Linear Systems in Matlab with Zero Residual*

Siegfried M. Rump §
Institute for Reliable Computing
Hamburg University of Technology
Schwarzenbergstraße 95, Hamburg 21071, Germany,
and
Visiting Professor at Waseda University
Faculty of Science and Engineering
3–4–1 Okubo, Shinjuku-ku, Tokyo 169–8555, Japan
rump@tuhh.de

Abstract

Several examples of linear systems are given for which the residual of the computed approximation (computed in working precision) is entirely zero. Nevertheless, the computed approximation by Matlab's backslash operator is highly inaccurate, and often no warning is issued. Examples for matrix inversion in that spirit are given as well. Moreover, it is demonstrated that rounding the exact real result to the nearest floating-point numbers may be worse than using an approximation computed by some floating-point algorithm. Concluding, some advice is given to users and developers.

Keywords: Matlab, backslash operator, residual, ill-conditioned, linear system, matrix inversion
AMS subject classifications: 15A06, 15A04

1 Introduction

To begin with I want to stress that nothing in this note comes as a surprise to experts in numerical analysis. However, ordinary users may be astonished that elementary rules of linear algebra such as that a singular matrix cannot be inverted are no longer valid in the regime of floating-point computations. It may be helpful to have a small collection of explicit examples in that direction at hand.

In [7] examples of simple numerical problems are listed where traditional numerical algorithms may compute partially wrong or even completely wrong approximations. Worst of all, occasionally that happens without a warning issued. Moreover, for some examples the (in working precision) computed residual was entirely zero, possibly supporting an illusion of additional security.

In this note we are particularly interested in systems of linear equations and the solution by Matlab [4]. We hasten to stress that this is no campaign against Matlab's

*Submitted: January 15, 2018; Revised: May 24, 2018; Accepted:June 4, 2018).
§This research was partially supported by CREST, Japan Science and Technology Agency.

algorithms but describes a fundamental problem of floating-point calculations. We give examples of (ill-conditioned) linear systems for which inaccurate approximations are computed, but nevertheless the computed residual is entirely zero. In that case a residual iteration does not change the approximation. It is particularly hazardous that often inaccurate results are computed without warning.

The backslash operator in Matlab changed several times over the years. For our examples we tested all Matlab releases from 2008a through 2017b. We state exemplary results for release 2012a and 2017b, and comment on between releases. As details of the algorithms are confidential, we cannot point to specifics but have to use the algorithms as a black box.

Matlab's `inv` function changed as well. Examples are given with large residual for not too ill-conditioned matrices. Moreover, using the exact inverse rounded to the nearest floating-point matrix may be worse than using `inv(A)`.

One may ask why we should be interested in extremely ill-conditioned linear systems at all, in particular because often the input data is not exactly given, and a small perturbation can produce a singular problem. Although the latter does not apply to integer entries, we do not know the condition number beforehand, and all condition estimators come with a set of counterexamples. Given that there still may be not too much harm if a warning is issued; if not, further use of wrong results may be hazardous.

For the displayed results, all computations are performed in Matlab's default precision binary64 corresponding to about 16 decimal digits. In tables displaying numerical results, "residual" refers to the residual computed in working precision. Otherwise, all results except of Matlab's backslash operator and Matlab's `inv` function are computed with verification using INTLAB [8], the Matlab/Octave toolbox for Reliable Computing, and/or using some multiple precision package. That applies in particular to residual norms such as $\|A\tilde{x} - b\|$ or $\|I - RA\|$, where we always use the 2-norm. All displayed results are correctly rounded to the displayed precision.

We close this note with some recommendations to users and to experts writing software for the approximate solution of linear systems. Some small changes/additions would improve the situation, though there is no panacea.

2 Some examples

For the random generation of ill-conditioned matrices, we cannot use the standard way by `randsvd` in Matlab's matrix gallery because we are interested in condition numbers beyond 10^{16}. That is because rounding into the floating-point screen has a certain smoothing effect.

Take, for example, a random integer matrix and define the last row to be the sum of the first $n-1$ rows. Then, divide that singular matrix A by some φ not being a power of 2, and round the result to the nearest matrix in $\mathbb{F}^{n\times n}$. The $\log_{10}$ of the resulting (true) condition number is shown in Table 1. Obviously, rounding to nearest limits the condition number roughly to $\mathbf{u}^{-1}$, where $\mathbf{u}$ denotes the unit of roundoff error $\mathbf{u} = 2^{-53} \approx 10^{-16}$. To construct more ill-conditioned matrices with floating-point entries, a random matrix with small integer entries is generated such that $\kappa(A^{2^k})$ corresponds roughly to the anticipated condition number.

A side effect is that such matrices typically have one single very small singular value. As a consequence, the numerical solution $A\backslash b$ of a linear system is typically a scalar multiple of the true solution $A^{-1}b$. That is seen as follows.

φ	n=3	n=5	n=10	n=50	n=100
3	17.6	17.1	17.4	17.9	18.5
5	16.4	17.1	17.6	18.9	18.6
6	16.8	17.6	17.1	17.9	18.2
7	16.7	17.3	17.3	18.4	17.9
9	16.9	16.9	17.5	17.9	18.4

Table 1: $\log_{10} \kappa(\mathrm{fl}(A/\varphi))$ for singular $A \in \mathbb{Z}^{n\times n}$

All elements of the L- and U-factor of A are typically of moderate size, and only $\alpha := U_{nn}$ is very small in absolute value compared to all other elements in L and U. Thus, $U^{-1} \approx \alpha^{-1} \cdot ve_n^T$ is basically a rank-1 matrix, where (in Matlab notation) $v := [-U(I,I)^{-1} * U(I,n); 1]$ with $I := 1 : n-1$. Note that v does not depend on α. The computed factors $\tilde{L}$ and $\tilde{U}$ are very good approximations of the true factors L and U, except $\beta := \tilde{U}_{nn}$. Hence, the inverse of the computed factor changes into $\tilde{U}^{-1} \approx \beta^{-1} \cdot ve_n^T$, basically the true U^{-1} up to some constant factor.

The generation of very ill-conditioned matrices $A \in \mathbb{F}^{n\times n}$ with $\kappa(A)$ way above $\mathbf{u}^{-1}$ and prescribed singular values seems difficult, cf. [6].

2.1 Zero residual and no warning for older Matlab releases

In [7], in particular Example 6, the linear system $Ax = b$ with

$$A = \begin{pmatrix} 64919121 & -159018721 \\ 41869520.5 & -102558961 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{1}$$

received some attention [5] because an erroneous approximation is computed, but nevertheless the residual `A*x-b` is identically zero. Table 2 shows the approximate solution $\tilde{x}$ computed by $A\backslash b$ in Matlab 2012a and Matlab 2017b, together with the true solution computed by INTLAB's `verifylss` [8] using the option `'illco'`. INTLAB's verified inclusion is accurate to 15 decimal figures; all the displayed results are rounded to four decimal digits.

Matlab 2012a		Matlab 2017b		
$\tilde{x}$	residual	$\tilde{x}$	residual	$A^{-1}b$
$1.060 \cdot 10^8$	0	$1.459 \cdot 10^8$	0.566	$2.051 \cdot 10^8$
$0.433 \cdot 10^8$	0	$0.596 \cdot 10^8$	-0.037	$0.837 \cdot 10^8$

Table 2: Approximate and true solution of (1)

Matlab 2012a computes the erroneous approximation without warning. The backslash operator in Matlab changed several times, but from release 2008a through 2015b the residual was always entirely zero and no warning was issued. From 2016a, the approximate solution is still completely wrong due to the large (2-norm) condition number $\kappa(A) = 8.35 \cdot 10^{16}$, however, the residual became non-zero and a warning is issued.

We hasten to mention that, from a numerical point of view, the approximate solution $\tilde{x}$ is perfectly backward stable. The backward error for normwise and componentwise relative perturbations of the matrix and right-hand side is given by Rigal/Gache's and Oettli/Prager's formulas [3]

$$\eta(\tilde{x}) = \frac{\|A\tilde{x} - b\|}{\|A\|\|\tilde{x}\| + \|b\|} \quad \text{and} \quad \omega(\tilde{x}) = \max_i \frac{|A\tilde{x} - b|_i}{(|A||\tilde{x}| + |b|)_i},$$

respectively. For (1) these compute to $\eta(\tilde{x}) = 6.63 \cdot 10^{-18}$ and $\omega(\tilde{x}) = 1.09 \cdot 10^{-17}$ for 2017b's approximation, and for $\tilde{x}$ computed by release 2012a the numbers are similar. Thus, both $\tilde{x}$ are the true solution of a linear system with matrix and right-hand side normwise and/or componentwise perturbed way below the relative rounding error unit **u**. In that sense and in view of the condition number, both approximations $\tilde{x}$ in Table 2 are about the best a numerical algorithm can produce.

2.2 Zero residual for the newest Matlab release

Consider

$$A = \begin{pmatrix} -14637 & 14789 & -13910 & -1402 \\ 32357 & -75618 & 82279 & 8408 \\ 53498 & -103555 & 120215 & 12332 \\ 5962 & -10784 & 12812 & 1316 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \qquad (2)$$

The condition number of A is $\kappa(A) = 3.29 \cdot 10^{17}$, and the matrix has two real and two complex eigenvalues. Matlab computes an approximate solution $\tilde{x}$ with about one correct digit according to the following Table 3.

Matlab 2012a		Matlab 2017b		
$\tilde{x}$	residual	$\tilde{x}$	residual	$A^{-1}b$
$0.0021 \cdot 10^9$	0.00098	$0.0027 \cdot 10^9$	0	$0.0025 \cdot 10^9$
$-0.0295 \cdot 10^9$	0	$-0.0375 \cdot 10^9$	0	$-0.0351 \cdot 10^9$
$-0.4406 \cdot 10^9$	0	$-0.5594 \cdot 10^9$	0	$-0.5236 \cdot 10^9$
$4.0383 \cdot 10^9$	-0.00098	$5.1271 \cdot 10^9$	0	$4.7986 \cdot 10^9$

Table 3: Approximate and true solution of (2)

Now the residual $A\tilde{x} - b$ is identically zero for the latest Matlab release 2017b, but non-zero for earlier releases. The normwise and componentwise backward errors are $\eta(\tilde{x}) = 1.49 \cdot 10^{-18}$ and $\omega(\tilde{x}) = 9.02 \cdot 10^{-17}$, respectively, so that again there exists a real matrix $\hat{A}$ and right-hand side $\hat{b}$ such that $\hat{A}\tilde{x} = \hat{b}$, and the rounded to nearest results of $\hat{A}$ and $\hat{b}$ are A and b as in (2), respectively.

2.3 Zero residual for all Matlab release

One may ask whether for *all* releases of Matlab between 2008a and 2017b the residual can be zero. That is indeed possible. Exemplary consider

$$A = \begin{pmatrix} 3300958 & 1040363 & 452656 \\ 1040363 & 2934401 & 1198768 \\ 452656 & 1198768 & 489984 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{3}$$

The matrix is symmetric positive definite with condition number $\kappa(A) = 1.76 \cdot 10^{17}$, and Matlab computes approximate solutions $\tilde{x}$ according to Table 4.

Matlab 2012a		Matlab 2017b		
$\tilde{x}$	residual	$\tilde{x}$	residual	$A^{-1}b$
$-0.016 \cdot 10^{10}$	0	$-0.016 \cdot 10^{10}$	0	$-0.032 \cdot 10^{10}$
$-0.696 \cdot 10^{10}$	0	$-0.696 \cdot 10^{10}$	0	$-1.362 \cdot 10^{10}$
$1.718 \cdot 10^{10}$	0	$1.718 \cdot 10^{10}$	0	$3.361 \cdot 10^{10}$

Table 4: Approximate and true solution of (3)

For example (3), the computed residual $A\tilde{x} - b$ is identically zero for all Matlab releases from 2008a through 2017b. The eigenvalues of A are $4.50 \cdot 10^6$, $2.23 \cdot 10^6$ and $2.56 \cdot 10^{-11}$. The normwise and componentwise backward errors are $\eta(\tilde{x}) = 1.24 \cdot 10^{-17}$ and $\omega(\tilde{x}) = 2.49 \cdot 10^{-17}$, respectively, so that again both approximations are backward stable. For this example, Matlab 2017a and 2017b produce a warning that the matrix may be ill-conditioned, but all other releases from 2012a onward compute the wrong approximation without warning.

For that example, the Matlab 2012a and 2017b approximations differ only in the first component and only by two bits (using INTLAB's `getbits`):

```
x̃1(2012a) = -1.0011010011110001110001100110011001011000111010011001 * 2^27
x̃1(2017b) = -1.0011010011110001110001100110011001011000111010010111 * 2^27
```

However, despite the zero residuals illustrated in Table 4, both approximations are off by about a factor 2 compared to the true solution.

2.4 Zero residual for all Matlab releases from 2012a on and approximations with different signs

Next we ask whether older and newer Matlab releases may produce very different solutions but both with zero residual. That is indeed also possible. Consider

$$A = \begin{pmatrix} 22398 & 47002 & -3639 & -7728 & 2160 \\ 47002 & 122427 & -14358 & -13584 & 4268 \\ -3639 & -14358 & 4573 & 1145 & -518 \\ -7728 & -13584 & 1145 & 3756 & -993 \\ 2160 & 4268 & -518 & -993 & 274 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{4}$$

The matrix is again symmetric and positive definite, and the condition number of A is $\kappa(A) = 1.03 \cdot 10^{20}$. Now Matlab computes approximate solutions $\tilde{x}$ according to the following Table 5.

Matlab 2012a		Matlab 2017b		
$\tilde{x}$	residual	$\tilde{x}$	residual	$A^{-1}b$
$-0.0055 \cdot 10^9$	0	$0.0082 \cdot 10^9$	0	$0.0785 \cdot 10^9$
$-0.0497 \cdot 10^9$	0	$0.0745 \cdot 10^9$	0	$0.7119 \cdot 10^9$
$0.6954 \cdot 10^9$	0	$-1.0431 \cdot 10^9$	0	$-9.9666 \cdot 10^9$
$3.8361 \cdot 10^9$	0	$-5.7541 \cdot 10^9$	0	$-54.9780 \cdot 10^9$
$16.0339 \cdot 10^9$	0	$-24.0508 \cdot 10^9$	0	$-229.7945 \cdot 10^9$

Table 5: Approximate and true solution of (4)

```
function [A,b] = linsys(n,T)
  h = T/(n-1);
  s = (0:n-1)'*h;
  A = -h * tril(ones(n),-1);
  A = A + (1-h/2) * eye(n);
  A(:,1) = -h/2 * ones(n,1);
  A(1,1) = 1;
  A(:,n) = -ones(n,1);
  A(n,n) = -h/2;
  b = -s;
```

Figure 1: Discretization of $x' = x - 1$ in $[0, T]$ with $x(0) = x(T)$ by trapezoidal rule

The residual $A\tilde{x} - b$ is identically zero for Matlab releases 2012a and 2017b, but the two Matlab approximations have opposite signs in all components. Moreover, both are off by at least one order magnitude compared to the true solution.

In fact, the residual is entirely zero for all releases from 2008a through 2017b except 2009b. In any case, the approximate solution is completely inaccurate.

The normwise and componentwise backward errors are $\eta(\tilde{x}) = 9.39 \cdot 10^{-19}$ and $\omega(\tilde{x}) = 3.71 \cdot 10^{-17}$, respectively, so that again both approximations are backward stable. For that example, both releases 2012a and 2017b of Matlab produce a warning that the matrix may be ill-conditioned.

2.5 Wrong approximation and no warning for all Matlab releases from 2012a on

It is well-known that, strictly speaking, Gaussian elimination with partial pivoting is not backward stable. However, vast experience suggests that circumstances leading to instability, i.e., a large growth factor, are extremely rare. Nevertheless, the worst case growth factor is 2^{n-1}.

Until 1993, it was common belief that such a growth occurs only in constructed examples. However, Wright [10] showed the contrary. Based on his work, Foster [2] considered the discretization using the trapezoidal rule of the boundary value problem $x' = x - 1$ in $[0, T]$ with $x(0) = x(T)$. He gave the program in Figure 1 to compute a linear system for n meshpoints.

For $n = 65$ and $T = 40$, a matrix A and right-hand side b is computed such that

$16A \in \mathbb{Z}^{n\times n}$ and $8b \in \mathbb{Z}^n$. Thus, all data are exactly representable. The true solution of $Ax = b$ is the vector of all ones. The results by Matlab are shown in Table 6. The results for other releases are similarly inaccurate. All releases from 1999 through 2017b produce the result without warning.

$\tilde{x}$ by Matlab 2012a	$\tilde{x}$ by Matlab 2017b	`xnew`	$A^{-1}b$
1.0000	1.0000	1.0000	1
1.0000	1.0000	1.0000	1
1.0000	1.0000	1.0000	1
...	...	...	
1.0000	1.0000	1.0000	1
1.0001	1.0000	1.0000	1
1.0002	1.0000	1.0000	1
1.0002	1.0000	1.0000	1
1.0002	1.0000	1.0000	1
1.0007	1.0000	1.0000	1
1.0014	1.0000	1.0000	1
1.0028	1.0000	1.0000	1
1.0057	1.0057	1.0000	1
1.0114	1.0000	1.0000	1
1.0227	1.0000	1.0000	1
1.0455	1.0000	1.0000	1
1.0909	1.0000	1.0000	1
1.1818	1.0909	1.0000	1
1.4545	1.0909	1.0000	1
1.4545	0.7273	1.0000	1
2.1818	0.7273	1.0000	1
2.9091	-1.4545	1.0000	1
0	-2.9091	1.0000	1
-5.8182	-5.8182	1.0000	1
-11.6264	0	1.0000	1
-23.2727	-23.2727	1.0000	1
0	-46.5455	1.0000	1
-93.0909	0	1.0000	1
1.0000	1.0000	1.0000	1

Table 6: Approximate and true solution (leading 3 and trailing 25 components) for $Ax = b$ as in Table 1 for $n = 65$ and $T = 40$, and `xnew` by (5)

The matrix is well-conditoned with $\kappa(A) = 29.5$, correctly computed by `cond(A)`. Nevertheless, some components of the approximations are completely wrong. Therefore, the numerical instability is solely due to the instability of Gaussian elimination with partial pivoting, not due to the difficulty of the problem.

No warning is issued, and neither the condition number nor the residual give any hint to a problem, but the growth factor [3] of $6.5 \cdot 10^{17}$ would.

Solving the linear system with total pivoting produces an approximation with very

small error or even the exact solution in all Matlab releases from 1999 onwards.

Moreover, it is well-known [9] that one residual iteration in working precision produces a backward stable result. Indeed, for our example

$$\texttt{xnew} = \texttt{x} + \texttt{A}\backslash(\texttt{b} - \texttt{A} * \texttt{x}) \tag{5}$$

produces not only a backward stable result, but the forward error of `xnew` is of the order 10^{-15} for all Matlab releases from 2008a through 2017b.

2.6 Rounded-to-Nearest inverse of a matrix

Denote by I the identity matrix of appropriate size. As is well-known, a small componentwise relative error to the true inverse A^{-1} does not imply a small residual. In fact, the true inverse of a matrix rounded to the nearest floating-point matrix $\mathrm{fl}(A^{-1})$ may be worse than an approximate inverse computed by `inv(A)`. Consider the following example.

$$A = \begin{pmatrix} 0.3917 & 0.0865 & -0.7947 \\ -0.1725 & -0.0381 & 0.3501 \\ 0.1021 & 0.0225 & -0.2071 \end{pmatrix}. \tag{6}$$

The matrix components are not integers, therefore we reproduce the hex data.

$$A = \begin{pmatrix} \texttt{3fd911327cf431e7} & \texttt{3fb6242dbf5aaf53} & \texttt{bfe96e12edafe06f} \\ \texttt{bfc61579dd14b36b} & \texttt{bfa381996f4dcdad} & \texttt{3fd6674ca2d9da97} \\ \texttt{3fba2207efb6ca3a} & \texttt{3f97152aa09b00e1} & \texttt{bfca82db4d1be392} \end{pmatrix}.$$

The condition number is $\kappa(A) = 8.64 \cdot 10^{17}$. The approximate inverse `inv(A)`, computed in Matlab's release 2017b, and the rounded-to-nearest true inverse $\mathrm{fl}(A^{-1})$ are shown in Table 7. Both displayed matrices have to be scaled by a factor 10^{17} to obtain the correct data.

$$\begin{pmatrix} 1.6799 & 4.4570 & 1.0873 \\ -1.3969 & -3.7062 & -0.9042 \\ 0.6759 & 1.7933 & 0.4375 \end{pmatrix} \text{ and } \begin{pmatrix} 2.1813 & 5.7873 & 1.4119 \\ -1.8139 & -4.8125 & -1.1741 \\ 0.8777 & 2.3286 & 0.5681 \end{pmatrix}.$$

Table 7: `inv(A)` and $\mathrm{fl}(A^{-1})$, both scaled by a factor 10^{-17}

Obviously, hardly any digit of the approximate floating-point inverse `inv(A)` is correct. Nevertheless, the residual, as shown in Table 8, for Matlab's approximate inverse `inv(A)` is less than 1; thus a residual iteration converges. Table 8 displays the mathematically correct results using the computed R given in the first column.

However, for the floating-point matrix $R := \mathrm{fl}(A^{-1})$ closest to the true inverse A^{-1}, the residual $\|I - RA\|$ is considerably larger than 1; see Table 8. Naturally, the componentwise relative error of R to A^{-1} is below $\mathbf{u}$. Although the quality of the approximate inverse is poor, it suffices to compute an inclusion of a linear system with the matrix A including the proof of non-singularity of A. Consider the INTLAB statements

R	$\|I - RA\|$	$\max_{ij} \frac{\|(A^{-1})_{ij} - R_{ij}\|}{\|(A^{-1})_{ij}\|}$
`inv(A)`	0.26	0.23
$\mathrm{fl}(A^{-1})$	23.1	$6.66 \cdot 10^{-17}$

Table 8: Results for the matrix in (6)

```
R = inv(A); Res = AccDot(-1,eye(3),R,A,[]), normres = norm(res)
X = infsup(-1,1)*ones(3,1); Y = Res*X
```

The result is

```
intval Res =
[   -0.0385,   -0.0384] [    0.1653,    0.1654] [   -0.1342,   -0.1341]
[    0.1644,    0.1645] [   -0.1083,   -0.1082] [   -0.1574,   -0.1573]
[    0.0070,    0.0071] [    0.0714,    0.0715] [   -0.0997,   -0.0996]
intval normres =
[    0.2558,    0.2559]
intval Y =
[   -0.3379,    0.3379]
[   -0.4301,    0.4301]
[   -0.1782,    0.1782]
```

including a warning that the matrix is close to singular. As can be seen, `Y` is included in the interior of `X`, proving that A is non-singular. For the floating-point matrix $R := \mathrm{fl}(A^{-1})$ closest to the true inverse A^{-1}, proof of non-singularity is obviously not possible since the norm of the residual is larger than 1.

2.7 Approximate inverse of a matrix

We want to stress again that, as throughout the paper, all following results are the mathematically correctly rounded results based on the computed data.

For a computed approximate inverse R of a matrix A, we can expect that $\|I - RA\|$ is of the order $\mathbf{u}\|R\|\|A\|$. In fact, in some sense $\|I - RA\| \approx \mathbf{u}\|R\|\|A\|$ characterizes a stable algorithm for computing an approximate inverse R.

Indeed, let `A = hilb(10)` be the approximate Hilbert 10×10 matrix. Then the latest Matlab release 2017b computes `R = inv(A)` with

$$\|I - RA\| = 2.04 \cdot 10^{-4} \qquad \text{and} \qquad \mathbf{u}\|R\|\|A\| = 1.78 \cdot 10^{-3}.$$

That is consistent with the condition number $\kappa(A) = 1.60 \cdot 10^{13}$. The results are similar for Matlab releases 2008a through 2012b, and for 2015b through 2017b.

However, for releases 2013a through 2015a, the `inv` routine in Matlab was changed. The results for the Hilbert 10×10 matrix are shown in Table 9.

Now the approximate inverse computed by `inv(a)` in 2013a is closer to the true inverse as by Table 9, but the residual is way beyond 1. In fact, the median of all $|I - RA|_{ij}$ is 689.1.

For all releases from 2013a through release 2015a the residual is of the order 10^5, so that a residual iteration using R does not converge. Note that there is no difference when treating R as a right inverse because both A and R are symmetric.

	$\|I - RA\|$	$\mathbf{u}\|R\|\|A\|$	$\max_{ij} \frac{\|(A^{-1})_{ij} - R_{ij}\|}{\|(A^{-1})_{ij}\|}$
2013a	$1.48 \cdot 10^{+5}$	$1.78 \cdot 10^{-3}$	$2.81 \cdot 10^{-5}$
2013a columnwise	$8.24 \cdot 10^{-5}$	$1.78 \cdot 10^{-3}$	$2.40 \cdot 10^{-5}$
2017b	$2.04 \cdot 10^{-4}$	$1.78 \cdot 10^{-3}$	$6.33 \cdot 10^{-5}$

Table 9: Results for the Hilbert 10×10 matrix

3 Conclusion and Advice

As mentioned at the beginning, nothing in this note comes as a surprise to experts in numerical analysis. Today's numerical algorithms are very stable, but no panacea, and severely wrong results are not avoidable.

Once again it is important to stress that we described general phenomena being true for any library, not only for Matlab. We cannot comment on the particular details of the different Matlab implementations due to confidentiality.

The following is advice to software developers and users. Two remedies would help without causing a major impact on computing time.

First, the growth factor could be made available. Then, at least it would become apparent that the problem described in Section 2.5 gave questionable results. The growth factor is available in libraries like LAPACK [1]; however, reportedly it is hardly used. Often a small $|U_{nn}|$ indicates problems, so further measures could be taken.

Second, one residual iteration in working precision would ensure backward stability. With this, the problem in Section 2.5 would be solved with high accuracy.

Again, that is, of course, known to experts. However, in the race of best computing time, safety is traded against speed.

Generally, wrong results are unavoidable, also without warning. Verification methods follow another strategy: no false answer is possible. If the problem is too difficult to be solved in the available working precision, a corresponding message is given, never a wrong result, in particular, in security sensitive areas that may be desirable.

4 Acknowledgement

The author wishes to thank Florian Bünger and Marko Lange for fruitful remarks. Moreover, we wish to thank an anonymous referee and the editor Baker Kearfott for constructive comments.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D.C. Sorensen. *LAPACK User's Guide, Release* 2.0. SIAM Publications, Philadelphia, second edition, 1995.

[2] L.V. Foster. Gaussian elimination with partial pivoting can fail in practice. *Siam J. Matrix Anal. Appl.*, 14:1354–1362, 1994.

[3] N. J. Higham. *Accuracy and stability of numerical algorithms*. SIAM Publications, Philadelphia, 2nd edition, 2002.

[4] MATLAB. User's Guide, Versions 2012a to 2017b, the MathWorks Inc., 2012-2017.

[5] M.R. Nakao and Y. Watanabe. *Self-validating Numerical Computations by Learning from Examples: Theory and Implementation*. The Library for Senior & Graduate Courses 85, 2011.

[6] T. Nishi, S.M. Rump, and S. Oishi. On the generation of very ill-conditioned integer matrices. *Nonlinear Theory and Its Applications, IEICE*, 2(2):226–245, 2011.

[7] S.M. Rump. Wie zuverlässig sind die Ergebnisse unserer Rechenanlagen? *Jahrbuch Überblicke Mathematik*, pages 163–168, 1983.

[8] S.M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. `http://www.ti3.tuhh.de/intlab`.

[9] R. Skeel. Iterative Refinement Implies Numerical Stability for Gaussian Elimination. *Math. Comp.*, 35(151):817–832, 1980.

[10] S.J. Wright. A collection of problems for which Gaussian elimination with partial pivoting is unstable. *SIAM J. Sci. Comput. (SISC)*, 14(1):231–238, 1993.