published in Journal of Computational and Applied Mathematics (JCAM) 199(2): 199–206, 2006

SUPER-FAST VALIDATED SOLUTION OF LINEAR SYSTEMS

SIEGFRIED M. RUMP* AND TAKESHI OGITA†

Abstract. Validated solution of a problem means to compute error bounds for a solution in finite precision. This includes the proof of existence of a solution. The computed error bounds are to be correct including all possible effects of rounding errors. The fastest known validation algorithm for the solution of a system of linear equations requires twice the computing time of a standard (purely) numerical algorithm. In this paper we present a super-fast validation algorithm for linear systems with symmetric positive definite matrix. This means that the entire computing time for the validation algorithm including computation of an approximated solution is the same as for a standard numerical algorithm. Numerical results are presented.

1. Introduction.

Standard methods for the computation of rigorous error bounds for the solution of a linear system $Ax = b$ are based on the Krawczyk operator $K(X) := \tilde{x} + R(b - A\tilde{x}) + (I - RA)X$. Here $\tilde{x} \in \mathbb{R}^n$ denotes an approximate solution of the linear system, R denotes an approximate inverse of A, and $X \in \mathbb{IR}^n$ an n-dimensional interval vector supposed to be a potential inclusion of the solution $A^{-1}b$. If $\|I - RA\| < 1$ for some norm, then [5] $K(X) \subseteq X$ implies $A^{-1}b \in X$. The computational effort is $2n^3$ operations[1] to compute R and $2n^3$ operations each to compute a lower and upper bound for $I - RA$, respectively.

Possibly some iterations $X^{k+1} := K(X^k)$ are necessary to find some $X := X^k$ with $K(X) \subseteq X$. In practice this either happens after few iterations, or the verification process fails. This implies a total computing time of $6n^3 + \mathcal{O}(n^2)$ operations, 9 times as much as for Gaussian elimination with $\frac{2}{3}n^3 + \mathcal{O}(n^2)$ operations. We mention that in practice the measured factor is about 6 to 7 because matrix-matrix multiplication is faster than 3 times LU-decomposition.

Recently, a validation method was presented in [9] based on an LU-decomposition. The additional amount of work for the validation process is the computation of approximate inverses of L and U. This implies a total computing time of $2 \cdot \frac{2}{3}n^3 + \mathcal{O}(n^2)$ operations, or twice the time of Gaussian elimination. We call an algorithm with this property *fast*, i.e. the validation process costs as much as the computation of an (approximate) solution by a standard numerical algorithm.

One may ask whether it is possible to compute rigorous error bounds in the *same* (total) computation time as a standard numerical method, so that validation comes essentially for free. We call an algorithm with this property *super-fast*.

There are such algorithms for some classes of matrices. For example, for A being an M-matrix, the following was presented in [6]. Denote the n-vector of all 1's by e, and let $\tilde{y} \in \mathbb{R}^n$. Then $A^{-1} > 0$ implies $\|A^{-1}\|_\infty = \|A^{-1}e\|_\infty = \|\tilde{y} + A^{-1}(e - A\tilde{y})\|_\infty$, and a standard estimation yields

$$\|A^{-1}b - \tilde{x}\|_\infty \le \|A^{-1}\|_\infty \|b - A\tilde{x}\|_\infty \le \frac{\|\tilde{y}\|_\infty}{1 - \|e - A\tilde{y}\|_\infty} \|b - A\tilde{x}\|_\infty. \tag{1}$$

The computational effort to evaluate the right hand side of (1) is $\mathcal{O}(n^2)$ and therefore negligible compared to a factorization of A. The latter can be used to compute $\tilde{y}$ as an approximation of the solution of $Ay = e$, which requires $\mathcal{O}(n^2)$ operations as well. So this is a super-fast algorithm for bounding the solution of $Ax = b$ in case A is an M-matrix. Clearly, the approach can be used when the signs of the entries of A^{-1} are known, for example for totally positive matrices and others.

* Institute for Realible Computing, Hamburg University of Technology, Schwarzenbergstraße 95, Hamburg 21071, Germany (rump@tu-harburg.de)
† Graduate School of Science and Engineering, Waseda University, 3-4-1 Okubo Shinjuku-ku, Tokyo 169-8555, Japan
[1] addition, subtraction, multiplication, division or square root are counted as one operation

In this paper we will present a super-fast algorithm for the solution of a system of linear equations with symmetric positive definite matrix. We assume an arithmetic according to IEEE 754 which is available nowadays on many if not most computers. Moreover, to compute narrow bounds, accumulation of inner products in some higher precision is used.

Denote the floating point result of a computation by $fl(\cdot)$. This means especially that a relative rounding error unit u and an underflow constant η are given such that

$$\begin{aligned} fl(a \pm b) &= (a \pm b)(1+\delta_1) = (a \pm b)/(1+\delta_2), \\ fl(a \circ b) &= (a \circ b)(1+\delta_1) + \eta_1 = (a \circ b)(1+\delta_2) + \eta_2 \text{ for } \circ \in \{\cdot, /\}, \\ fl(a^{1/2}) &= a^{1/2}(1+\delta_1) = a^{1/2}/(1+\delta_2) \end{aligned} \tag{2}$$

for all floating numbers a, b with constants $|\delta_i| \le u$ and $|\eta_i| \le \eta$. For double precision and rounding to nearest we have $u = 2^{-53}$ and $\eta = 2^{-1074}$. Note that the underflow correction is only necessary for multiplication and division. This is because the square root cannot underflow, and addition and subtraction is *exact* in case of underflow.

The paper is organized as follows. Following we derive some refined analysis for Cholesky decomposition. Based on that, we present a super-fast algorithm for symmetric positive definite linear systems in Section 3, prove its correctness and discuss its behavior. We finish the paper with computational results.

2. Some error estimates.

We first proceed as in standard error analysis. Assume a symmetric matrix $A = (a_{ij})$ to be given and consider the following algorithm.

```
for $j = 1 : n$
    for $i = 1 : j-1$
        $r_{ij} = (a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}) / r_{ii}$
    end
    $r_{jj} = (a_{jj} - \sum_{k=1}^{j-1} r_{kj}^2)^{1/2}$
end
```

ALGORITHM 2.1. *Cholesky decomposition*

This is Cholesky's decomposition which is, up to order of evaluation, identical to any library implementation. We do not assume A to be positive definite; this will follow from later considerations. We say Algorithm 2.1 *runs to completion* if all square roots are real. In this case and in exact arithmetic an upper triangular matrix R is produced with $R^T R = A$.

When executed in finite precision, and if Algorithm 2.1 runs to completion, then approximate $\tilde{R}$ is produced with $\tilde{R}^T \tilde{R} = A + \Delta A$. The residual ΔA satisfies the error estimates [3, (10.4)f.]

$$\begin{aligned} \left| a_{ij} - \sum_{k=1}^{i} \tilde{r}_{ki} \tilde{r}_{kj} \right| &\le \gamma_i \sum_{k=1}^{i} |\tilde{r}_{ki}|\, |\tilde{r}_{kj}| \quad \text{for } i < j \text{ and} \\ \left| a_{jj} - \sum_{k=1}^{j} \tilde{r}_{kj}^2 \right| &\le \gamma_{j+1} \sum_{k=1}^{j} \tilde{r}_{kj}^2. \end{aligned} \tag{3}$$

As usual, define $\gamma_k := ku/(1-ku)$. Those estimates are valid, no matter what the order of evaluation in Algorithm 2.1 is, barring overflow and underflow. Using the symmetry and (3) implies

$$\tilde{R}^T \tilde{R} = A + \Delta A \quad \text{with } |\Delta A| \le \operatorname{diag}(\gamma_2, \ldots, \gamma_{n+1}) |\tilde{R}^T|\, |\tilde{R}|.$$

It does not necessarily imply A to be positive definite. If Algorithm 2.1 runs to completion, then $a_{jj} \ge 0$. So following the analysis in [1], see also [3, Theorem 10.5], we obtain

$$\|\tilde{r}_j\|_2^2 = \tilde{r}_j^T \tilde{r}_j \le a_{jj} + \gamma_{j+1} |\tilde{r}_j^T|\, |\tilde{r}_j|,$$

where $\tilde{r}_j$ denotes the j-th column of $\tilde{R}$. So we have a rigorous estimation of the growth factor of Cholesky decomposition, although all computations are performed in floating point. It follows

$$\|\tilde{r}_j\|_2^2 \le (1-\gamma_{j+1})^{-1} a_{jj} =: d_j.$$

Define the vector $\overline{d}$ by $\overline{d}_j := (\gamma_{j+1}(1-\gamma_{j+1})^{-1} a_{jj})^{1/2}$. Then, with $k = \min(i,j)$,

$$|\Delta a_{ij}| \le \gamma_{k+1} |\tilde{r}_i^T| \, |\tilde{r}_j| \le \gamma_{i+1}^{1/2} \|\tilde{r}_i\|_2 \gamma_{j+1}^{1/2} \|\tilde{r}_j\|_2 \le \overline{d}_i \overline{d}_j,$$

so that

$$\|\Delta A\|_2 \le \| \, |\Delta A| \, \|_2 \le \|\overline{d}\overline{d}^T\|_2 = \overline{d}^T \overline{d} = \sum_{j=1}^{n} \gamma_{j+1}(1-\gamma_{j+1})^{-1} a_{jj} = \sum_{j=1}^{n} \gamma_{j+1} d_j.$$

LEMMA 2.2. *Let $A = A^T$ be given, and suppose Algorithm 2.1 runs to completion when executed in finite precision. Then, barring overflow and underflow, the computed matrix $\tilde{R}$ satisfies*

$$\tilde{R}^T \tilde{R} = A + \Delta A \quad \text{with} \quad \|\Delta A\| \le \sum_{j=1}^{n} \varphi_{j+1} a_{jj},$$

where $\varphi_k := \gamma_k (1-\gamma_k)^{-1}$.

To avoid bad scaling it is preferable to replace A by $D^{-1}AD^{-1}$, where $D := \mathrm{diag}(A)^{1/2}$ [12, 1]. Van der Sluis proved this scaling to be nearly optimal (up to a factor n). To avoid rounding errors one may replace D by a diagonal matrix $\tilde{D}$ with suitable powers of 2 on the diagonal. This improves a theorem by Demmel [1].

THEOREM 2.3. *Suppose Algorithm 2.1 is applied to a symmetric matrix A with $a_{jj} \ge 0$, and set $\varphi_k := \gamma_k(1-\gamma_k)^{-1}$. Then for execution in finite precision and barring overflow and underflow the following is true:*

i) If $\lambda_{\min}(A) \ge \sum_{j=1}^{n} \varphi_{j+1} a_{jj}$, then Algorithm 2.1 runs to completion.

ii) If $\lambda_{\min}(A) < -\sum_{j=1}^{n} \varphi_{j+1} a_{jj}$, then Algorithm 2.1 ends prematurely with an imaginary square root.

PROOF. To *i*) assume $\tilde{r}_{jj} \ge 0$ for $1 \le j \le k-1$ and $\tilde{r}_{kk}$ to be purely imaginary. Then the previous estimates are still true for

$$\tilde{R}_k^H \tilde{R}_k = A_k + \Delta A_k \text{ with real symmetric } \Delta A_k \text{ and } \quad \|\Delta A_k\|_2 \le \sum_{j=1}^{k} \varphi_{j+1} a_{jj},$$

where the subindex k refers to the upper left $k \times k$ submatrix. So standard perturbation theory of eigenvalues of symmetric matrices and $\lambda_{\min}(\tilde{R}_k^H R_k) < 0$ implies

$$\lambda_{\min}(A_k) \le \lambda_{\min}(\tilde{R}_k^H \tilde{R}_k) + \|\Delta A_k\|_2 < \sum_{j=1}^{k} \varphi_{j+1} a_{jj} \le \sum_{j=1}^{n} \varphi_{j+1} a_{jj},$$

and the interlacing theorem yields a contradiction.

To *ii*) assume Algorithm 2.1 runs to completion. Then $\tilde{R}^T \tilde{R} = A + \Delta A$ and

$$\lambda_{\min}(A) \ge \lambda_{\min}(\tilde{R}^T \tilde{R}) - \|\Delta A\|_2 \ge -\sum_{j=1}^{n} \varphi_{j+1} a_{jj}. \qquad \blacksquare$$

3. Super-fast verification for symmetric positive definite matrices.

3. Super-fast verification for symmetric positive definite matrices. Our method computes a Cholesky decomposition of a shifted matrix $A - cI$ rather than A itself, where the shift c will be based on Theorem 2.3. If the floating point decomposition runs to completion, this will imply a lower bound on the smallest singular value of A and thereby proves positive definiteness of A.

For a given linear system $Ax = b$ with symmetric matrix define $\alpha := \sum_{j=1}^{n} \varphi_{j+1} a_{jj}$. Suppose $\alpha > 0$. If Algorithm 2.1 applied to $A - 2\alpha I$ runs to completion, then Theorem 2.3 implies

$$\lambda_{\min}(A) - 2\alpha = \lambda_{\min}(A - 2\alpha I) \geq -\sum_{j=1}^{n} \varphi_{j+1}(a_{jj} - 2\alpha) \geq -\sum_{j=1}^{n} \varphi_{j+1} a_{jj} = -\alpha.$$

This implies $\lambda_{\min}(A) \geq \alpha$ and A to be symmetric positive definite. Therefore $\sigma_{\min}(A) = \lambda_{\min}(A) = \|A^{-1}\|_2^{-1} \geq \alpha$. Hence, for any $\tilde{x} \in \mathbb{R}^n$,

$$\|A^{-1}b - \tilde{x}\|_\infty \leq \|A^{-1}(b - A\tilde{x})\|_2 \leq \|A^{-1}\|_2 \, \|b - A\tilde{x}\|_2 \leq \alpha^{-1}\|b - A\tilde{x}\|_2. \tag{4}$$

That means that (4) is valid for any $\tilde{x} \in \mathbb{R}^n$ provided that the Cholesky decomposition of $A - 2\alpha I$, executed in pure floating-point, runs to completion. This leaves us with the problem of computing an approximate solution $\tilde{x}$ based on a given (approximate) Cholesky decomposition of $A - 2\alpha I$.

Denote $B := A - 2\alpha I$ and assume $\tilde{x}$ is computed by forward and backward substitution using the factorization of B. If as previously mentioned, diagonal scaling is applied, then $a_{jj} \sim 1$ and α is of size $\sum \varphi_{j+1} \sim n^2/2 \cdot u$. Hence, $\tilde{x}$ and $A^{-1}b$ differ by approximately $\operatorname{cond}(A)\alpha$. For ill-conditioned matrix A we need some iterative refinement. Consider

$$x^{k+1} := x^k + B^{-1}(b - Ax^k) \tag{5}$$

with $x^0 := \tilde{x}$. In practical application, multiplication of B^{-1} is, of course, replaced by forward and backward substitution using the Cholesky factors of B. Assuming $\|2\alpha A^{-1}\| < 1$, a standard computation yields

$$\begin{aligned} x^{k+1} &= x^k + (I - 2\alpha A^{-1})^{-1} A^{-1}(b - Ax^k) \\ &= x^k + (I + 2\alpha A^{-1}(I - 2\alpha A^{-1})^{-1}) A^{-1}(b - Ax^k) \\ &= A^{-1}b + 2\alpha A^{-1}(I - 2\alpha A^{-1})^{-1}(A^{-1}b - x^k). \end{aligned}$$

Abbreviating $c := \|2\alpha A^{-1}\|$ shows

$$\|x^{k+1} - A^{-1}b\| \leq \frac{c}{1-c}\|x^k - A^{-1}b\| \leq \left(\frac{c}{1-c}\right)^{k+1} \|\tilde{x} - A^{-1}b\|.$$

So the residual iteration (5) with perturbed iteration matrix $B = A - 2\alpha I$ instead of A behaves similar to the usual residual iteration provided $\|2\alpha A^{-1}\| < 1$, which means $\alpha < \frac{1}{2}\sigma_{\min}(A)$.

Estimation (4) will in general be very poor because α is small. Indeed, α is of order $\|b\|u$, and $\|b - A\tilde{x}\|$ of order $\|b\|u$, so the right hand side of (4) is of order 1. To improve the quality of (4) we store the approximate solution in two parts $\tilde{x}$ and $\tilde{y}$. This approach was used in [10] and later called "staggered correction". The technique makes only sense when a more accurate dot product is available. Then

$$A^{-1}b - \tilde{x} = \tilde{y} + A^{-1}(b - A\tilde{x} - A\tilde{y}),$$

and so (cf. [7])

$$\begin{aligned} |A^{-1}b - \tilde{x}| &\leq |\tilde{y}| + \|A^{-1}\|_2 \|b - A\tilde{x} - A\tilde{y}\|_2 e \\ &\leq |\tilde{y}| + \alpha^{-1}\|b - A\tilde{x} - A\tilde{y}\|_2 e, \end{aligned} \tag{6}$$

where e denotes the n-vector of all 1's. To apply (6), we first improve $\tilde{x}$ by residual iteration, where the residual $A\tilde{x} - b$ is computed in higher precision. This can be performed using the algorithms proposed in

[8]. For the computation of an error bound for the residual $\|b - A\tilde{x} - A\tilde{y}\|_\infty$ in 6) working precision suffices. The inclusion algorithm is given in Algorithm 3.1. Here setround(i) switches the rounding mode to nearest for $i = 0$, towards $+\infty$ for $i = 1$ and towards $-\infty$ for $i = -1$. This implies that all floating point operations are calculated in that rounding mode until the next call of setround. Based on this is the following INTLAB [11] implementation. Step 1) is based on van der Sluis' result on optimal symmetric diagonal scaling.

```
Input: $A = A^T \in \mathbb{R}^{n\times n}$ with $a_{jj} > 0$, $b \in \mathbb{R}^n$
1)  If $\max a_{jj} / \min a_{jj} > n$
        $D := \operatorname{diag}(d_j)$ with $d_j = 2\hat{}(-\mathrm{round}(0.5\log_2 a_{jj}))$
        $A = DAD$; $b = Db$; scale $= 1$;
    else  scale$=0$
    end
2)  setround$(+1)$; $\alpha = \sum_{j=1}^{n} \varphi_{j+1} a_{jj}$;
    setround$(-1)$; $B = A - 2\alpha I$
    setround$(0)$
3)  $R = \mathtt{chol}(B)$
4)  $x = R\backslash(R^T\backslash b)$
    iterate $x = x + R\backslash(R^T\backslash \mathrm{dot}(b - Ax))$ until "sufficiently" accurate
    $[\underline{\mathrm{res}}, \overline{\mathrm{res}}] = \mathrm{dot}_\Diamond(b - Ax)$
    $y = R\backslash(R^T\backslash \underline{\mathrm{res}})$
    $y = y + R\backslash(R^T\backslash \mathrm{dot}(\underline{\mathrm{res}} - Ay))$
5)  $x_m = x$
    $x_r = |y| + \|Ay - [\underline{\mathrm{res}}, \overline{\mathrm{res}}]\|_2 e/\alpha$
    if scale, $x_m = Dx_m$; $x_r = Dx_r$; end
```

ALGORITHM 3.1. *Super-fast validation algorithm for symmetric (positive definite) matrix*

All operations in Algorithm 3.1 are pure floating point except the computation of $[\underline{\mathrm{res}}, \overline{\mathrm{res}}]$ in step 4) and the computation of x_r in step 5). The routine "dot" depicts some method to calculate dot products in higher precision, preferably doubled working precision, "dot$_\Diamond$" calculates an inclusion of the result. An elegant way to perform this using only working precision is given in [8]. Note A is only assumed to be symmetric; positive definiteness is shown a posteriori by Algorithm 3.1.

The calculation of the residual $Ay - \mathrm{res}$ is performed in working precision, so the size is of order $\|\mathrm{res}\|u \approx u^2$ and small enough for a good overall error bound for the solution $A^{-1}b$. We have to prove that Algorithm 3.1 computes a validated error bound.

THEOREM 3.2. *Let symmetric $A \in \mathbb{R}^{n\times n}$ with $a_{ii} > 0$ and $b \in \mathbb{R}^n$ be given. Assume Algorithm 3.1 runs to completion. Then, barring overflow and underflow, the matrix A is positive definite and the solution $A^{-1}b$ of the linear system $Ax = b$ satisfies*

$$|A^{-1}b - x_m| \le x_r$$

for the computed vectors x_m and x_r.

PROOF. Any computation with the matrix D in steps 1) and 5) is exact since the d_j are powers of 2. Therefore we may assume without loss of generality $D = I$ for the following analysis. Denote the quantities computed by Algorithm 3.1 by $\tilde{\alpha}, \tilde{B}$ etc. Then, due to the rounding in use,

$$\lambda_{\min}(A) - 2\tilde{\alpha} = \lambda_{\min}(A - 2\tilde{\alpha} I) \ge \lambda_{\min}(\tilde{B})$$

because $\tilde{\alpha} \ge \sum \varphi_{j+1} a_{jj} > 0$. Therefore, because Cholesky decomposition applied to $\tilde{B}$ runs to completion,

TABLE 4.1

Results for $A = diag(-1, 2, -1)$

n	cond(A)	iter	ratio	ratio'	max rel err X
500	1.3e5	3	56	13	3.3e-16
1000	5.0e5	4	67	14	3.3e-16
2000	2.0e6	5	77	16	3.3e-16
5000	1.3e7	9	119	23	3.3e-16
10000	5.0e7	29	326	58	9.0e-15

TABLE 4.2

Matrices NOS from structural engineering*

name	n	p	cond(A)	iter	ratio	ratio'	max rel err X
NOS2	957	4.3	6e9	12	94	18	6.8e-10
NOS3	960	16.5	7e4	4	1.90	1.36	3.2e-16
NOS6	675	4.8	8e6	4	2.07	1.58	1.1e-13
NOS7	729	6.3	4e9	7	1.39	1.27	1.8e-14

Lemma 2.2 implies

$$\begin{aligned}\lambda_{\min}(A) - 2\tilde{\alpha} &\geq \lambda_{\min}(\tilde{B}) \geq -\sum \varphi_{j+1}\tilde{b}_{jj} \geq -\sum \varphi_{j+1}(a_{jj} - \tilde{\alpha}) \\ &\geq -\sum \varphi_{j+1}a_{jj} \geq -\tilde{\alpha},\end{aligned}$$

so that

$$\lambda_{\min}(A) \geq \tilde{\alpha} > 0.$$

Hence, $\lambda_{\min}(A) = \sigma_{\min}(A) = \|A^{-1}\|_2^{-1} \geq \tilde{\alpha} > 0$, and by (6) and proper use of directed rounding it follows

$$\tilde{x}_m - \tilde{x}_r \leq \tilde{x} + \tilde{d} \leq \tilde{x}_m + \tilde{x}_r. \qquad \blacksquare$$

4. Numerical results. A standard model problem is the 3-point second difference operator diag(-1,2,-1). For various dimensions n we generate right hand sides such that the approximate solution is $e = (1, \ldots, 1)^T$. In the following Table 4.1 "ratio" denotes the ratio of the total number of floating point operations (flops) of our Algorithm 3.1 and the number of flops for Cholesky decomposition (Algorithm 2.1, `chol` in Matlab). Note that

- our flop count takes sparsity into account, and
- we use Algorithm 5.3 (Dot2) in [8] for Dot requiring $25k$ flops for quadruple precision evaluation of a dot product of length k.

So "ratio" is the ratio in computing time which is achieved *when only working precision is available.* Note that the used algorithm from [8] is the fastest available (for example, XBLAS [4] requires $37k$ flops). The following column ratio' displays the ratio in computing time when quadruple precision is available. Furthermore, cond(A) denotes the estimated condition number by Matlab routine `condest`, iter the number of iterations in step 4) of Algorithm 3.1, and max rel err X the maximum relative error over all inclusion components $X = [x_m - x_r, x_m + x_r]$.

The comparison in Table 4.1 is not really fair because the matrix has only bandwidth 2 so that the Cholesky decomposition requires less than $6n$ flops. Hence the residual iterations become very expensive. We tuned the iteration in step 4) of Algorithm 3.1 for high precision error bounds. For less accurate inclusions the ratios look a little better.

TABLE 4.3
*Matrices *BUS from power system networks*

name	n	p	cond(A)	iter	ratio	ratio'	max rel err X
662BUS	662	3.7	8.3e5	3	2.17	1.54	3.3e-16
685BUS	685	4.7	5.9e5	3	1.83	1.46	3.3e-16
1138BUS	1138	3.6	1.2e7	4	1.37	1.39	2.2e-16

TABLE 4.4
Matrices BCSSTK from static analysis of structural engineering*

name	n	p	cond(A)	iter	ratio	ratio'	max rel err X
BCSSTK14	1806	35.1	1.3e10	3	1.28	1.12	1.4e-13
BCSSTK15	3948	29.8	8.0e9	4	1.09	1.06	9.3e-15
BCSSTK16	4884	59.5	7.0e9	3	1.34	1.12	9.4e-15
BCSSTK17	10974	39.1	2.0e10	6	1.24	1.11	2.9e-13
BCSSTK18	11948	12.5	6.5e11	4	1.04	1.03	4.3e-14

We turn to more practical examples. The following matrices are taken from the Harwell-Boing collection [2]. The first set of examples stem from structural engineering. We now display in addition the average number p of nonzero entries per row. Except the first row in Table 4.2 the numbers are not so bad. The first example NOS2 is a band matrix with bandwidth 4 and not too well conditioned. So the increased number of iterations implies the poor ratio because Cholesky decomposition requires very few flops.

Finally, we show some examples of larger dimension. They arose in power system networks and structured engineering. The results show the expected behavior that for increasing dimension the ratios (slowly) approach 1.

REFERENCES

[1] J.B. 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.

[2] I.S. Duff, R.G. Grimes, and J.G. Lewis. User's guide for Harwell- Boeing sparse matrix test problems collection. Technical Report RAL-92-086, Computing and Information Systems Department, Rutherford Appleton Laboratory, Didcot, UK, 1992.

[3] N.J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM Publications, Philadelphia, 2nd edition, 2002.

[4] X. Li, J. Demmel, D. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Kang, A. Kapur, M. Martin, B. Thompson, T. Tung, and D. Yoo. Design, Implementation and Testing of Extended and Mixed Precision BLAS. *ACM Trans. Math. Softw.*, 28(2):152–205, 2002.

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

[6] T. Ogita, S. Oishi, and Y. Ushiro. Fast verification of solutions for sparse monotone matrix equations. volume 15 of *Comput. Suppl.*, pages 175–187. Springer, Wien, 2001.

[7] T. Ogita, S. Oishi, and Y. Ushiro. Computation of sharp rigorous componentwise error bounds for the approximate solutions of systems of linear equations. *Reliable Computing*, 9(3):229–239, 2003.

[8] T. Ogita, S.M. Rump, and S. Oishi. Accurate Sum and Dot Product. *SIAM Journal on Scientific Computing (SISC)*, 26(6):1955–1988, 2005.

[9] S. Oishi and S.M. Rump. Fast verification of solutions of matrix equations. *Numer. Math.*, 90(4):755–773, 2002.

[10] S.M. Rump. *Kleine Fehlerschranken bei Matrixproblemen.* PhD thesis, Universität Karlsruhe, 1980.

[11] S.M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999.

[12] J.H. Wilkinson. A Priori Error Analysis of Algebraic Processes. *Proc. International Congress Math. (Moscow: Izdat Mir,)*, pages 629–639, 1968.