NOLTA, IEICE

Paper

Accurate and robust inverse Cholesky factorization

Takeshi Ogita [1,3 a)] *and Shin'ichi Oishi* [2,3]

[1] *Division of Mathematical Sciences, School of Arts and Sciences, Tokyo Woman's Christian University, Tokyo 167-8585, Japan*

[2] *Department of Applied Mathematics, Faculty of Science and Engineering, Waseda University, Tokyo 169-8555, Japan*

[3] *JST, CREST*

[a)] *ogita@lab.twcu.ac.jp*

Received May 30, 2010; Revised October 17, 2010; Published January 1, 2012

Abstract: In this paper, an algorithm for an accurate matrix factorization based on Cholesky factorization for extremely ill-conditioned matrices is proposed. The Cholesky factorization is widely used for solving a system of linear equations whose coefficient matrix is symmetric and positive definite. However, it sometimes breaks down by the presence of an imaginary root due to the accumulation of rounding errors, even if the matrix is actually positive definite. To overcome this, a completely stable algorithm named inverse Cholesky factorization is investigated, which never breaks down as long as the matrix is symmetric and positive definite. The proposed algorithm consists of standard numerical algorithms and an accurate algorithm for dot products. Moreover, it is shown that the algorithm can also verify the positive definiteness of a given real symmetric matrix. Numerical results are presented for illustrating the performance of the proposed algorithms.

Key Words: Cholesky factorization, ill-conditioned problem, accurate numerical algorithm, floating-point arithmetic, positive definiteness

1. Introduction

Matrix factorizations such as LU, QR and Cholesky factorizations are frequently discussed in numerical linear algebra since they are used as building blocks of scientific computing. Following the previous work [11] by the first author, we propose an algorithm for accurately computing an inverse Cholesky factorization of a real symmetric and positive definite $n \times n$ matrix A, especially for A being extremely ill-conditioned. Namely, the condition number of A is beyond the limit of working precision of floating-point arithmetic.

Let $\mathbb{R}$ be the set of real numbers. Let $\mathbb{F}$ be a set of floating-point numbers conforming IEEE 754 standard [2, 3]. Let $\mathbf{u}$ denote the unit roundoff of floating-point arithmetic. In double precision (binary64) arithmetic, $\mathbf{u} = 2^{-53}$. Let $\kappa(A) := \|A\| \cdot \|A^{-1}\|$ as the condition number of $A \in \mathbb{R}^{n \times n}$, where $\|\cdot\|$ stands for spectral norm for matrices and Euclidean norm for vectors throughout the paper. In general, $\kappa(A)$ can be used to measure how singular the matrix A is. For example, consider solving

Nonlinear Theory and Its Applications, IEICE, vol. 3, no. 1, pp. 103–111 DOI: 10.1587/nolta.3.103

a linear system $Ax = b$ with $A \in \mathbb{R}^{n\times n}$ and a nonzero vector $b \in \mathbb{R}^n$. Let us also consider a perturbed linear system $(A + \Delta A)y = b$ where $\Delta A \in \mathbb{R}^{n\times n}$ with $\|\Delta A\| \le \varepsilon\|A\|$ for some small constant ε. Then the following holds (cf. e.g. [5, 7]): If $\mu := \varepsilon \cdot \kappa(A) < 1$, then $A + \Delta A$ is nonsingular and

$$\frac{\|x^* - y^*\|}{\|x^*\|} \le \frac{\varepsilon}{1-\mu}\kappa(A), \tag{1}$$

where $x^* := A^{-1}b$ and $y^* := (A + \Delta A)^{-1}b$.

Suppose $A \in \mathbb{F}^{n\times n}$ is symmetric and positive definite. When using Cholesky factorization to obtain a computed solution $\widetilde{x}$ of $Ax = b$ by floating-point arithmetic, it holds that

$$(A + \Delta A)\widetilde{x} = b \quad \text{with } \|\Delta A\| \le c_n\mathbf{u}\|A\|,\ c_n = \mathcal{O}(n^2)$$

from backward error analysis [4, 7, 23]. So, if $\mu_1 := c_n\mathbf{u} \cdot \kappa(A) < 1$, then (1) yields

$$\frac{\|x^* - \widetilde{x}\|}{\|x^*\|} \le \frac{c_n\mathbf{u}}{1-\mu_1}\kappa(A) \approx c_n\mathbf{u} \cdot \kappa(A).$$

Therefore, if $\kappa(A) \gtrsim \mathbf{u}^{-1}$, then $\mu_1 \gtrsim 1$ and few or no correct digit can be expected in the computed solution $\widetilde{x}$ when solving a linear system $Ax = b$ in working precision.

In this paper, we consider to treat the case where

$$\kappa(A) \gg \mathbf{u}^{-1}.$$

As mentioned above, it is not possible in general to treat this case by standard numerical algorithms. In our previous work [11], we have presented algorithms for accurately calculating inverse LU and inverse QR factorizations in such cases. In those algorithms we rely on the fact that standard numerical algorithms for LU and QR using pure floating-point arithmetic rarely break down due to the rounding error, which works as some kind of the regularization. However, when a floating-point Cholesky factorization for an ill-conditioned matrix A is executed, it sometimes breaks down by the presence of an imaginary root due to the accumulation of rounding errors, even if A is actually positive definite. Namely, we cannot directly apply our framework proposed in [11] to the Cholesky factorization.

To overcome it, we develop a robust algorithm for calculating a good approximate inverse X of the Cholesky factor $\widehat{R}$ such that $A = \widehat{R}^T\widehat{R}$ satisfying

$$A^{-1} \approx XX^T.$$

The proposed algorithm is completely stable in the sense of numerical computations, i.e., barring the presence of underflow or overflow, it never fails as long as the matrix is symmetric and positive definite. In other words, if the algorithm breaks down, then the matrix A is proved to be not positive semidefinite, i.e. has at least one negative eigenvalue.

With the same spirit as the previous work [11] and Rump's method for inverting extremely ill-conditioned matrices [14, 16, 19], we emphasize that pure floating-point arithmetic and standard numerical algorithms are utilized as much as possible. The only one exception is that we need an algorithm for accurately computing dot products, more precisely, as if computed in k-fold working precision and rounded into ℓ pieces of working precision floating-point numbers for any $k \ge 2$ and $1 \le \ell \le k$. For example, such accurate dot product algorithms have been developed in [12, 21, 22] for the purpose, and they are very fast.

The rest of the paper is organized as follows: In the following section, we state notation and definitions used in this paper. In Section 3, we present a concrete algorithm for an accurate inverse Cholesky factorization. Numerical results are presented in Section 4 for illustrating the performance of our proposed algorithm. Finally, we conclude the paper in Section 5.

2. Notation and definitions

For $A = (a_{ij}), B = (b_{ij}) \in \mathbb{R}^{m\times n}$, we denote by $|A| = (|a_{ij}|) \in \mathbb{R}^{m\times n}$ a nonnegative matrix consisting of entrywise absolute values, and an inequality $A \le B$ is understood entrywise, i.e., $a_{ij} \le b_{ij}$ for

all (i, j). The notation $A \geq O$ (or $A > O$) means that all elements of A are nonnegative (positive). Similar notation applies to real vectors. The trace of $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ is defined by

$$\mathrm{tr}(A) := \sum_{i=1}^{n} a_{ii}.$$

Moreover, $\lambda_i(A)$ denotes the i-th eigenvalue of A in increasing order.

To construct a completely stable algorithm for an inverse Cholesky factorization, a little use of interval arithmetic (See, e.g. [1, 8, 10]) is required.

Let $\langle a, r\rangle$ denote an interval of the midpoint-radius representation such that

$$\langle a, r\rangle := \{x \in \mathbb{R} : |x - a| \leq r\}$$

with a midpoint $a \in \mathbb{R}$ and a radius $r \geq 0$. Let $\langle b, s\rangle$ denote an interval in a similar way. Then the basic operations (addition, subtraction and multiplication) are defined as follows:

$$\begin{aligned}\langle a, r\rangle + \langle b, s\rangle &:= \langle a + b, r + s\rangle\\ \langle a, r\rangle - \langle b, s\rangle &:= \langle a - b, r + s\rangle\\ \langle a, r\rangle \cdot \langle b, s\rangle &:= \langle ab, |a|s + r|b| + rs\rangle\end{aligned}$$

The above operations can apply to interval matrices as well.

For later use, we define the magnitude of an interval matrix $\langle C, R\rangle$ with $C, R \in \mathbb{R}^{m\times n}$ by

$$\mathrm{mag}\langle C, R\rangle := |C| + R.$$

For readability we denote by $\varphi(\gamma)$ any constant such as $\varphi(\gamma) = c \cdot \gamma$ for some $c > 0$. Then c may depend on the size n of the problem.

Let $x, y \in \mathbb{F}^n$. We assume that an accurate computation of a dot product $x^T y$ is available as the tool of obtaining $s_\ell := \sum_{i=1}^{\ell} s^{(i)}$ with $s^{(i)} \in \mathbb{F}$ such that

$$\left|\frac{x^T y - s_\ell}{x^T y}\right| \leq \varphi(\mathbf{u}^\ell) + \varphi(\mathbf{u}^k)\mathrm{cond}(x^T y),\ k \geq 2,\ 1 \leq \ell \leq k$$

for $x^T y \neq 0$. Here $\mathrm{cond}(x^T y)$ is the condition number of dot product [12] defined by

$$\mathrm{cond}(x^T y) := 2\frac{|x^T||y|}{|x^T y|}, \quad x^T y \neq 0.$$

This means we can calculate $x^T y$ as if computed in k-fold working precision and rounded into ℓ pieces of working precision floating-point numbers s_i, $1 \leq i \leq \ell$. Fortunately, we already have such accurate dot product algorithms proposed in [12, 19, 21, 22] at hand.

Throughout the paper, we use the notation

$$C_\ell = \{A \cdot B\}_k^\ell$$

which satisfies

$$|AB - C_\ell| \leq \varphi(\mathbf{u}^\ell)|AB| + \varphi(\mathbf{u}^k)|A||B| \tag{2}$$

for $A \in \mathbb{R}^{m\times p}$ and $B \in \mathbb{R}^{p\times n}$. In the case of $\ell = 1$, we abbreviate it as $C = \{A \cdot B\}_k$.

3. Accurate inverse Cholesky factorization

In this section, we present an algorithm for computing an accurate inverse Cholesky factorization.

We have discussed about the accuracy of LU and QR factorizations in [11], respectively. Similarly, we shall consider how to define the accuracy of a Cholesky factorization.

Let $A = A^T \in \mathbb{F}^{n\times n}$ with $a_{ii} > 0$ for $1 \leq i \leq n$. Suppose a standard numerical Cholesky factorization of A runs to completion. Here "run to completion" means that no imaginary root appears in the factorization process. Throughout the paper, the Matlab-style notation

$$R = \mathtt{chol}(A)$$

means a floating-point Cholesky factorization of A using a standard numerical algorithm (e.g., `xPOTRF` in LAPACK) such that

$$A \approx R^T R,$$

where R is an upper triangular matrix. Then it is known [7] that the computed Cholesky factor R always satisfies

$$\frac{\|A - R^T R\|}{\|A\|} \leq \varphi(\mathbf{u})$$

if the floating-point Cholesky factorization runs to completion. Thus it is similar to the case of the LU factorization that not so much information on the accuracy of the Cholesky factorization can be obtained from the residual norm $\|A - R^T R\|$ or its relative norm. This means that we again need an another criterion.

Suppose the exact Cholesky factorization of A runs to completion such that $A = \widehat{R}^T \widehat{R}$. Then it holds that

$$\kappa(\widehat{R}) = \kappa(A)^{\frac{1}{2}}.$$

On the other hand, heuristics tells us that a computed factor R satisfies

$$\kappa(R) \approx \sqrt{\min\{\kappa(A), \varphi(\mathbf{u}^{-1})\}} \tag{3}$$

for any positive definite matrix A as long as the numerical Cholesky factorization of A runs to completion. This can be interpreted as follows: By the backward error analysis of the floating-point Cholesky factorization [4, 23], we have

$$A + \Delta = R^T R, \quad \|\Delta\| \lesssim 12n^2 \mathbf{u} \|A\|. \tag{4}$$

The 'perturbation' Δ due to rounding errors in floating-point arithmetic work as some kind of regularization for A, which has been observed in [11, 14, 19]. If $\kappa(A) \gg \mathbf{u}^{-1}$, then $\kappa(A + \Delta)$ is down to $\mathcal{O}(\mathbf{u}^{-1})$ in almost all cases. Otherwise, $\kappa(A + \Delta) \approx \kappa(A)$. Thus (3) follows.

However, as mentioned before, the floating-point Cholesky factorization of an ill-conditioned matrix A sometimes breaks down due to an accumulation of the rounding errors. To avoid the break-down, a diagonal shift can apply to A such as $A + \delta I$ for some suitable $\delta > 0$. Let $\lambda_{\max}$ and $\lambda_{\min}$ be the largest and the smallest eigenvalues of A, respectively. Then

$$\kappa(A + \delta I) = \frac{\lambda_{\max} + \delta}{\lambda_{\min} + \delta}.$$

Let $\alpha := \delta / \lambda_{\max}$. In the case of $0 < \lambda_{\min} \ll \delta$, it holds that

$$\kappa(A + \delta I) \approx \frac{\lambda_{\max}}{\delta} = \alpha^{-1}. \tag{5}$$

On the other hand, in the case of $\delta \ll \lambda_{\min}$, it holds that

$$\kappa(A + \delta I) \approx \frac{\lambda_{\max}}{\lambda_{\min}} = \kappa(A). \tag{6}$$

Let $\widetilde{R}$ be a computed Cholesky factor of $A + \delta I$, i.e.,

$$\widetilde{R} = \texttt{chol}(A + \delta I).$$

Combining (5) and (6) yields

$$\kappa(\widetilde{R}) \approx \sqrt{\min\{\kappa(A), \alpha^{-1}\}}. \tag{7}$$

The problem is how to choose a suitable constant α. In the accurate inverse LU factorization in [11], it holds that

$$\kappa(AU^{-1}) \approx \varphi(\mathbf{u}) \cdot \kappa(A),$$

where U is an upper triangular matrix as a computed LU factor of A. Similarly, the constant α in (7) becomes a drop factor for the condition number of A such that

$$\kappa(\widetilde{R}^{-T}A\widetilde{R}^{-1}) \approx \alpha \cdot \kappa(A).$$

So, it is desirable to choose α as small as possible for dropping $\kappa(A)$ efficiently. However, if α (and therefore δ) is too small, then $\texttt{chol}(A+\delta I)$ may break down due to the accumulation of the rounding errors. Namely, we need to obtain an optimal δ before executing $\texttt{chol}(A+\delta I)$. Fortunately, it is possible; In [20], Rump has modified (4) for verification of positive definiteness:

$$A + \Delta = R^T R, \quad \|\Delta\| \le c'_n \mathbf{u} \cdot \mathrm{tr}(A) =: \delta, \tag{8}$$

where c'_n is some computable constant with $c'_n \approx n$ (see [20]). Let $\widetilde{A} = (\widetilde{a}_{ij})$ be a floating-point $n \times n$ matrix satisfying

$$\begin{cases} \widetilde{a}_{ii} \ge a_{ii} + \delta \\ \widetilde{a}_{ij} = a_{ij} \quad \text{for } i \ne j \end{cases}.$$

Such $\widetilde{A}$ can easily be obtained by directed rounding (rounding-upwards in this case) defined in IEEE standard 754. See [20] for details.

As long as A is positive definite, $\texttt{chol}(\widetilde{A})$ never breaks down (see [20, Corollary 2.9]), even if taking the rounding errors into account, and

$$\widetilde{A} + \widetilde{\Delta} = \widetilde{R}^T \widetilde{R}, \quad \|\widetilde{\Delta}\| \le \delta.$$

In other words, if $\texttt{chol}(\widetilde{A})$ fails, then A is proved to be not positive semidefinite.

The idea of the proposed algorithm is basically as follows: First, we compute an approximate Cholesky factor R_1 of $A + \delta_1$ such that $A + \delta_1 I \approx R_1^T R_1$ in working precision. Next, we compute an approximate inverse of $X_1 \approx R_1^{-1}$ in working precision. Then we can expect

$$\kappa(X_1^T A X_1) = \varphi(\alpha)\kappa(A).$$

After that, $X_1^T A X_1$ is computed in *higher* precision with its error bound and rounded into an interval matrix $\langle G_2, E_2 \rangle$. Iteratively we compute an approximate Cholesky factor R_2 of $G_2 + \delta_2 I$ in working precision, and then compute an approximate inverse of $T_2 \approx R_2^{-1}$ in working precision. After that, $X_1 \cdot T_2$ is computed in *higher* precision and stored into a matrix X_2 in *higher* precision. Then we can expect

$$\kappa(X_2^T A X_2) = \varphi(\alpha^2)\kappa(A).$$

After that, $X_2^T A X_2$ is computed in *higher* precision with its error bound and rounded into an interval matrix $\langle G_3, E_3 \rangle$, and so forth. In general, we aim to develop an algorithm satisfying

$$\kappa(X_k^T A X_k) = \max\{\varphi(\alpha^k)\kappa(A), 1\}.$$

for $k = 1, 2, \ldots$.

Suppose interval matrices $\langle G_k, E_k \rangle$ can be obtained such that $X_k^T A X_k \in \langle G_k, E_k \rangle$ for all k. By Sylvester's law of inertia, if A is positive definite, then $X_k^T A X_k$ are also positive definite with non-singular X_k for all k. However, G_k can be not positive semidefinite for some k due to the rounding errors. Taking the treatment of interval matrices $\langle G_k, E_k \rangle$ into account, we need to slightly modify the diagonal shift δ. By Weyl's theorem, it holds that

$$|\lambda_i(X_k^T A X_k) - \lambda_i(G_k)| \le \|E_k\| \quad \text{for all } i \tag{9}$$

since $X_k^T A X_k \in \langle G_k, E_k \rangle$. To ensure the positive definiteness of $G_k + \delta_k I$ with taking care of the rounding errors, we set δ_k as

$$\delta_k := c'_n \mathbf{u} \cdot \mathrm{tr}(G_k) + \|E_k\|. \tag{10}$$

Suppose $\|E_k\| \approx \mathbf{u}\|G_k\|$. Then

$$\delta_k \approx n\mathbf{u}\|G_k\| + \mathbf{u}\|G_k\| \approx n\mathbf{u}\|G_k\|.$$

If that is the case, then we have $\alpha \approx n\mathbf{u}$ and

$$\kappa(X_k^T A X_k) = \max\{\varphi((n\mathbf{u})^k)\kappa(A), 1\}.$$

To achieve $\|E_k\| \approx \mathbf{u}\|G_k\|$, high precision computation for dot product mentioned in Section 2 is necessary. If $\|E_k\| \gg \mathbf{u}\|G_k\|$ for some k, then δ_k may become too large to drop $\kappa(A)$ as mentioned before.

Our algorithm for an accurate inverse Cholesky factorization is based on iterative refinement of an approximate inverse X of the exact Cholesky factor $\widehat{R}$ of A:

Algorithm 1 (Robust inverse Cholesky factorization)
For a symmetric matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ with $a_{ii} > 0$ for all i and a specified tolerance $\varepsilon_{\rm tol} \leq 1$, this algorithm calculates an upper triangular matrix $X = \sum_{\nu=1}^{m} X^{(\nu)}$, $X^{(\nu)} \in \mathbb{F}^{n\times n}$ such that $\|X^T AX - I\| \leq \varepsilon_{\rm tol}$ if A is positive definite.

0: *Put* $X_0 := \mathrm{diag}(2^{-\lceil \frac{1}{2}\log_2(a_{ii})\rceil})$ *for diagonal scaling.*
for $k = 0, 1, 2, \ldots$
1: $p := \lceil -\log(\|A\|\|X_k\|^2)/\log \mathbf{u}\rceil + 2$.
2: $\langle B_k, W_k\rangle := \{A \cdot X_k\}_p^p$.
3: $\langle C_k, V_k\rangle := \{X_k^T \cdot B_k\}_p + \langle O, |X_k^T|W_k\rangle$.
4: $\langle G_k, E_k\rangle := \frac{1}{2}(\langle C_k, V_k\rangle + \langle C_k^T, V_k^T\rangle)$.
5: *If* $\|\mathrm{mag}\langle G_k - I, E_k\rangle\| < \varepsilon_{\rm tol}$, *then* $X := X_k$ *and stop.*
6: *Compute* $S_k \in \mathbb{F}^{n\times n}$ *s.t.* $(S_k)_{ii} \geq (G_k)_{ii} + \delta_k$ *for* $1 \leq i \leq n$ *and* $(S_k)_{ij} = (G_k)_{ij}$ *for* $i \neq j$
where $\delta_k = c'_n\mathbf{u} \cdot \mathrm{tr}(G_k) + \|E_k\|$.
7: *Check whether* $(S_k)_{ii} > 0$ *for all* i. *If not, then stop.*
8: *Floating-point Cholesky factorization* $R_k = \mathtt{chol}(S_k)$ *s.t.* $S_k \approx R_k^T R_k$. *If it failed, then stop.*
9: *Compute* $T_k \approx R_k^{-1}$ *by solving a triangular matrix equation* $R_k T = I$ *for* T.
10: $m := \lceil -\log(\|X_k\|\|T_k\|)/\log \mathbf{u}\rceil + 1$.
11: $X_{k+1} := \{X_k \cdot T_k\}_m^m$.
end

Remark 1
To avoid bad scaling of A, we first apply diagonal scaling as $X_0 AX_0$ with suitable powers of 2 as diagonal elements of the diagonal matrix X_0. It is based on Bauer's and van der Sluis's results for the optimal scaling. See [4, 20, 23] for details. Note that no rounding error occurs in the computation of $X_0 AX_0$ baring the presence of underflow. Therefore, at the first iteration ($k = 0$), steps from 1 to 4 can be skipped, i.e. just set $G_0 := X_0 AX_0$ and $E_0 := O$.

Remark 2
The stopping criterion $\|\mathrm{mag}\langle G_k - I, E_k\rangle\| < \varepsilon_{\rm tol}$ at step 5 basically means $\|X_k^T AX_k - I\| < \varepsilon_{\rm tol}$, which is a standard way to check whether X_k is a good preconditioner for A.

Note that high precision computations (of dot product) are necessary only at steps 2, 3 and 11. Among them, the results of the high precision computations are stored in high precision at steps 2 and 11. From step 2 to 4 at the k-th iteration, an interval matrix $\langle G_k, E_k\rangle$ as an inclusion of $X_k^T AX_k$ is computed satisfying

$$|X_k^T AX_k - G_k| \leq \varphi(\mathbf{u})|X_k^T AX_k| + \varphi(\mathbf{u}^p)|X_k^T||A||X_k| = E_k. \tag{11}$$

Step 1 determines how much precision is used for calculating $X_k^T AX_k$ in steps 2 and 3 by adapting to $|X_k^T||A||X_k|$ in (11). Moreover, step 4 just ensures that G_k are symmetric. At step 6, it is easy to obtain such S_k via rounding mode controlled computation, especially in rounding upwards (see, e.g. [13, 17]). In step 11 at k-th iteration, an inverse Cholesky factor X_k is updated to X_{k+1} satisfying

Table I. Results for scaled Hilbert matrix with $n = 21$ and $\kappa(A) \approx 8.16 \cdot 10^{29}$ by the proposed algorithm.

k	$\kappa(X_k^T A X_k)$	$(c'_n\mathbf{u})^k \kappa(A)$	$\kappa(X_k)$	$(c'_n\mathbf{u})^{-\frac{k}{2}}$
0	$3.14 \cdot 10^{29}$	$8.16 \cdot 10^{29}$	4	1
1	$9.60 \cdot 10^{14}$	$4.57 \cdot 10^{16}$	$2.92 \cdot 10^{7}$	$4.23 \cdot 10^{6}$
2	$3.49 \cdot 10^{1}$	$2.56 \cdot 10^{3}$	$1.53 \cdot 10^{14}$	$1.79 \cdot 10^{13}$
3	1.00	< 1	$9.03 \cdot 10^{14}$	$> \sqrt{\kappa(A)}$

$$|X_k T_k - X_{k+1}| \leq \varphi(\mathbf{u}^m)|X_k T_k| + \varphi(\mathbf{u}^m)|X_k||T_k|. \tag{12}$$

Step 10 also determines how much precision is used (and required) for calculating $X_k T_k$ and storing the result by adapting to $|X_k||T_k|$ in (12).

The computational complexity of Algorithm 1 in terms of time and space complexity strongly depends on that of accurate matrix multiplication. We assume that an algorithm for accurate dot product based on the algorithms in [12, 21, 22] is used. Let $x, y \in \mathbb{R}^n$. Then computing $\{x^T y\}_\alpha$ or $\{x^T y\}_\alpha^\alpha$ requires $\mathcal{O}(\alpha^2 n)$ flops. In practice, $p \approx k + 2$ and $m \approx \lceil k/2 \rceil + 1$ in Algorithm 1. Thus, it requires $\mathcal{O}(k^2 n^3)$ flops to compute the matrix multiplication $\{A \cdot X_k\}_p^p$ in step 2, $\{X_k^T \cdot B_k\}_p$ in step 3 and $\{X_k \cdot T_k\}_m^m$ in step 11. The rest of the algorithm requires $\mathcal{O}(n^3)$ flops. If the algorithm stops at $k = K$, then the total time complexity of Algorithm 1 becomes $\mathcal{O}(K^3 n^3)$ flops since $\sum_{k=1}^{K} k^2 n^3 = \mathcal{O}(K^3 n^3)$. Of course, K depends on the condition number of A.

If the algorithm stops in the absence of overflow, then we can determine whether A is positive definite or not:

Theorem 1
Let A be a real symmetric $n \times n$ matrix. Then the following is true:

(i) If Algorithm 1 stops at step 5 for any $\varepsilon_{\mathrm{tol}} \leq 1$, then A is positive definite, also in the presence of underflow.

(ii) If Algorithm 1 stops at step 7, then A is not positive semidefinite, also in the presence of underflow.

(iii) If Algorithm 1 stops at step 8, then A is not positive semidefinite if no overflow occurs.

Proof. Recall that $X_k^T A X_k \in \langle G_k, E_k \rangle$. Since $X_k^T A X_k - I \in \langle G_k - I, E_k \rangle$, it holds that $\|X_k^T A X_k - I\| \leq \|\mathrm{mag}\langle G_k - I, E_k \rangle\| < \varepsilon_{\mathrm{tol}}$ if Algorithm 1 stops at step 5. Then $\lambda_i(X_k^T A X_k) > 1 - \varepsilon_{\mathrm{tol}} \geq 0$ for all i, and (i) follows. If Algorithm 1 stops at step 7, then $0 > (G_k)_{ii} + \delta_k \geq (G_k)_{ii} + \|E_k\| \geq (X_k^T A X_k)_{ii}$ for some i, and (ii) follows. Finally, suppose Algorithm 1 stops at step 8. Then there are two possibilities: Overflow occurs or an imaginary root appears in $\texttt{chol}(S_k)$. In the former case, we cannot determine the positive definiteness of A. In the latter case, Corollary 2.9 in [20] implies $X_k^T A X_k$ is not positive semidefinite, and (iii) follows. □

Thus, if we merely focus on verification of positive definiteness of A, then setting $\varepsilon_{\mathrm{tol}} = 1$ is the most efficient in terms of computational cost.

4. Numerical results

We present some numerical results showing the behavior of our proposed algorithm (Algorithm 1) of an inverse Cholesky factorization. All computations are done on Matlab 2009a with IEEE 754 double precision (binary64) arithmetic as working precision ($\mathbf{u} = 2^{-53} \approx 1.1 \cdot 10^{-16}$). We also use INTLAB [18] toolbox on Matlab for implementing the proposed algorithm. As a stopping criterion for Algorithm 1, we set $\varepsilon_{\mathrm{tol}} = 10^{-6}$.

First, scaled Hilbert matrix H_n is treated. Here H_n is an integer (symmetric and positive definite) matrix whose elements are exactly representable in double precision (binary64) floating-point numbers for $n \leq 21$. For $n = 21$, $\kappa(H_{21}) \approx 8.16 \cdot 10^{29}$. We put $A := H_{21}$. The result is displayed in Table I.

Next, a slightly modified version of Rump matrix [15] is treated, which is based on the function `randmat(n,cnd)` in INTLAB, and surely generates symmetric and positive definite matrices. We set

Table II. Results for Rump matrix with $n = 500$ and $\kappa(A) \approx 4.76 \cdot 10^{53}$ by the proposed algorithm.

k	$\kappa(X_k^T A X_k)$	$(c'_n\mathbf{u})^k \kappa(A)$	$\kappa(X_k)$	$(c'_n\mathbf{u})^{-\frac{k}{2}}$
0	$1.07 \cdot 10^{53}$	$4.76 \cdot 10^{53}$	8	1
1	$1.48 \cdot 10^{41}$	$1.33 \cdot 10^{43}$	$2.87 \cdot 10^{6}$	$1.89 \cdot 10^{5}$
2	$3.46 \cdot 10^{30}$	$3.71 \cdot 10^{32}$	$5.56 \cdot 10^{11}$	$3.58 \cdot 10^{11}$
3	$9.06 \cdot 10^{19}$	$1.04 \cdot 10^{22}$	$1.03 \cdot 10^{17}$	$6.78 \cdot 10^{17}$
4	$2.46 \cdot 10^{9}$	$2.89 \cdot 10^{11}$	$1.64 \cdot 10^{22}$	$1.28 \cdot 10^{23}$
5	1.07	8.07	$6.67 \cdot 10^{26}$	$2.43 \cdot 10^{26}$
6	1.00	< 1	$6.90 \cdot 10^{26}$	$> \sqrt{\kappa(A)}$

Table III. Results for Rump matrix with $n = 1000$ and $\kappa(A) \approx 8.30 \cdot 10^{102}$ by the proposed algorithm.

k	$\kappa(X_k^T A X_k)$	$(c_n\mathbf{u})^k \kappa(A)$	$\kappa(X_k)$	$(c_n\mathbf{u})^{-\frac{k}{2}}$
0	$1.92 \cdot 10^{102}$	$8.30 \cdot 10^{102}$	8	1
1	$4.18 \cdot 10^{90}$	$9.24 \cdot 10^{92}$	$2.78 \cdot 10^{6}$	$9.48 \cdot 10^{4}$
2	$2.99 \cdot 10^{80}$	$1.03 \cdot 10^{83}$	$3.07 \cdot 10^{11}$	$8.98 \cdot 10^{9}$
3	$2.73 \cdot 10^{70}$	$1.15 \cdot 10^{73}$	$2.99 \cdot 10^{16}$	$8.51 \cdot 10^{14}$
4	$2.72 \cdot 10^{60}$	$1.28 \cdot 10^{63}$	$2.78 \cdot 10^{21}$	$8.06 \cdot 10^{19}$
5	$2.83 \cdot 10^{50}$	$1.42 \cdot 10^{53}$	$2.50 \cdot 10^{26}$	$7.64 \cdot 10^{24}$
6	$3.01 \cdot 10^{40}$	$1.58 \cdot 10^{43}$	$2.30 \cdot 10^{31}$	$7.24 \cdot 10^{29}$
7	$3.26 \cdot 10^{30}$	$1.76 \cdot 10^{33}$	$2.07 \cdot 10^{36}$	$6.86 \cdot 10^{34}$
8	$3.57 \cdot 10^{20}$	$1.96 \cdot 10^{23}$	$1.90 \cdot 10^{41}$	$6.50 \cdot 10^{39}$
9	$3.94 \cdot 10^{10}$	$2.18 \cdot 10^{13}$	$1.61 \cdot 10^{46}$	$6.16 \cdot 10^{44}$
10	$5.37 \cdot 10^{1}$	$2.43 \cdot 10^{3}$	$1.24 \cdot 10^{51}$	$5.84 \cdot 10^{49}$
11	1.00	< 1	$2.88 \cdot 10^{51}$	$> \sqrt{\kappa(A)}$

`n` $= 500$ with `cnd` $= 10^{50}$ and `n` $= 100$ with `cnd` $= 10^{100}$, respectively. Then $A \in \mathbb{F}^{500\times 500}$ with $\kappa(A) \approx 4.76 \cdot 10^{53}$ and $A \in \mathbb{F}^{1000\times 1000}$ with $\kappa(A) \approx 8.30 \cdot 10^{102}$ are generated, respectively. The results are displayed in Tables II and III, respectively.

In all of the above test cases, the condition number of the input matrices (H_{21} and A) is dropped by a factor around $c_n\mathbf{u}$ in each step until $\kappa(X_k^T A X_k) \approx 1$. It turns out that we obtain an adaptive and robust algorithm for an inverse Cholesky factorization and verification of positive definiteness.

5. Conclusion

We presented an accurate and robust algorithm for inverse Cholesky factorization. The proposed algorithm is based on standard numerical algorithms in pure floating-point arithmetic and an algorithm for accurate dot product. The performance of our algorithm was also evaluated in the numerical experiments.

Using the proposed algorithm, we can also check whether an input symmetric matrix is positive definite or not. If the algorithm stops successfully, then the positive definiteness of the matrix is also proved. Otherwise, the matrix is proved to be not positive semidefinite with the exception of overflow or underflow in floating-point arithmetic.

One may be interested in comparing the proposed algorithm with a standard numerical algorithm by straightforwardly using long precision arithmetic library (for example, GMP [6]) in terms of measured computing time. However, the latter approach does not have any information on the accuracy of the computed results, so that the authors think such a comparison is neither fair nor meaningful. On the other hand, by using MPFR library [9], which is based on GMP, one can implement the verified algorithm in [20] for testing positive definiteness of arbitrarily ill-conditioned symmetric matrices. In this case, the comparison would be interesting and perfectly fair.

Analysis of the proposed algorithm has not been given in this paper. It should be similar to the analysis of the algorithms proposed in [11]. It will be presented as a future work.

Acknowledgments

The authors would like to express their sincere thanks to Prof. Siegfried M. Rump from Hamburg University of Technology for his stimulating discussions. The authors wish to thank the two anonymous referees for their valuable comments.

References

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.

[2] ANSI/IEEE Std 754-1985: IEEE Standard for Binary Floating-Point Arithmetic, IEEE Computer Society, 1985.

[3] ANSI/IEEE Std 754-2008: IEEE Standard for Floating-Point Arithmetic, IEEE Computer Society, 2008.

[4] J. Demmel, *On floating point errors in Cholesky*, LAPACK Working Note 14 CS-89-87, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, 1989.

[5] G.H. Golub and C.F. Van Loan, *Matrix Computations*, *Third ed.*, The Johns Hopkins University Press, Baltimore and London, 1996.

[6] GMP: GNU Multiple Precision Arithmetic Library, version 5.0.1, 2010. Code and documentation available at `http://gmplib.org/`.

[7] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, *2nd ed.*, SIAM, Philadelphia, PA, 2002.

[8] R.E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1966.

[9] MPFR: The GNU MPFR Library, version 3.0.0, 2010. Code and documentation available at `http://www.mpfr.org/`.

[10] A. Neumaier, *Interval Methods for Systems of Equations, Encyclopedia of Mathematics and its Applications*, Cambridge University Press, 1990.

[11] T. Ogita, "Accurate matrix factorization: Inverse LU and inverse QR factorizations," *SIAM J. Matrix Anal. Appl.*, 31:5 pp. 2477–2497, 2010.

[12] T. Ogita, S.M. Rump, and S. Oishi, "Accurate sum and dot product," *SIAM J. Sci. Comput.*, 26:6, pp. 1955–1988, 2005.

[13] S. Oishi, "Fast enclosure of matrix eigenvalues and singular values via rounding mode controlled computation," *Linear Alg. Appl.*, 324:1–3, pp. 133–146, 2001.

[14] S. Oishi, K. Tanabe, T. Ogita, and S.M. Rump, "Convergence of Rump's method for inverting arbitrarily ill-conditioned matrices," *J. Comp. Appl. Math.*, 205:1, pp. 533–544, 2007.

[15] S.M. Rump, "A class of arbitrarily ill-conditioned floating-point matrices," *SIAM J. Matrix Anal. Appl.*, 12:4, pp. 645–653, 1991.

[16] S.M. Rump, *Approximate inverses of almost singular matrices still contain useful information, Forschungsschwerpunktes Informations- und Kommunikationstechnik*, Technical Report 90.1, Hamburg University of Technology, 1990.

[17] S.M. Rump, "Fast and parallel interval arithmetic," *BIT*, 39:3, pp. 534–554, 1999.

[18] S.M. Rump, *INTLAB – INTerval LABoratory, Developments in Reliable Computing (Tibor Csendes ed.)*, Kluwer Academic Publishers, Dordrecht, pp. 77–104, 1999.

[19] S.M. Rump, "Inversion of extremely ill-conditioned matrices in floating-point," *Japan J. Indust. Appl. Math.*, 26:2–3, pp. 249–277, 2009.

[20] S.M. Rump, "Verification of positive definiteness," *BIT Numerical Mathematics*, 46, pp. 433–452, 2006.

[21] S.M. Rump, T. Ogita, and S. Oishi, "Accurate floating-point summation part I: faithful rounding," *SIAM J. Sci. Comput.*, 31:1, pp. 189–224, 2008.

[22] S.M. Rump, T. Ogita, and S. Oishi, "Accurate floating-point summation part II: sign, K-fold faithful and rounding to nearest," *SIAM J. Sci. Comput.*, 31:2, pp. 1269–1302, 2008.

[23] J.H. Wilkinson, "A priori error analysis of algebraic processes," *Proc. International Congress in Mathematics*, Izdat Mir, Moscow, pp. 629–639, 1968.