SLICOT Working Note 2010-1

How a numerical rank revealing instability affects Computer Aided Control System Design [1]

Zvonimir Bujanović[2] Zlatko Drmač[3]

January 2010

[1]Submitted to the ACM Transactions on Mathematical Software, December 2009. This work is supported by the Croatian Ministry of Science, Education and Sport under grant 0372783–2750 (*Spectral decompositions–numerical methods and applications*). A preliminary version of this report is available as Project Technical Report PTR–2009–2750–02, Department of Mathematics, University of Zagreb.
[2]Department of Mathematics, University of Zagreb, Croatia; Email: `zbujanov@math.hr`
[3]Department of Mathematics, University of Zagreb, Croatia; Email `drmac@math.hr`

Abstract

Since numerical libraries are used in engineering design in a variety of industrial applications, it is important that their numerical reliability is the top priority of both the developers of numerical algorithms and users from industry. Following that principle, we have examined a state of the art control library (case study: SLICOT) with respect to use of rank revealing subroutines in computing various canonical decompositions of linear time invariant systems. This issue seems to be critical, with potential for causing numerical catastrophes, because the deployed rank revealing code is prone to severe instabilities, causing completely wrongly computed parameters of systems under analysis. We analyze the SLICOT library in detail and propose modifications of critical parts of the code, based on our recent work published in the ACM Trans. Math. Softw. 35, 2008., where we analyze and solve the problem. The proposed modifications increase numerical reliability of all of the sixty affected subroutines. We recommend that the developers of other control theory numerical libraries examine their codes with respect to the issue discussed in this paper.

Keywords: numerical methods, software, MATLAB, matrix equations, system identification, model reduction.

1 Introduction

Numerical algorithms are at the core of modern CACSD (Computer Aided Control System Design) packages. Reliability, numerical accuracy and robustness of numerical software are top priorities of both the developers of the software and users from industry. The SLICOT (Subroutine Library In COntrol Theory) has been developed with those numerical aspects on the top of the list of requirements [5], [9]. SLICOT is used as computational layer in sophisticated CACSD packages such as EASY5 (since 2002. MSC.Software, initially developed in the Boeing Company), Matlab (The MathWorks) and Scilab (INRIA, Institut national de recherche en informatique et en automatique). Since its initial release, SLICOT has been growing at an impressive rate, from 90 user–callable subroutines in 1997., 200 subroutines in 2004. up to 470 subroutines in 2009. Another relevant library is the RASP, developed by the DLR (Deutsches Zentrum für Luft- und Raumfahrt e.V.), in close collaboration with the SLICOT development teams at NICONET WGS (Numerics In Control NETwork, Working Group on Software [13]) and NAG (Numerical Algorithms Groups [12]).

Parallel with widening the spectrum of control problems and enriching the functionality of the library, the numerical properties of all SLICOT subroutines remain under close watch by the numerical mathematics community. The same holds for all state of the art libraries of mathematical software, because modern developments in numerical mathematics and its applications in engineering and natural sciences increase the expectations from mathematical software in both efficiency and numerical quality of the output.

In this report, we focus on one particular issue – numerical rank and rank revealing software. Numerical rank revealing is one of the important tasks performed by SLICOT subroutines, very often in course of computation of various canonical representations of a LTI system

$$\begin{aligned} \mathsf{E}\dot{x} &= \mathsf{A}x + \mathsf{B}u \\ y &= \mathsf{C}x + \mathsf{D}u. \end{aligned} \tag{1}$$

Why are we interested in this? In [6], we described a subtle numerical instability in the LAPACK [1] rank revealing QR factorization software (the subroutines xGEQPF, xGEQP3, $\mathrm{x} \in \{\mathrm{S}, \mathrm{D}, \mathrm{C}, \mathrm{Z}\}$). The problem, that goes back to LINPACK [11] (subroutines xQRDC) and to the strategy of down–dating partial column norms that are required during the pivoting process, can cause severe failure of any computation based on pivoted QR factorization (e.g. least squares solvers). Our solution to the problem has been included in the LAPACK 3.1. release, and thus calling the new xGEQP3 and xGEQPF in other subroutines automatically resolves the issue. However, since LINPACK (in the 1970s and the 1980s) and LAPACK (since the 1990s) have been the source of inspiration and of model routines for matrix computations, the numerical bug has spread to many other libraries where, undetected, silently interferes with subtle decisions about the numerical rank.

The goal of this report is to raise a warning flag and to point to subroutines in the SLICOT library that are at high risk. More precisely, we have identified sixty of them (out of 470). The most effective and at the same time the most horrifying description of the problem is as follows:

Few strategically placed `"WRITE(*,*) variable"` *statements in the affected subroutines, requiring just written output of certain variables, can completely change the computed properties of (1). Other substantial variations of the output can be obtained by changing the compiler and optimizer options.*

So, for instance, one `" WRITE(*,*) variable"` statement can completely change the computed Kronecker indices of (1), or the computed zeros of the periodic descriptor system (See e.g. the algorithm in [15]). This is certainly undesired behavior, even if the computation is backward stable, and even if the computation is doomed to fail, due to ill–conditioning.

A tricky point here is that the problem occurs only at certain distance to singularity, and the rank revealing task itself is usually performed if we expect the matrix being close to singularity. And, since many things can happen close to singularity, any ill–behavior is usually attributed to ill–conditioning and the true problem can remain inconspicuous.

We show how to fix the problem using [6]. In this way, we contribute to better numerical reliability of SLICOT and all CACSD platforms that use SLICOT as computing engine. We also recommend to the developers of other libraries to examine their codes with respect to this issue.

2 The problem

Let us briefly describe the problem. The above mentioned LINPACK and LAPACK subroutines implement the Businger–Golub [4] pivoting which, for[1] $A \in \mathbb{R}^{m\times n}$, computes a permutation matrix P, an orthogonal Q, and an upper triangular matrix R such that

$$AP = Q\begin{pmatrix} R \\ 0 \end{pmatrix}, \quad \text{where } |R_{ii}| \geq \sqrt{\sum_{k=i}^{j} |R_{kj}|^2}, \quad \text{for all } 1 \leq i \leq j \leq n. \tag{2}$$

Note that (2) in particular implies $|R_{ii}| \geq |R_{i+1,i+1}|$ for $i = 1, \ldots, n-1$.

If the matrix A is close to a rank k matrix, and if the pivoting P performs well, then the matrix R can be block–partitioned as

$$R = \begin{pmatrix} R_{[11]} & R_{[12]} \\ 0 & R_{[22]} \end{pmatrix}, \quad R_{[11]} \in \mathbb{R}^{k\times k}, \tag{3}$$

where $\|R_{[22]}\|_F$ is small. Because of (2), the index k is revealed by a sharp drop on the diagonal of R, $|R_{kk}| \gg |R_{k+1,k+1}| \geq \|R_{[22]}\|_F/\sqrt{n-k}$.

The best way to understand the nature of the problem is to take a look at Figure 2 for a visual inspection of the structural properties (2) of the computed upper triangular matrix R. In short, one can construct an input matrix A such that the computed output violates the basic specification of the routine – the computed upper triangular matrix is not diagonally dominant, and the diagonal entries are not monotonically decreasing in modulus. Clearly, the problem illustrated in Figure 2 puts at high risk any computation that is based on the pivoted QR factorization and that assumes the structure (2) as granted. We discovered the problem during the development of a Jacobi type SVD algorithm [7], [8] where the structure of R was heavily used.

2.1 Down–dating the partial column norms

For the reader's convenience, we briefly describe one step of the factorization (2). Consider the k–th elimination step. Let $A^{(1)} = A = (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \mathbb{R}^{m\times n}$ and let

$$A^{(k)}\Pi_k = \begin{pmatrix} \cdot & \cdot & \odot & \cdot & \oplus & \cdot \\ & \cdot & \odot & \cdot & \oplus & \cdot \\ & & \blacksquare & \cdot & \circledast & \cdot \\ & & \circledcirc & \cdot & \ast & \cdot \\ & & \circledcirc & \cdot & \ast & \cdot \\ & & \circledcirc & \cdot & \ast & \cdot \end{pmatrix}, \quad \mathbf{a}_j^{(k)} = \begin{pmatrix} \oplus \\ \oplus \\ \circledast \\ \ast \\ \ast \\ \ast \end{pmatrix} \equiv \begin{pmatrix} \mathbf{x}_j^{(k)} \\ \eta_j^{(k)} \\ \mathbf{y}_j^{(k)} \end{pmatrix}, \quad \begin{array}{l} \eta_j^{(k)} = \circledast \equiv (A^{(k)})_{kj}, \\ \mathbf{z}_j^{(k)} = \begin{pmatrix} \eta_j^{(k)} \\ \mathbf{y}_j^{(k)} \end{pmatrix}. \end{array} \tag{4}$$

Elements to be annihilated are denoted by $\circledcirc$, and $\blacksquare$ denotes the element that will contain R_{kk}, computed in the k–th step, after the $\circledcirc$'s have been eliminated.

[1]For the sake of brevity, and without loss of generality, we consider only real $m \times n$ matrices with $m \geq n$.

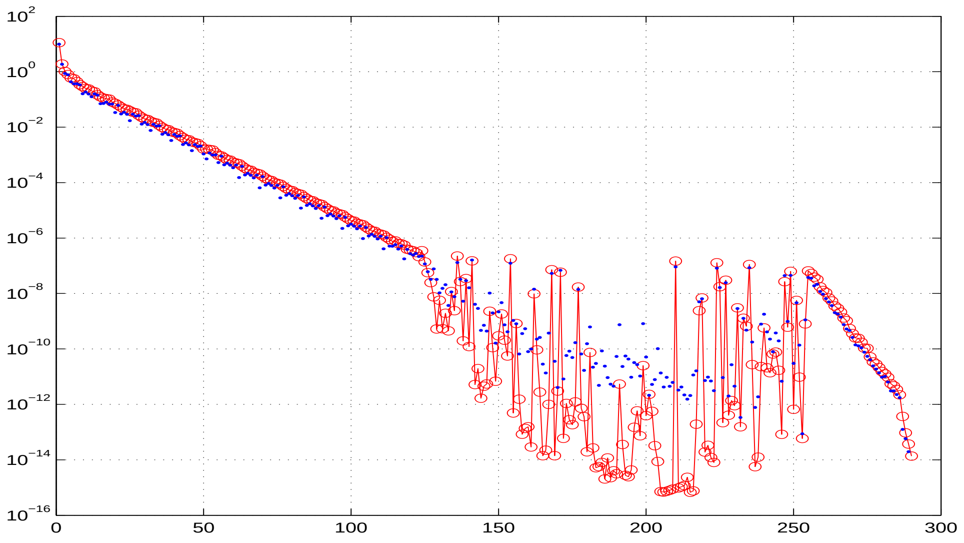

Figure 1: *Matlab 6.5. computation* `[Q,R,P]=qr(A)`*. The values of* $|R_{ii}|$ *(plotted as* `'-ro'`*) and* $\mu_i = \max_{j=i+1:n} |R_{ij}|$, $i = 1 : n-1$ *(plotted as* `'b.'`*) for a particular matrix* A*.*

Let $\omega_j^{(k)} = \|\mathbf{z}_j^{(k)}\|_2$. The permutation Π_k ensures that $|R_{kk}| \geq \omega_j^{(k)}$ for all $j \geq k$. Let H_k be Householder reflector such that

$$\mathrm{H}_k \begin{pmatrix} \eta_k^{(k)} \\ \mathbf{y}_k^{(k)} \end{pmatrix} = \begin{pmatrix} R_{kk} \\ 0 \end{pmatrix}, \text{ and let, for } j > k, \quad \begin{pmatrix} \beta_j^{(k+1)} \\ \mathbf{z}_j^{(k+1)} \end{pmatrix} = \mathrm{H}_k \mathbf{z}_j^{(k)}. \tag{5}$$

The goal is to compute $\omega_j^{(k+1)} = \|\mathbf{z}_j^{(k+1)}\|_2$ by a simple scalar formula with guaranteed and controlled number of correct digits whenever numerically feasible. The values of $\omega_j^{(k+1)}$, $j = k+1, \ldots, n$ will be used in the next step to determine the next pivotal column.

It clearly holds that $\omega_j^{(k)} = \sqrt{(\beta_j^{(k+1)})^2 + \|\mathbf{z}_j^{(k+1)}\|_2^2}$, and thus

$$\omega_j^{(k+1)} = \sqrt{(\omega_j^{(k)})^2 - (\beta_j^{(k+1)})^2} = \omega_j^{(k)} \sqrt{1 - \left(\frac{\beta_j^{(k+1)}}{\omega_j^{(k)}}\right)^2}. \tag{6}$$

Since initially $\omega_j^{(1)} = \|\mathbf{a}_j\|_2$, each $\omega_j^{(k+1)}$ can be recursively computed from $\omega_j^{(k)}$ and $\beta_j^{(k+1)}$, using (6).

Of course, severe cancelations – a consequence of ill–conditioning or, equivalently, closeness to singularity, can ruin the results of the down–dating formula. LINPACK and LAPACK subroutines have a safety device that has been installed to predict and avoid severe cancelations. Alas, one of the first victims of a nearby singularity is the estimated distance to singularity. As a result, the safety switch does not work properly. We refer the reader to [6] for detailed analysis and proposed solutions to this problem.

3 How this impacts SLICOT

SLICOT contains several subroutines that explicitly use the down–dating formula (6) for the partial column norms, and further use the computed norms to determine pivots. The list of those subroutines in the Release 5.0 of SLICOT contains nine items:

> AG08BY, AG8BYZ, MB02CU, MB03OY, MB03PY, MB3OYZ,
> MB3PYZ, MB04GD, TG01HX.

These subroutines are called by quite a few other routines, mainly for computing various canonical representations of LTI systems (1).

Remark 1 Determining numerical rank of a matrix is a delicate issue. Here we do not argue with the method for revealing the numerical rank (although we could), but rather with its implementation in numerical software. It should be mentioned here that in SLICOT the updating formula (6) has been implemented with a safety device as in LINPACK and in LAPACK prior to the 3.1. release.

3.1 Examples

We will use MB03OY as a prototype for problem description and for explaining our proposed solution. MB03OY is based on the LAPACK's subroutine DGEQPF, which, as shown in [6], suffers from the problem illustrated in Figure 2. In addition, MB03OY uses an incremental condition estimator (ICE), similarly as xGEQPX in ACM TOMS Algorithm 782 [3], [2]. We note that xGEQPX is also blacklisted in [6] as sensitive to the instability of the down–dating formula. In other words, the ICE cannot detect the problem.

Example 2 This example is entirely artificial, but it perfectly well illustrates the problem. Our matrix is a 100×100 matrix, obtained by symmetrization of the Kahan matrix, $0.5(\mathfrak{K}(n,c) + \mathfrak{K}(n,c)^T)$, where

$$\mathfrak{K}(1,c) = (1); \;\; \mathfrak{K}(n,c) = \left(\begin{array}{c|c} 1 & -c \; -c \; \ldots \; -c \\ \hline 0 & s\mathfrak{K}(n-1,c) \end{array}\right), \;\; c = \cos\psi, \;\; s = \sin\psi,$$

with a particular value of the cosine. Both the columns and the rows are nearly equilibrated in the sense that the ratio of the largest to the smallest column (row) norm is about eight. We set the parameter RCOND of MB03OY rather low, of the order of ε^2, where ε is the machine precision. This is not essential – the default value $n^2\varepsilon$ will also suffice. The only reason is that we want more columns to be included, to let the down–dating formula for the column norms run longer. We reiterate here that we are not testing the rank revealing property of the method, but its software implementation.

In this example, MB03OY computes the numerical rank as $r = 49$, and the neglected part of R (cf. (3)) has the norm

$$\|R_{[22]}\|_F/\|R\|_F \approx \|R(50:100,50:100)\|_F/(3.7\cdot 10^{17}) \approx 7.5\cdot 10^{-9}.$$

Now we repeat the computation under slightly changed conditions:

> In the routine MBO3OY we insert one `WRITE` statement. More precisely, " `WRITE(*,*) TEMP2`" is inserted after the line 339 of the SLICOT file `MB03OY.f`.

As a result, the computed numerical rank is now $r = 82$ and

$$\|R_{[22]}\|_F/\|R\|_F \approx \|R(83:100,83:100)\|_F/(3.7\cdot 10^{17}) \approx 6.9\cdot 10^{-34}.$$

The results of this two runs are shown on Figure 2. Note the substantial difference between the two results, and remember that the two routines differ in one WRITE statement. It is instructive to check the absolute values of the diagonal entries of $R_{[11]}$ and the column norms of the $R_{[22]}$ block in the partition

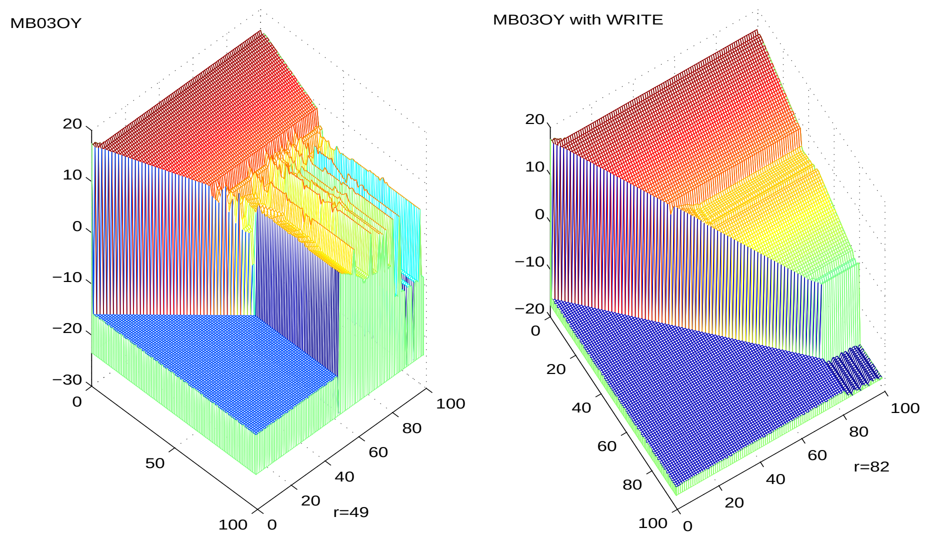

Figure 2: *Example 2: Left: The matrix R computed by MB03OY, shown by meshz(log10(abs(R))). The computed numerical rank is $r = 49$. Right: The matrix R computed with MB03OY, with* `"WRITE(*,*) TEMP2"` *statement added after the line 339 in MB03OY.f. The computed numerical rank is $r = 82$.*

(3). See Figure 3. Due to an undetected cancelation in the down–dating, small column appeared to the pivoting device much bigger than it actually was, and it was called in as pivotal. The ICE detected the drop on the diagonal, and declared numerical rank, tacitly assuming that the rest of the matrix is even smaller. A WRITE statement in a critical moment caused kicking the key variable from the processor's long register into memory (to be printed out) causing one rounding that was enough to change the rounding history and to change the computed numerical rank from $r = 49$ to $r = 82$.

Example 3 In this 90×90 example, we set the RCOND parameter to the realistic value of $n^2\varepsilon$, which is also used in SLICOT. The matrix is obtained from $\mathfrak{K}(n, c)$ by copying its strict upper triangle into its strict lower triangle, with opposite signs. The rank computed by MB03OY was $r = 58$, and the same subroutine with the WRITE statement added after the line 339 computed $r = 89$. See Figure 4. A severe underestimate of the numerical rank is caused by the unfortunate fact that, during the pivoted QR factorization, the smallest column in the sub–matrix appeared (according to the down–dated norms) as the dominant one and was thus declared pivotal.[2] The ICE detected sharp peak and terminated the process. Here $\kappa_2(R(1:58, 1:58)) \approx 4.9 \cdot 10^7$ and $\kappa_2(R(1:59, 1:59)) \approx 3.1 \cdot 10^{21}$. The remaining columns are considered small, but the neglected part $R_{[22]}$ of R (cf. (3)) in this case is $\approx 2.9 \cdot 10^{10}$. (Had the sharp increase in the condition number been, as observed by the ICE in MB03OY, just below $1/(n^2\varepsilon)$, the procedure would have taken the 59th column into the leading $(1, 1)$ block, and the computed numerical rank would have been at least 59, with unnecessarily ill–conditioned block $R_{[11]}$.) On the other hand, the modified routine (with the WRITE statement) has the neglected part of norm $\approx 3.4 \cdot 10^{-13}$. Also, in this case $\kappa_2(R(1:89, 1:89)) \approx 3.4 \cdot 10^{11}$.

Remark 4 The mere fact that a `WRITE` statement can change the computed numerical rank, and then e.g. the computed canonical structure of the system (1) is disturbing, and it certainly justifies the

[2]In fact, it can happen that null vector is taken as pivotal, while the remaining columns have huge norms!

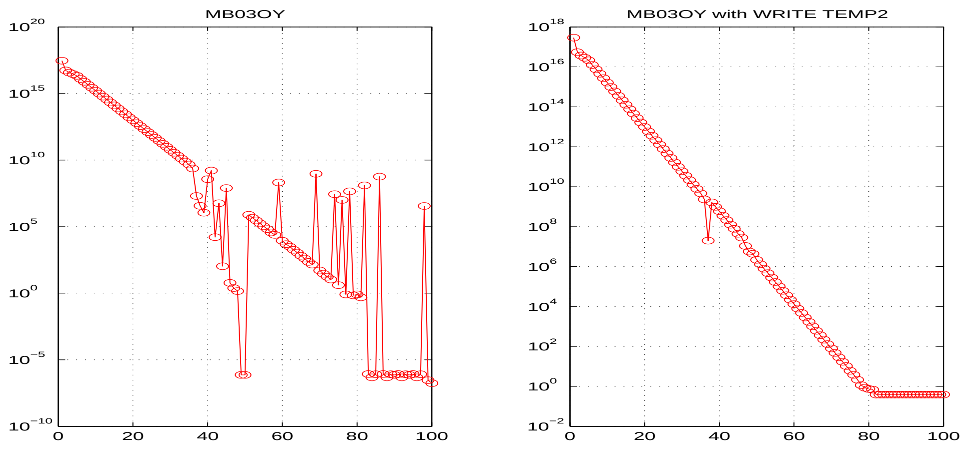

Figure 3: *Example 2: Left: The absolute values of the diagonal entries of $R_{[11]}$ and the column norms of the $R_{[22]}$ block in the matrix R computed by MB03OY. Right: The absolute values of the diagonal entries of $R_{[11]}$ and the column norms of the $R_{[22]}$ block in the matrix R computed with MB03OY, with* `"WRITE(*,*) TEMP2"` *statement added after the line 339 in MB03OY.f.*

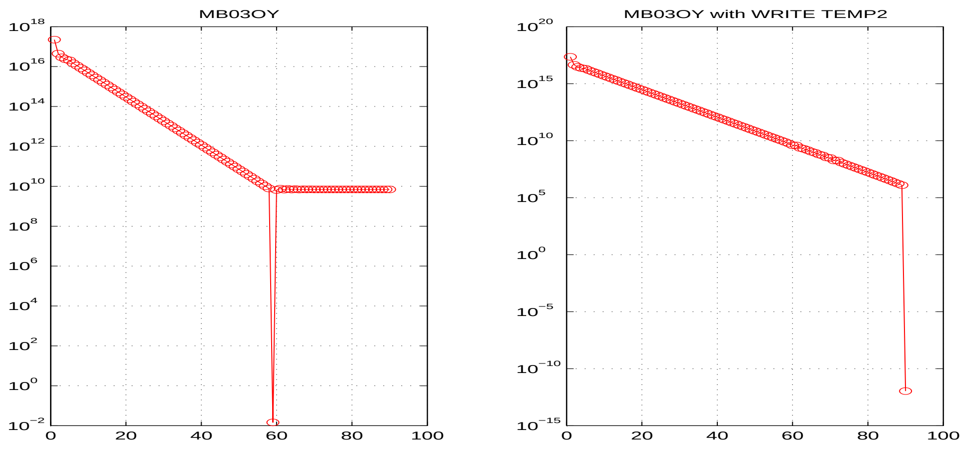

Figure 4: *Example 3: Left: The absolute values of the diagonal entries of $R_{[11]}$ and the column norms of the $R_{[22]}$ block in the matrix R computed by MB03OY. Right: The absolute values of the diagonal entries of $R_{[11]}$ and the column norms of the $R_{[22]}$ block in the matrix R computed with MB03OY, with* `"WRITE(*,*) TEMP2"` *statement added after the line 339 in MB03OY.f.*

effort to resolve the issue. In the next section we list sixty SLICOT routines[3] whose results can be manipulated by strategically placed `WRITE` statements, or changing compiler options. We underline here that this is not a programming bug, but a numerical instability.

Remark 5 Note that in the Examples 2, 3 we have not given the value of the cosines used to generate the matrices. These were $c = 0.8$ and $c = 0.653$, but if the reader tries to reproduce the experiments, it is very likely that MB03OY from his/her SLICOT library will work just fine, but some value $c + \delta c$ will exhibit the behavior illustrated in the Example. Changing the hardware, operating system, compiler, compiler and optimizer options can also substantially change the output. Again, we refer the reader to [6] for more details.

Remark 6 The erratic behavior illustrated in this report is not restricted to rare artificially constructed examples. If A is $m \times n$ of full column rank (in rank deficient case consider the maximal subset if linearly independent columns) and A_c is obtained from A by scaling its columns to have unit Euclidean lengths, then an analysis shows that $\|A_c^\dagger\|_2 > 1/\sqrt{\varepsilon}$ indicates that problems with the down–dating formula may appear and that miss–pivoting may occur. (Here $\|A_c^\dagger\|_2$ is the spectral norm of the Moore–Penrose generalized inverse of A_c, and ε is the machine rounding.) Note that $\|A_c^\dagger\|_2 > 1/\sqrt{\varepsilon}$ holds if e.g. $\min_{D=\mathrm{diag}} \kappa_2(AD) > \frac{\sqrt{n}}{\sqrt{\varepsilon}}$.

Example 7 Just to illustrate how the problem described in the previous two examples spreads throughout the library, consider the SLICOT description of the subroutines TB01UD:

TB01UD computes a controllable realization for the linear time–invariant multi–input system

$$\begin{aligned} \dot{x} &= Ax + Bu, \\ y &= Cx, \end{aligned}$$

where A, B, and C are $n \times n$, $n \times m$, and $p \times n$ matrices, respectively. The matrices A and B are reduced by this routine to orthogonal canonical form using (and optionally accumulating) orthogonal similarity transformations, which are also applied to C. Specifically, the system (A, B, C) is reduced to the triplet (A_c, B_c, C_c), where $A_c = Z^T A Z$, $B_c = Z^T B$, $C_c = CZ$, with

$$A_c = \begin{pmatrix} \hat{A}_c & \star \\ 0 & \hat{A}_u \end{pmatrix}, \quad B_c = \begin{pmatrix} \hat{B}_c \\ 0 \end{pmatrix},$$

and

$$\hat{A}_c = \begin{pmatrix} A_{11} & A_{12} & A_{13} & \cdots & \cdots & A_{1,p-1} & A_{1p} \\ A_{21} & A_{22} & A_{23} & \cdots & \cdots & A_{2,p-1} & A_{2p} \\ 0 & A_{32} & A_{33} & \cdots & \cdots & A_{3,p-1} & A_{3p} \\ \vdots & 0 & \ddots & \ddots & & \vdots & \vdots \\ \vdots & . & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & . & . & & A_{p-1,p-2} & A_{p-1,p-1} & A_{p-1,p} \\ 0 & \cdots & 0 & \cdots & 0 & A_{p,p-1} & A_{pp} \end{pmatrix}, \hat{B}_c = \begin{pmatrix} B_1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

where the sub–matrices $B_1, A_{21}, \ldots, A_{p,p-1}$ have full row ranks (all computed using MB03OY !) and p is the controllability index of the pair. The size of the block $\hat{A}_u$ is equal to the dimension of the uncontrollable subspace of the pair (A, B).

Now, TB01UD is called by TB01PD, and then TB01PD is called by TD04AD, which is called by SB10ZP, and SB10ZP is called by SB10YD, which is finally called by SB10MD, and SB10MD performs the D-step of the D-K iterations. Thus, a robust controller design might be influenced by `WRITE` statements inserted at certain places in the source code. We suspect that SLICOT is not the only control library that might exhibit this behavior.

[3]The release 5 of SLICOT contains 470 routines.

3.2 Affected subroutines

By examining the source code of SLICOT (470 subroutines), we have identified altogether sixty routines affected by the problem illustrated in the previous section.[4] For the sake of brevity, we will only list the routine names and indicate dependencies. For detailed descriptions of the computational tasks performed by these subroutines the reader is referred to [14].

In the following scheme "$\twoheadleftarrow$" is read as "is called by the subroutine", and "$\twoheadleftarrow$:" is understood as "is called by the following subroutines:".

1. $\boxed{\text{MB03OY}}$ $\twoheadleftarrow$:

- **1.1** $\boxed{\text{AB01ND}}$ $\twoheadleftarrow$ **1.1.1** $\boxed{\text{AB01OD}}$
- **1.2** $\boxed{\text{AB08NX}}$ $\twoheadleftarrow$:
  - **1.2.1** $\boxed{\text{AB08ND}}$
  - **1.2.2** $\boxed{\text{AB08MD}}$ $\twoheadleftarrow$ **1.2.2.1** $\boxed{\text{AB09JD}}$
- **1.3** $\boxed{\text{AG08BY}}$ $\twoheadleftarrow$ **1.3.1** $\boxed{\text{AG08BD}}$
- **1.4** $\boxed{\text{MB02QD}}$ $\twoheadleftarrow$ **1.4.1** $\boxed{\text{SB01DD}}$
- **1.5** $\boxed{\text{TB01UD}}$ $\twoheadleftarrow$:
  - **1.5.1** $\boxed{\text{TB01PD}}$ $\twoheadleftarrow$:
    - **1.5.1.1** $\boxed{\text{TD04AD}}$ $\twoheadleftarrow$ **1.5.1.1.1** $\boxed{\text{SB10ZP}}$ $\twoheadleftarrow$ **1.5.1.1.1.1** $\boxed{\text{SB10YD}}$ $\twoheadleftarrow$ **1.5.1.1.1.1.1** $\boxed{\text{SB10MD}}$
    - **1.5.1.2** $\boxed{\text{AB09ID}}$
  - **1.5.2** $\boxed{\text{TB03AD}}$ $\twoheadleftarrow$ **1.5.2.1** $\boxed{\text{TD03AD}}$
  - **1.5.3** $\boxed{\text{TB04AY}}$ $\twoheadleftarrow$ **1.5.3.1** $\boxed{\text{TB04AD}}$
- **1.6** $\boxed{\text{TG01FD}}$ $\twoheadleftarrow$ **1.6.1** $\boxed{\text{AG08BD}}$

2. $\boxed{\text{MB04GD}}$ $\twoheadleftarrow$ **2.1** $\boxed{\text{MB03PD}}$

3. $\boxed{\text{MB03OD}}$ $\twoheadleftarrow$:

- **3.1** $\boxed{\text{IB01ND}}$ $\twoheadleftarrow$ **3.1.1** $\boxed{\text{IB01AD}}$ $\twoheadleftarrow$:
  - **3.1.1.1** $\boxed{\text{IB03AD}}$
  - **3.1.1.2** $\boxed{\text{IB03BD}}$
- **3.2** $\boxed{\text{IB01PD}}$ $\twoheadleftarrow$ **3.2.1** $\boxed{\text{IB01BD}}$ $\twoheadleftarrow$:
  - **3.2.1.1** $\boxed{\text{IB03AD}}$
  - **3.2.1.2** $\boxed{\text{IB03BD}}$
- **3.3** $\boxed{\text{IB01PY}}$ $\twoheadleftarrow$ **3.3.1** $\boxed{\text{IB01PD}}$ $\twoheadleftarrow$ **3.3.1.1** $\boxed{\text{IB01BD}}$ $\twoheadleftarrow$:
  - **3.3.1.1.1** $\boxed{\text{IB03AD}}$
  - **3.3.1.1.2** $\boxed{\text{IB03BD}}$
- **3.4** $\boxed{\text{MB02QD}}$ $\twoheadleftarrow$ **3.4.1** $\boxed{\text{SB01DD}}$
- **3.5** *$\boxed{\text{MB02YD}}$ $\twoheadleftarrow$:
  - **3.5.1** $\boxed{\text{NF01BQ}}$ $\twoheadleftarrow$ **3.5.1.1** $\boxed{\text{NF01BP}}$
  - **3.5.2** $\boxed{\text{MD03BY}}$ $\twoheadleftarrow$:
    - **3.5.2.1** $\boxed{\text{MD03BB}}$
    - **3.5.2.2** $\boxed{\text{NF01BP}}$

[4]Here we do not claim that we have found all occurrences of the problem.

3.6 $*\boxed{\text{MD03BY}} \twoheadleftarrow$:
3.6.1 $\boxed{\text{MD03BB}}$
3.6.2 $\boxed{\text{NF01BP}}$

3.7 $*\boxed{\text{NF01BR}} \twoheadleftarrow$:
3.7.1 $\boxed{\text{NF01BP}}$
3.7.2 $\boxed{\text{NF01BQ}} \twoheadleftarrow$ **3.7.2.1** $\boxed{\text{NF01BP}}$

4. $\boxed{\text{MB3OYZ}} \twoheadleftarrow$:

4.1 $\boxed{\text{AB8NXZ}} \twoheadleftarrow$:
4.1.1 $\boxed{\text{AB08MZ}}$
4.1.2 $\boxed{\text{AB08NZ}}$

4.2 $\boxed{\text{AG8BYZ}} \twoheadleftarrow \boxed{\text{AG08BZ}}$

4.3 $\boxed{\text{TG01FZ}} \twoheadleftarrow \boxed{\text{AG08BZ}}$

5. $\boxed{\text{MB03PY}} \twoheadleftarrow$ **5.1** $\boxed{\text{AB08NX}} \twoheadleftarrow$:
5.1.1 $\boxed{\text{AB08MD}} \twoheadleftarrow$ **5.1.1.1** $\boxed{\text{AB09JD}}$
5.1.2 $\boxed{\text{AB08ND}}$

6. $\boxed{\text{MB3PYZ}} \twoheadleftarrow$ **6.1** $\boxed{\text{AB8NXZ}} \twoheadleftarrow$:
6.1.1 $\boxed{\text{AB08MZ}}$
6.1.2 $\boxed{\text{AB08NZ}}$

7. $\boxed{\text{MB02CU}} \twoheadleftarrow$:

7.1. $\boxed{\text{MB02GD}}$; **7.2.** $\boxed{\text{MB02HD}}$; **7.3.** $\boxed{\text{MB02ID}}$

7.4. $\boxed{\text{MB02JD}}$; **7.5.** $\boxed{\text{MB02JX}}$

8. $\boxed{\text{AG08BY}} \twoheadleftarrow$ **8.1** $\boxed{\text{AG08BD}}$

9. $\boxed{\text{AG8BYZ}} \twoheadleftarrow$ **9.1** $\boxed{\text{AG08BZ}}$

10. $\boxed{\text{TG01HX}} \twoheadleftarrow$:

10.1 $\boxed{\text{TG01HD}}$; **10.2** $\boxed{\text{TG01ID}}$; **10.3** $\boxed{\text{TG01JD}}$

Remark 8 The subroutines marked with a $*$ (**3.5**, **3.6**, **3.7**) may optionally require already computed upper triangular factor.

Remark 9 Given the fact that LINPACK (since 1970s), LAPACK (since 1992.), SLICOT (since 1997.) have been used as computing engines by several platforms in broad spectrum of industrial applications, and that the problem with the rank revealing software has been detected and fixed in 2006. (only in LAPACK) may give a wrong impression that this problem is not that important. It would also be wrong to dismiss the problem with an argument that no useful results can be expected in the case of ill–conditioning, and to consider backward stability as a mitigating circumstance. We note that the problem may occur at the condition number $1/\sqrt{\varepsilon}$, which means that only half (roughly) of the accuracy is lost due to the ill–conditioning. In fact, it should be clear that, from the mathematical, software engineering, and also applications' points of view, the issue is relevant and serious. It is not hard to imagine how this problem may impact a mission critical design using SLICOT for large scale dense computation on a distributed (say, heterogenous) parallel machine.

4 Backward compatible solution

We now describe our proposed modifications. One of the constraints in resolving this issue is backward compatibility. In other words, the problem should be removed without changing the specifications of the affected routines. This is a quick fix solution. A proper but not backward compatible solution also exists and it can be obtained from the authors.

4.1 Modifications through LAPACK (MB03OD)

MB03OD computes the pivoted QR factorization by a direct call to the LAPACK's subroutine DGEQP3, and problems related to MB03OD are automatically solved if SLICOT is linked with LAPACK 3.1 or later. (LAPACK 3.1 contains backward compatible resolution of the problem, see [6], [10].)

4.2 Modified source

We have identified nine subroutines that need changes in their source codes.

4.2.1 MB03OY

The critical part of the code of MB03OY starts at the line 330:

```
C C     Update partial column norms. C
      DO 30 J = I + 1, N
         IF( DWORK(J).NE.ZERO ) THEN
             TEMP = ONE-(ABS(A(I,J))/DWORK(J))**2
             TEMP = MAX(TEMP,ZERO)
             TEMP2 = ONE+P05*TEMP*(DWORK(J)/DWORK(N+J))**2
             IF( TEMP2.EQ.ONE ) THEN
                 IF( M-I.GT.0 ) THEN
                     DWORK(J) = DNRM2(M-I,A(I+1,J),1)
                     DWORK(N+J) = DWORK(J)
                 ELSE
                     DWORK(J)   = ZERO
                     DWORK(N+J) = ZERO
                 END IF
             ELSE
                 DWORK(J) = DWORK(J)*SQRT(TEMP)
             END IF
         END IF
 30   CONTINUE
```

Here `P05` is a parameter with value 0.05, defined in the lines 187. and 188.

The proposed modification, based on [6], reads:

```
      TOL3Z = DSQRT(DLAMCH('Epsilon'))
C C     Update partial column norms. C
      DO 30 J = I + 1, N
```

```
         IF( DWORK(J).NE.ZERO ) THEN
             TEMP = ABS(A(I,J))/DWORK(J)
             TEMP = MAX(ZERO,(ONE+TEMP)*(ONE-TEMP))
             TEMP2 = TEMP*(DWORK(J)/DWORK(N+J))**2
             IF( TEMP2.LE.TOL3Z ) THEN
                 IF( M-I.GT.0 ) THEN
                     DWORK(J) = DNRM2(M-I,A(I+1,J),1)
                     DWORK(N+J) = DWORK(J)
                 ELSE
                     DWORK(J)   = ZERO
                     DWORK(N+J) = ZERO
                 END IF
             ELSE
                 DWORK(J) = DWORK(J)*SQRT(TEMP)
             END IF
         END IF
   30   CONTINUE
```

A few remarks are in order:

- The new routine requires the machine precision parameter ε, here computed using the LAPACK function DLAMCH.
- We were not able to prove that the above modification is safe. However, the analysis that we were able to conduct indicates that it should work well, but rather tedious estimates are inconclusive. Further, numerical experiments have shown that the new code is hard to break. However, if the threshold TOL3Z is lowered even only slightly below $\sqrt{\varepsilon}$, examples of ill–behavior can be found. For more details, see [6].
- We have a more sophisticated and provably safe modification that requires extra `N` scalar locations of workspace. More details can be obtained from the authors.
- The new code is more conservative with respect to the use of the scalar formula for updating partial column norms. However, numerical experiments with well–behaved examples show no or only negligible penalty in terms of efficiency (runtime). This is in particular true for a more elaborate modification (not described here, see [6]). Increased runtime is to be expected only in cases where more explicit norm computations are necessary – in those cases the difference between the computed results is substantial. On the other hand, the rank revealing QR factorization is usually just a part of a more complex algorithm, and this increased runtime is negligible, especially in light of more sound numerical properties.

4.2.2 Other subroutines

The proposed change for MB03OY can be used as a template for modifications of the remaining eight subroutines. We skip the details for the sake of brevity. The source codes of the modified routines can be obtained from the authors.

4.3 Concluding remarks

We classify this problem a serious threat and strongly suggest modifications of SLICOT and other libraries that use the LINPACK's down–dating formula (6). The modifications discussed here, and also included in [10], are quick fixes that probably (but not provably) resolve the issue, and in a backward compatible way. A proper, more sophisticated fix is available (from the authors) and it is our hope that it will be adopted in future releases of state of the art numerical libraries such as LAPACK and SLICOT.

In general, this example shows how a numerical ill–conditioning of a fairly simple formula can surreptitiously influence the results of rather sophisticated computations based on several state of the art packages of mathematical software. This stresses the importance of extending the numerical analysis of an algorithm to the executable code, as well as more detailed testing.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenny, S. Ostrouchov, and D. Sorensen. *LAPACK users' guide, second edition.* SIAM, Philadelphia, PA, 1992.

[2] C. H. Bischof and G. Quintana-Orti. Algorithm 782: codes for rank–revealing QR factorizations of dense matrices. *ACM Transactions on Mathematical Software*, 24(2):254–257, 1998.

[3] C. H. Bischof and G. Quintana-Orti. Computing rank–revealing QR factorizations of dense matrices. *ACM Transactions on Mathematical Software*, 24(2):226–253, 1998.

[4] P. A. Businger and G. H. Golub. Linear least squares solutions by Householder transformations. 7:269–276, 1965.

[5] P. Van Dooren. The basics of developing numerical algorithms. *IEEE Control Systems Magazine*, 24(1):18–27, 2004.

[6] Z. Drmač and Z. Bujanović. On the failure of rank revealing QR factorization software – a case study. *ACM Trans. Math. Softw.*, 35(2):1–28, 2008.

[7] Z. Drmač and K. Veselić. New fast and accurate Jacobi SVD algorithm: I. *SIAM J. Matrix Anal. Appl.*, 29(4):1322–1342, 2008.

[8] Z. Drmač and K. Veselić. New fast and accurate Jacobi SVD algorithm: II. *SIAM J. Matrix Anal. Appl.*, 29(4):1343–1362, 2008.

[9] S. Van Huffel, V. Sima, A. Varga, S. Hammarling, and F. Delebecque. High–performance numerical software for control. *IEEE Control Systems Magazine*, 24(1):60–76, 2004.

[10] LAPACK 3.1. http://netlib2.cs.utk.edu/lapack/lapack-3.1.0.changes, 2006.

[11] LINPACK. http://www.netlib.org/linpack/, 2009.

[12] NAG. http://www.nag.co.uk/, 2009.

[13] NICONET. http://www.icm.tu-bs.de/niconet/, 2009.

[14] SLICOT. http://www.slicot.org/, 2009.

[15] A. Varga and P. Van Dooren. Computing the zeros of periodic descriptor systems. *Systems Control Lett.*, 50(5):371–381, 2003.