspaces in generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 35(3): 1205–1225, 2014.
[25] S. Miyajima, T. Ogita, S. M. Rump, S. Oishi: Fast verification of all eigenpairs in symmetric positive definite generalized eigenvalue problem. *Reliable Computing*, 14: 24–45, 2010.
[26] A. Neumaier: *Interval methods for systems of equations*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1990.
[27] T. Ogita, S. M. Rump, S. Oishi: Accurate sum and dot product. *SIAM J. Sci. Comput.*, 26(6): 1955–1988, 2005.
[28] S. Oishi, K. Ichihara, M. Kashiwagi, T. Kimura, X. Liu, H. Masai, Y. Morikura, T. Ogita, K. Ozaki, S. M. Rump, K. Sekine, A. Takayasu, N. Yamanaka: *Principle of Verified Numerical Computations*. Corona Publisher, Tokyo, Japan, 2018. [in Japanese].
[29] K. Ozaki, T. Ogita, S. Oishi: Error-free transformation of matrix multiplication with a posteriori validation. *Numer. Linear Alg. Appl.*, 23(5): 931–946, 2016.
[30] F. Rellich: Störungstheorie der Spektralzerlegung. *Math. Ann.*, 113: 79–91, 1994.
[31] S. M. Rump: INTLAB – INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Springer Netherlands, Dordrecht, 1999.
[32] S. M. Rump: Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19: 287–449, 2010.
[33] S. M. Rump: Verified error bounds for all eigenvalues and eigenvectors of a matrix. *SIAM J. Matrix Anal. Appl.*, 43(4): 1736–1754, 2022.
[34] S. M. Rump, M. Lange: Fast computation of error bounds for all eigenpairs of a Hermitian and all singular pairs of a rectangular matrix with emphasis on eigen- and singular value clusters. *J. Comput. Appl. Math.*, 434: Article 115332, 2023.
[35] S. M. Rump, T. Ogita: On a quality measure for interval inclusions. *BIT Numerical Mathematics*, 64: Article 22, 2024.
[36] S. M. Rump, T. Ogita, S. Oishi: Accurate floating-point summation part I: Faithful rounding. *SIAM J. Sci. Comput.*, 31(1): 189–224, 2008.
[37] G. W. Stewart: On the perturbation of LU, Cholesky, and QR factorizations. *SIAM J. Matrix Anal. Appl.*, 14: 1141–1146, 1993.
[38] J. Sun: Multiple eigenvalue sensitivity analysis. *Linear Alg. Appl.*, 137/138: 183–211, 1990.
[39] A. E. Teretenkov. Singular value decomposition for skew-Takagi factorization with quantum applications. *Linear and Multilinear Algebra*, 70(22): 7762–7769, 2022.
[40] A. N. Tikhonov: Regularization of incorrectly posed problems. *Dokl. Akad. Nauk SSSR*, 153(1): 49–52, 1963.
[41] A. N. Tikhonov, V. L. Arsenin: *Solutions of Ill-posed Problems*. John Wiley & Sons, New York, 1977.
[42] J. M. Wolfe: Reducing truncation errors by programming. *Comm. ACM*, 7(6): 355–356, 1964.
[43] G. Zielke, V. Drygalla: Genaue Lösung linearer Gleichungssysteme. *GAMM Mitt. Ges. Angew. Math. Mech.*, 26: 7–108, 2003.