Verified Inclusion of a Basis of the Null space*†

R. Kobayashi
Faculty of Science and Engineering, Waseda University, 3–4–1 Okubo, Shinjuku–ku, Tokyo 169–8555, Japan
r.kobayashi@thu.ac.jp

M. Lange
Institute for Reliable Computing, Hamburg University of Technology, Am Schwarzenberg Campus 3, Hamburg 21071, Germany
m.lange@tuhh.de

A. Minamihata‡
Department of Information and System Engineering, Chuo University, 1–13–27 Kasuga, Bunkyo–ku, Tokyo 112–8551, Japan
minamihata.71e@g.chuo-u.ac.jp

S.M. Rump
Institute for Reliable Computing, Hamburg University of Technology, Am Schwarzenberg Campus 3, 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

We describe methods to compute a verified inclusion of a basis of the null space of a rectangular, real or complex matrix. The quality of the inclusion correlates with the ratio of the largest and smallest singular value of the input matrix. Some executable INTLAB routines are given.

Keywords: Null space, orthogonal basis, verified inclusion, INTLAB
AMS subject classifications: 65G40, 15A06

*This research was partially supported by CREST, Japan Science and Technology Agency.
†Submitted: October 12, 2019; Revised: May 20, 2020; Accepted: July 23, 2020.
‡This work was supported by JSPS KAKENHI Grant Number JP17K12692

1 Notation and previous work

Let a rectangular matrix $A \in \boldsymbol{K}^{m\times n}$ with $m < n$ and $\boldsymbol{K} \in \{\boldsymbol{R}, \boldsymbol{C}\}$ be given. We aim to compute an inclusion of a basis of the null space of A.

We assume some familiarity by the reader with interval arithmetic [5, 8] and verification methods [10, 14]. Interval quantities are in bold-face, and an interval matrix $\mathbf{A}$ is said to have full rank if all $A \in \mathbf{A}$ have that property.

If the null space of A has dimension k, then the right singular vectors to the k smallest singular values of A form an orthonormal basis of null(A). One obvious way to obtain an inclusion of such a basis is to compute a verified inclusion of the respective singular vectors.

If A is rank deficient, then an ε-perturbation may change its rank and therefore the dimension of null(A). Hence the problem becomes ill-posed. As verification methods are restricted to well-posed problems [14], a verified inclusion of the null space is, except for exotic cases, only possible if A has full rank.

In [11], Rohn presented a verification method to compute an inclusion of a matrix, the range of which spans the null space of A. His method is based on a verified inclusion $\mathbf{B}$ of the Moore-Penrose pseudoinverse $A^\dagger$ of A. Then $\mathbf{N} = I - \mathbf{B}A$ contains a matrix whose range is identical to the null space of A. In order to compute $\mathbf{B}$, Rohn considers three methods based on $A^\dagger = (A^*A)^{-1}A^*$, on Greville's algorithm [3], and on a verified inclusion of a singular value decomposition of A, respectively.

In the following we want to improve on this by giving another four different methods for computing an inclusion of a basis of the null space of A.

2 Main result

Let a matrix $A \in \boldsymbol{K}^{m\times n}$ with $m < n$ be given, and let $A^* = QR$ be a QR-decomposition of the Hermitian of A. Denote by $0_{p,q}$ and $I_{p,q}$ the $p \times q$ zero and identity matrix, respectively. We consider the block structure

$$A^* = QR = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_1 \\ 0_{n-m,m} \end{pmatrix},$$

where $Q_1 \in \boldsymbol{K}^{n\times m}$, $Q_2 \in \boldsymbol{K}^{n\times(n-m)}$, and $R_1 \in \boldsymbol{K}^{m\times m}$. Then $A^* = Q_1R_1$, and if A has full rank, then Q_2 is an orthogonal basis of the null space of A.

2.1 Enclosure for underdetermined linear systems

An approximate QR-decomposition of A^* produces a numerical basis $\tilde{Q}_2$ for the null space of A. Suppose that A has full rank and that E is a solution to the underdetermined linear system $AE = A\tilde{Q}_2$. Define $X := \tilde{Q}_2 - E$. Then $AX = 0$ and X is a basis of the null space of A provided X has also full rank. The latter is implied by $\|E\|_2 \leq \sigma_{\min}(\tilde{Q}_2)$ which in turn can be verified by checking that $\|E\|_2^2 \leq 1 - \|\tilde{Q}_2^T\tilde{Q}_2 - I\|_2$. To reduce the computational effort, we use the Frobenius norm as an upper bound for the spectral norm. Using

INTLAB [12], the Matlab/Octave toolbox for Reliable Computing, a code to compute an inclusion $\mathbf{X}$ of X can be as follows:

```
[Q,~] = qr(mid(A)');
Q2 = Q(:,m+1:n);
B = A*intval(Q2);
E = verifylss(A,B);
X = Q2 - E;
success = ( norm(E,'fro').^2 < 1 - ...
     norm(Q2'*intval(Q2)-eye(n-m),'fro') );
```

Method 1: Solving an underdetermined linear system.

Here the function `verifylss` returns an enclosure for solutions of underdetermined systems or least squares problems (see [15]). A basis of the null space of A is included in $\mathbf{X}$ if the latter has full rank, which is verified by $\|E\|_F^2 < 1 - \|\tilde{Q}_2^T \tilde{Q}_2 - I\|_F$. That final verification step is necessary because, in principle, $\tilde{Q}_2 - E$ could be rank deficient.

It is noteworthy that the code is applicable for real or complex interval matrices $\mathbf{A}$ as well. In that case for every $\tilde{A} \in \mathbf{A}$ there exists some $X \in \mathbf{X}$ such that X is a basis of the null space of $\tilde{A}$.

2.2 Square linear system embedding

For a null space of small dimension we next exploit the extended abilities of INTLAB to solve square linear systems. Let $A \in \mathbb{K}^{m\times n}$, $\alpha \in \boldsymbol{K}$ and $B \in \boldsymbol{K}^{(n-m)\times n}$ be such that the square matrix

$$M_\alpha := \begin{pmatrix} A \\ \alpha B \end{pmatrix}$$

is nonsingular. This is not satisfiable if A or B are rank deficient, or $\alpha = 0$. A solution $X \in \boldsymbol{K}^{n\times(n-m)}$ to the square linear system

$$\begin{pmatrix} A \\ \alpha B \end{pmatrix} X = \begin{pmatrix} 0_{m,n-m} \\ \alpha I_{n-m,n-m} \end{pmatrix} \tag{1}$$

specifies a basis of the null space of A. Thus, to compute a basis of null(A), we need to find appropriate α and B.

If we set $B := Q_2^*$, then the solution to (1) is $X = Q_2$, an orthogonal basis of null(A). In practice we have only an approximate $\tilde{Q}_2$ for this solution and set $B = \tilde{Q}_2$. The corresponding linear system in (1) can be tackled by some verification method for square linear systems [7, 8, 15, 18]. The successful completion of any of these verification methods also proves that the system matrix M_α is regular, which in turn implies that A as well as X have full rank.

The singular values of M_α with $B := Q_2^*$ are $\sigma_i(A)$ and an $(n-m)$-fold singular value α. Thus, choosing α within the interval $[\sigma_m(A), \sigma_1(A)]$ ensures that the condition numbers, the ratio of the largest and smallest singular value,

of A and M_α are the same. The described approach can be realized, for instance, via the following INTLAB code:

```
[Q,~] = qr(mid(A)');
alpha = max(vecnorm(A,2,2));
Ma = [A; alpha*Q(:,m+1:n)'];
b = [zeros(m,n-m); alpha*eye(n-m)];
X = verifylss(Ma,b);
```

Method 2: Solving an extended square linear system.

Here `vecnorm(A,2,2)` computes the vector of Euclidean lengths of the rows of A. As already mentioned above, the successful application of `verifylss` not only yields a verified interval enclosure for an actual basis of null(A) but also implies that M_α and A have full rank.

For extremely ill-conditioned problems, `X = verifylss(Ma,b,'illco')` may be used [13]. The general idea behind this method is similar to the preconditioning approach that we propose in Subsection 3.1.

To save computing time, one may exploit the already computed (approximate) QR-decomposition together with the identity

$$M_\alpha \begin{pmatrix} Q_1(R_1^*)^{-1} & \alpha^{-1}Q_2 \end{pmatrix} = \begin{pmatrix} R_1^*Q_1^* \\ \alpha Q_2^* \end{pmatrix} \begin{pmatrix} Q_1(R_1^*)^{-1} & \alpha^{-1}Q_2 \end{pmatrix} = I_n.$$

Thus, an approximate inverse of M_α can be obtained at the cost of solving numerically a linear system with matrix R_1 and right-hand side Q_1^*. Then classical inclusion methods such as the Kahan-Krawczyk operator [4, 6], Yamamoto's approach [18], or methods based on H-matrices [7, 15] can be used.

2.3 Square linear system reduction

In contrast to the embedding discussed in the previous subsection, our third method is based on the reduction of the underdetermined system $AX = 0$ to a smaller square linear system. The benefit of this approach is that it does not require a QR-decomposition. The first step is to choose a permutation matrix P so that $AP = \begin{pmatrix} A_1 & A_2 \end{pmatrix}$ with nonsingular $A_1 \in \boldsymbol{K}^{m\times m}$. That is possible for any A that has full rank. The standard choice is based on an approximate LU-decomposition of A^*. To be precise, let

$$P^*A^* = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} U$$

be an LU-decomposition with partial pivoting, such that, provided A has full rank, $L_1 \in \boldsymbol{K}^{m\times m}$ and $U \in \boldsymbol{K}^{m\times m}$ are regular triangular matrices. Then, using the same P, the first m columns of AP are linearly independent and $A_1 = U^*L_1^*$ is regular.

We consider a similar permuted block structure of a solution $X \in \boldsymbol{K}^{n\times(n-m)}$ to $AX = 0$, by which

$$A \cdot X = \underbrace{\begin{pmatrix} A_1 & A_2 \end{pmatrix} P^*}_{=:A} \cdot \underbrace{P \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}}_{=:X} = A_1X_1 + A_2X_2 = 0.$$

The computation of X can therefore be reduced to choosing an appropriate $X_2 \in \boldsymbol{K}^{(n-m)\times(n-m)}$ and computing the corresponding $X_1 \in \boldsymbol{K}^{m\times(n-m)}$ by solving $A_1X_1 = -A_2X_2$. A natural choice for X_2 is $I_{n-m,n-m}$. This ensures full rank of X, simplifies the overall computation and leads to the following desired basis of the null space of A:

$$X = P \begin{pmatrix} -A_1^{-1}A_2 \\ I_{n-m,n-m} \end{pmatrix}.$$

This particular choice is often called *the fundamental basis* [1, 17].

Once more we can utilize a standard verification method for square linear systems to compute a verified enclosure for the matrix X given above. In INTLAB the code can be as follows:

```
[~,~,p] = lu(mid(A)',’vector’);
X = [verifylss(A(:,p(1:m)),-A(:,p(m+1:n))); eye(n-m)];
X = X(invperm(p),:);
```

Method 3: Solving a reduced square linear system.

Here the function `invperm` returns the inverse permutation. Note that the only purpose of the LU-decomposition in the first line is to compute an appropriate permutation vector p.

The computational effort for the third method is smaller than for the previous two methods as the size of the system matrix is $m \times m$ rather than $m \times n$ or $n \times n$, respectively. However, the computed basis is typically not close to orthogonality.

It is noteworthy that Method 3 can be interpreted as a special case of the method based on the linear system (1). If we assume that X_2 is regular, then for $B = \begin{pmatrix} 0_{n-m,m} & X_2^{-1} \end{pmatrix} P^*$, the linear system in (1) yields

$$M_\alpha \cdot X = \begin{pmatrix} A_1 & A_2 \\ 0_{n-m,m} & \alpha X_2^{-1} \end{pmatrix} P^* \cdot X = \begin{pmatrix} A_1X_1 + A_2X_2 \\ \alpha X_2^{-1}X_2 \end{pmatrix} = \begin{pmatrix} 0_{m,n-m} \\ \alpha I_{n-m,n-m} \end{pmatrix}.$$

The particular choice of B motivates a reduction of (1) to the linear system discussed in this subsection.

Naturally, a similar reduction is applicable for the choice $B := Q_2^+ = Q_2^*$. By applying a similar partitioning as above to Q_2, we derive $Q_2^*P = \begin{pmatrix} Q_{21} & Q_{22} \end{pmatrix}$, where $Q_{12} \in \boldsymbol{K}^{n-m\times m}$ and $Q_{22} \in \boldsymbol{K}^{n-m,n-m}$. If we now set $X_2 := Q_{22}$, then $X_1 = Q_{21}$ is the solution to the system $A_1X_1 = -A_2X_2$, which yields the desired orthogonal representation of the null space of A. The corresponding INTLAB code could be as follows:

```
[~,~,p] = lu(mid(A)','vector');
[A1,A2] = deal(A(:,p(1:m)),A(:,p(m+1:n)));
[Q,~] = qr(mid(A)');
Q22 = Q(p(m+1:n),m+1:n);
B1 = A2*intval(-Q22);
X = [verifylss(A1,B1); Q22];
X = X(invperm(p),:);
success = ( norm(mid(X)'*X-eye(n-m),inf) < 1 );
```

Method 4: Solving a reduced square linear system.

In the ideal case, Methods 2 and 4 produce enclosures for the same solution $X = Q_2$. In practice we use an approximation $\tilde{Q}_2$. Then Method 4 computes an inclusion for a solution X that is rather comparable with a solution to (1) with $B := \tilde{Q}_2^+$ instead of $B := \tilde{Q}_2^*$.

The additional cost for this nearly orthogonal solution is the QR-decomposition of A and the necessity to check non-singularity of X because $\tilde{Q}_{22}$ may be rank-deficient.

2.4 Rohn's Method

As an alternative to the previously proposed methods and to have comparative data for our numerical tests, we consider Rohn's verification approach [11]. To be specific, we consider Rohn's `vernull` function from the VERSOFT_10 package, which does the following. If successful, for given $m \times n$ matrix A an $n \times n$ interval matrix $\mathbf{W}$ is computed with the property that there exists $W \in \mathbf{W}$ such that $\{Wy \colon y \in \boldsymbol{R}^n\}$ spans the null space of A. It follows $\operatorname{rank}(W) = n - m$; however, as Rohn notes in the comments to `vernull`, it was not clear to him how to compute an inclusion of a basis of the null space of A.

One possibility to obtain a basis of $\operatorname{null}(A)$ from $\mathbf{W}$ is as follows. If a submatrix $\mathbf{X}$ of $n - m$ columns of $\mathbf{W}$ can be identified such that every matrix within $\mathbf{X}$ has full rank, then $\mathbf{X}$ is a verified inclusion of a basis of $\operatorname{null}(A)$. To find appropriate columns we perform an LU-decomposition of the transposed midpoint matrix of $\mathbf{W}$ resulting in a partial pivoting vector p. We then choose columns p_1 to p_{n-m} of $\mathbf{W}$ according to the pivoting result. The corresponding VERSOFT/INTLAB code could be as follows:

```
[W,~] = vernull(A);
[~,~,p] = lu(mid(W)','vector');
X = W(:,p(1:n-m));
```

Method 5: Rohn's Method with columns reduction.

Here we skipped the code for checking regularity of $\mathbf{X}$. It is noteworthy that `vernull` does not support interval input data. In contrast to the previously proposed methods, here we are restricted to point data.

Another way is to compute an approximate eigendecomposition of a matrix close to the midpoint of $\mathbf{W}$ by `[V,D] = eig(mid(W))`, to choose the set J of indices of $n-m$ largest eigenvalues, and to set `X = W*V(:,J)`. If $\mathbf{X}$ has full rank, then it is an inclusion of a basis of null(A). The benefit of this method is that $\mathbf{X}$ can be expected to be nearly orthogonal because Rohn's method produces a (nearly) Hermitian enclosure $\mathbf{W}$. To be on the safe side it is better to compute the approximate eigendecomposition of the Hermitian part of the midpoint of $\mathbf{W}$. A VERSOFT/INTLAB implementation of this approach could look like this:

```
[W,~] = vernull(A);
[V, D] = eig(mid(W) + mid(W)');
[~,J] = sort(diag(D),'descend');
X = W*V(:,J(1:n-m));
```

Method 6: Rohn's Method, projection of linear subspace.

As before, we skipped the code for checking regularity of $\mathbf{X}$.

3 On the accuracy of enclosures of a basis of the null space

The accuracy of an inclusion obtained by Method 1 depends on the condition number of A, the accuracy of the computation of the residual product $A\tilde{Q}_2$, and the quality of the approximate $\tilde{Q}_2$. It is noteworthy that, without an accurate computation of the residual product, the condition number affects the accuracy of the inclusion in two ways: once within the verification process by `verifylss`, and also in the evaluation of $A\tilde{Q}_2$. Nevertheless, even with a very good approximate $\tilde{Q}_2$ for a basis of the null space of A and computing the right-hand side $A\tilde{Q}_2$ with high accuracy, the accuracy of the inclusion is still restricted by the condition of A.

Another major drawback of Method 1 is the dependency on the quality of the initial approximate $\tilde{Q}_2$ for Q_2. The main reason for this dependency is that `verifylss` computes an inclusion of the actual minimum 2-norm solution. This prevents INTLAB to apply an (internal) residual iteration which is usually applied when solving square systems. In our application only an inclusion of *some* basis of null(A) is desired and thus a residual iteration as in the square case is eligible but would require a different implementation.

Method 2 circumvents this problem by the embedding into a larger square linear system. Due to the implementation of `verifylss` that applies appropriate residual iterations, the accuracy of the results computed with Method 2 is mainly restricted by the accuracy of the internal computations within INTLAB. These, in turn, are of course also influenced by the condition number of A but, as we will see in the next section, to a smaller extent than the direct computations in Method 1.

By a similar argument, the quality of initial approximations, such as (internal) preconditioners, have a relatively small influence on the accuracy of the enclosures computed by our third method. In contrast to Method 2, however, this approach could suffer from ill-conditioned solution matrices X. If the actual solution X is ill-conditioned, then even tight enclosures of X can be unusable for describing the null space of A.

Nevertheless, in practice, the basis produced by Method 3 can be expected to be well-conditioned. To see that consider the partially pivoted block LU-decomposition from the previous subsection:

$$P^*A^* = \begin{pmatrix} A_1^* \\ A_2^* \end{pmatrix} = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} U,$$

where $L_1 \in \boldsymbol{K}^{m\times m}$, $L_2 \in \boldsymbol{K}^{(n-m)\times m}$, and $U \in \boldsymbol{K}^{m\times m}$. We already discussed the regularity of L_1, U and $A_1 = U^*L_1^*$. Together with the identity $A_2^* = L_2U$, it is straightforward to show $-X_1 = A_1^{-1}A_2X_2 = (U^*L_1^*)^{-1}(U^*L_2^*)X_2 = (L_1^*)^{-1}L_2^*X_2$ (c.f. [2]). Moreover, partial pivoting implies that the absolute value of all entries of L_1 and L_2 does not exceed 1, and a well accepted rule of thumb implies that L_1 is well-conditioned. By $\|X_1\|_2 \leq \frac{\operatorname{cond}(L_1)\|L_2\|_2}{\|L_1\|_2}\|X_2\|_2$, it then follows that $\|X_1\|_2$ is of reasonable size as long as the same is true for $\|X_2\|_2$. For the specific choice $X_2 = I_{n-m,n-m}$, it follows $\|X_2\|_2 = 1$ and $\sigma_{\min}(X) \geq 1$. Hence, $\|X_1\|_2$ can be expected to be of the size of $\operatorname{cond}(L_1)$, i.e., small, and

$$\operatorname{cond}(X) = \frac{\|X\|_2}{\sigma_{\min}(X)} \leq \|X\|_2 = \sqrt{1 + \|X_1\|_2^2}$$

can be expected to be small as well. The computed basis X of the null space of A is typically not too ill-conditioned, and the accuracy of the result can be expected to be similar as for Method 2.

If a (nearly) orthogonal basis is desired, then Method 3 can be modified as described in the corresponding subsection. The accuracy of the inclusion obtained via Method 4 additionally depends an the accuracy of the computed right-hand side $A_2\tilde{Q}_{22}$ and can therefore be expected to be slightly lower than with Method 3.

3.1 Improving the accuracy

The accuracy of the computed inclusion is limited by the weakest link in the corresponding algorithm. If `verifylss` allows for a highly accurate solution of the respective linear system but the inclusion of $A_2\tilde{Q}_{22}$ in Method 4 is comparably poor, then also the accuracy of the returned basis will be poor. To circumvent this issue, intermediate computations should be done with higher precision. The algorithm "poor men's residual" introduced in INTLAB's `lssresidual` seems to be a good trade-off between accuracy and computational effort.

To apply the improved residual computation to Method 1, we exchange the products on the left-hand sides of the linear systems in Methods 1 and 4 by their `lssresidual` equivalents. This is the first method to improve the accuracy of

Methods 1 and 4. More precisely, the code is modified according to the following table.

Method	original code	replaced by
1	`B = A*intval(Q2);`	`B = lssresidual(A,-Q2);`
4	`B1 = A2*intval(-Q22);`	`B1 = lssresidual(A2,Q22);`

Table 1: Improved accuracy of left-hand sides: Method 1 → 1', Method 4 → 4'.

The other proposed Methods 2 and 3 do not suffer from inaccurate intermediate computations. There, the accuracy of the linear system solver `verifylss` is the limiting factor.

For a further improvement of the accuracy, we need to transform the problem to an easier equivalent one. A typical approach is the use of a suitable preconditioning matrix such as $S :\approx (\tilde{R}_1^*)^{-1}$. Since the quality of the interval enclosure for SA derived via interval arithmetic still depends on the condition number of A, it is important to compute the product SA using accurate dot products, for example, based on error-free transformations [9, 16].

Let `verifynull` implement any of our proposed methods, then an INTLAB code for improving the accuracy could be as follows.

```
[~,R] = qr(A');
S = inv(R(1:m,:)');
C = AccDot(S,A,[]);
X = verifynull(C);
```

Preconditioning approach to improve the accuracy of any of our methods.

The call of `AccDot` with last input parameter `[]` returns an accurate inclusion of SA using methods based on error-free transformations, cf. [9, 16]. In contrast to the first improvement, this applies to all our proposed methods.

The condition number of the interval matrix $\mathbf{C}$ is typically much better than the condition of the original system. The matrix $\mathbf{C}$ contains a matrix with the same null space as A provided that the preconditioner S is regular. A verification of this property is not necessary since our methods validate the full rank of all $C \in \mathbf{C}$, which in turn implies the regularity of S. In the following we will refer to methods with accurate preconditioning as Method 1", 2", 3" and 4", respectively.

An alternative approach to improve the accuracy of the computed inclusion is a residual iteration, where the residual is computed using accurate dot products. That is in particular more efficient if the dimension of null(A) is relatively small. Again this applies to all our proposed methods. Exemplary, the respective modification of our third method could be as follows:

```
[~,~,p] = lu(A','vector');
b = A(:,p(m+1:n));
x = A(:,p(1:m))\b;
res = AccDot(1,b,A(:,p(1:m)),-x,[]);
```

```
X = [x+verifylss(A(:,p(1:m)),res); eye(n-m)];
X = X(invperm(p),:);
```

Another accuracy improvement for Method 3.

Moreover, L and U could be used to improve the initial x.

4 Numerical results

We generate a random $m \times n$ matrix A with predefined condition number `cnd` by `gallery('randsvd',[m,n],cnd)` and compute a verified inclusion $\mathbf{X}$ of a basis of null(A) by the four proposed methods and the two methods derived from Rohn's `vernull` algorithm. To measure the accuracy of an enclosure $\mathbf{X}$ for a basis to the null space of A, we use the maximal error of the column vectors of $\mathbf{X}$, i.e.,

$$\operatorname{acc}(\mathbf{X}) := \max_k \frac{\|\operatorname{rad}(\mathbf{X}_{*k})\|_2}{\|\operatorname{m}(()\mathbf{X}_{*k})\|_2}.$$

In the following tables, the accuracy of the verified basis obtained by our four standard methods and the two proposed methods based on Rohn's `vernull` is denoted by acc1 till acc6, respectively. The computing times are denoted accordingly by t1 till t6. Moreover, we use ' and " to refer to the methods with the respective first and second improvement discussed in Subsection 3.1. For each test case, 100 samples were computed and the median is displayed. If the relative error is larger than 1 or the proof of regularity fails, then the computed inclusion gives hardly any useful information about the null space of A and the verification is considered to be failed. We use "–" to denote this case.

Table 2 suggests that the methods can be ordered according to the accuracy of the computed inclusions. The best results are obtained with Method 2, closely followed by Method 3. A possible reason for the slight difference in accuracy between these two methods is the typically increased condition number of the smaller system matrix. With a noticeable drop in accuracy the Methods 1, 4, 5 and 6 follow in order. For the samples with condition number 10^{14} and $n \geq 500$, a full rank of the computed inclusions could be verified only for Methods 2 and 3. All other approaches failed. Moreover, for ill-conditioned problems with condition number greater or equal to 10^{10} the relative accuracy of an inclusion computed by Rohn's method is larger than 1, thus giving no information about the null space except that A has full rank. For the reason above, we consider these cases as failures and use "–" to denote them.

The picture changes significantly if the improvements discussed in Subsection 3.1 are applied. In Table 3 we compare the accuracy results for our proposed methods with the respective improvements. The Methods 5 and 6 are missing in the table because Rohn's `vernull` function is not applicable to interval input so that the preconditioning approach cannot be used.

Using the accurate preconditioning approach, all methods provide more or less the same high accuracy with only minor differences. If the product $A\tilde{Q}_2$ in Method 1 is evaluated in higher precision, then the accuracy is comparable

m	n	cnd	acc1	acc2	acc3	acc4	acc5	acc6
50	100	10^5	8.2e-11	2.2e-14	4.1e-14	1.6e-10	3.0e-09	9.5e-09
50	100	10^{10}	5.5e-06	1.4e-09	2.6e-09	1.1e-05	–	–
50	100	10^{14}	5.4e-02	1.2e-05	2.4e-05	9.8e-02	–	–
50	200	10^5	1.4e-10	4.1e-14	4.8e-14	6.1e-10	4.2e-09	2.0e-08
50	200	10^{10}	9.7e-06	2.8e-09	3.6e-09	4.5e-05	–	–
50	200	10^{14}	1.2e-01	2.6e-05	3.2e-05	–	–	–
50	500	10^5	2.1e-10	4.2e-14	5.3e-14	1.6e-09	2.8e-09	2.2e-08
50	500	10^{10}	1.4e-05	2.7e-09	3.9e-09	1.1e-04	–	–
50	500	10^{14}	–	2.4e-05	3.6e-05	–	–	–
50	1000	10^5	2.0e-10	6.1e-14	5.7e-14	2.1e-09	2.4e-09	2.5e-08
50	1000	10^{10}	1.4e-05	3.5e-09	4.1e-09	1.6e-04	–	–
50	1000	10^{14}	–	2.9e-05	3.7e-05	–	–	–
100	200	10^5	2.0e-10	5.7e-14	1.2e-13	4.8e-10	7.7e-09	3.7e-08
100	200	10^{10}	1.2e-05	3.4e-09	7.6e-09	3.0e-05	–	–
100	200	10^{14}	1.5e-01	2.9e-05	6.8e-05	–	–	–
100	500	10^5	3.2e-10	6.8e-14	1.5e-13	2.1e-09	6.8e-09	6.1e-08
100	500	10^{10}	2.0e-05	3.9e-09	9.9e-09	1.4e-04	–	–
100	500	10^{14}	–	3.6e-05	9.6e-05	–	–	–
100	1000	10^5	3.1e-10	7.8e-14	1.6e-13	3.0e-09	5.6e-09	7.1e-08
100	1000	10^{10}	1.9e-05	4.5e-09	1.0e-08	1.9e-04	–	–
100	1000	10^{14}	–	3.6e-05	1.1e-04	–	–	–
200	500	10^5	5.0e-10	1.1e-13	4.4e-13	2.1e-09	2.1e-08	1.9e-07
200	500	10^{10}	2.8e-05	6.2e-09	2.7e-08	1.3e-04	–	–
200	500	10^{14}	–	7.0e-05	4.7e-04	–	–	–
200	1000	10^5	5.3e-10	1.1e-13	5.1e-13	4.6e-09	1.7e-08	2.3e-07
200	1000	10^{10}	3.0e-05	6.1e-09	3.1e-08	2.8e-04	–	–
200	1000	10^{14}	–	7.2e-05	6.5e-04	–	–	–

Table 2: Relative errors of inclusions obtained by our Methods 1-4 and Rohn's Methods 5-6.

to the results for Methods 2 and 3. By the same approach also the accuracy of Method 4 is increased significantly, providing useful inclusions for all our test samples. Nevertheless, in comparison to Methods 2 and 1', the accuracy of Method 4' still falls behind by two magnitudes.

An accurate residual iteration as discussed at the end of Subsection 3.1 can improve the accuracy even further. However, for the tested dimensions the accurate residual computation is more expensive than the preconditioning approach which already produces highly accurate enclosures for a basis of the null space. As mentioned before, the accurate residual iteration is only sensible for null spaces with comparably small dimensions. Since this approach involves further specific adaptations for each of our methods, we refrain from presenting numerical results for accuracy improvements based on accurate residual iterations.

m	n	cnd	acc1'	acc4'	acc1"	acc2"	acc3"	acc4"
50	100	10^5	2.1e-14	5.1e-12	1.6e-14	2.8e-14	2.7e-14	4.7e-14
50	100	10^{10}	1.3e-09	3.7e-07	1.6e-14	2.8e-14	2.8e-14	4.9e-14
50	100	10^{14}	1.4e-05	3.3e-03	1.6e-14	2.8e-14	2.7e-14	4.8e-14
50	200	10^5	4.0e-14	7.2e-12	2.3e-14	5.6e-14	3.0e-14	9.9e-14
50	200	10^{10}	2.8e-09	5.1e-07	2.3e-14	5.6e-14	3.1e-14	1.0e-13
50	200	10^{14}	3.1e-05	4.6e-03	2.3e-14	5.6e-14	3.0e-14	9.9e-14
50	500	10^5	3.8e-14	9.8e-12	3.1e-14	9.7e-14	3.1e-14	2.0e-13
50	500	10^{10}	2.5e-09	7.2e-07	3.1e-14	9.7e-14	3.1e-14	2.0e-13
50	500	10^{14}	4.2e-05	7.0e-03	3.1e-14	9.7e-14	3.1e-14	1.9e-13
50	1000	10^5	4.1e-14	1.1e-11	2.8e-14	9.5e-14	3.1e-14	2.5e-13
50	1000	10^{10}	2.6e-09	8.2e-07	2.8e-14	9.5e-14	3.0e-14	2.5e-13
50	1000	10^{14}	4.5e-05	7.7e-03	2.8e-14	9.5e-14	3.1e-14	2.5e-13
100	200	10^5	5.4e-14	7.6e-12	4.1e-14	7.3e-14	8.6e-14	1.6e-13
100	200	10^{10}	3.1e-09	5.0e-07	4.1e-14	7.3e-14	8.5e-14	1.6e-13
100	200	10^{14}	3.8e-05	4.9e-03	4.1e-14	7.3e-14	8.5e-14	1.6e-13
100	500	10^5	6.3e-14	1.2e-11	5.8e-14	1.5e-13	9.7e-14	3.5e-13
100	500	10^{10}	3.7e-09	8.0e-07	5.8e-14	1.5e-13	9.9e-14	3.6e-13
100	500	10^{14}	6.6e-05	8.0e-03	5.8e-14	1.5e-13	9.8e-14	3.6e-13
100	1000	10^5	6.0e-14	1.4e-11	5.4e-14	1.6e-13	9.9e-14	4.7e-13
100	1000	10^{10}	3.5e-09	9.9e-07	5.4e-14	1.6e-13	9.8e-14	4.7e-13
100	1000	10^{14}	7.4e-05	1.0e-02	5.4e-14	1.6e-13	9.9e-14	4.7e-13
200	500	10^5	1.1e-13	1.3e-11	1.1e-13	2.1e-13	3.0e-13	6.3e-13
200	500	10^{10}	5.9e-09	8.0e-07	1.1e-13	2.1e-13	3.0e-13	6.4e-13
200	500	10^{14}	1.5e-04	–	1.1e-13	2.1e-13	3.0e-13	6.3e-13
200	1000	10^5	1.0e-13	1.8e-11	1.1e-13	2.7e-13	3.3e-13	9.7e-13
200	1000	10^{10}	5.8e-09	1.1e-06	1.1e-13	2.7e-13	3.2e-13	9.6e-13
200	1000	10^{14}	2.2e-04	–	1.1e-13	2.7e-13	3.2e-13	9.6e-13

Table 3: Accuracy results for our proposed methods with applied improvements.

In the numerical tests of our proposed verification methods, we did not observe a correlation between the computing time and the condition number of A. For the comparison of computing times in Table 4, we therefore restrict the presentation to the samples with condition number 10^5 for which all discussed methods produce useful enclosures for a basis of the null space. We display the ratio of the corresponding computing times with respect to Method 3, the fastest of the proposed methods.

By t5 we do not refer to the complete computing time of Method 5 but only to the time taken by Rohn's `vernull` function. This is the reason why the computing times for Method 6 are omitted from the table. The MATLAB implementation `AccDot` has a significant interpretation overhead and dictates the overall running time. In this context, t1" serves as a representative for the computing times t2", t3" and t4" which are all very similar. Without

m	n	t1	t2	t4	t5	t1'	t4'	t1"
50	100	8.0	4.6	2.1	52.7	8.6	2.2	292.6
50	200	3.6	2.0	1.8	16.4	3.8	1.9	144.0
50	500	10.5	7.0	3.1	28.9	10.7	3.2	235.4
50	1000	25.7	20.1	5.1	59.9	26.0	5.5	305.2
100	200	2.3	1.2	1.6	14.0	2.5	1.7	174.0
100	500	7.8	4.8	2.5	26.1	8.0	2.6	334.0
100	1000	17.7	12.9	3.5	45.6	18.0	3.7	404.7
200	500	6.4	3.2	2.0	27.2	6.6	2.1	548.0
200	1000	10.4	6.8	2.3	30.3	10.7	2.4	455.2

Table 4: Computing times relative to t3.

that significant interpretation overhead the computing times would be more competitive. In this scenario, however, the preconditioning makes sense only for extremely ill-conditioned matrices A.

Compared to the computational overhead of the accurate preconditioning, the impact of the `lssresidual` calls in Methods 1' and 4' is rather small. Considering the significantly improved accuracy results presented in Table 3, it seems to be generally beneficial to choose Methods 1' and 4' over their unmodified counterparts. Moreover, for some reason `verifylss` solves the embedding square linear system of Method 2 more efficiently than the smaller underdetermined linear system of Method 1. Since the former produces more accurate results, there seems to be no actual benefit in using Method 1 or 1' over Method 2. Method 4', on the other hand, may not produce the same accurate results but involves significantly less computational effort.

Finally, in Table 5, we show verified upper bounds for the condition number of the inclusions $\mathbf{X}$ obtained by the respective methods. The values are computed by INTLAB's routine `verifysingvalall` and satisfy $\frac{\sigma_1(X)}{\sigma_{n-m}(X)} \leq \text{cnd}$ for all $X \in \mathbf{X}$. Bounds close to 1 are desirable.

The condition numbers for Methods 2 and 3 show a correlation with the problem dimension but no significant influence of the condition number of A. To a certain extend this is also true for the other columns. In particular the good values for cnd1' demonstrate a good quality of the approximation $\tilde{Q}_2$ for a basis of the null space. The outliers in the columns for cnd1, cnd4 and cnd4' can be explained by the accuracy of the corresponding inclusions rather than the condition number of the actual solution X. As an example, consider Method 4' applied to the samples ($m = 100, n = 200, \texttt{cnd} = 10^{14}$). The corresponding median of the maximum relative error of the basis vector inclusions in Table 3 is acc4' $= 4.9 \cdot 10^{-3}$. Assuming similar relative errors for all columns of $\mathbf{X}$, the spectral norm grows roughly with $\sqrt{n-m}$, so that $\|\mathbf{X} - \mathrm{m}(()\mathbf{X})\|_2 \leq \sqrt{m-n} \cdot \max_k \frac{\|\mathrm{rad}\,(\mathbf{X}_{*k})\|_2}{\|\mathrm{m}(()\mathbf{X}_{*k})\|_2} = 0.049$. To compute the actual value of $\sup_{X \in \mathbf{X}} \frac{\sigma_1(X)}{\sigma_{n-m}(X)}$ is an NP-hard problem that we do not attempt to tackle here. The values presented in Table 5 are based on perturbation bounds for the singular values.

m	n	cnd	cnd1	cnd1'	cnd2	cnd3	cnd4	cnd4'	cnd5	cnd6
50	100	10^5	1.00	1.00	1.00	11.3	1.00	1.00	14.6	1.00
50	100	10^{10}	1.00	1.00	1.00	11.1	1.00	1.00	–	–
50	100	10^{14}	1.51	1.00	1.00	11.5	1.96	1.03	–	–
50	200	10^5	1.00	1.00	1.00	16.0	1.00	1.00	26.6	1.00
50	200	10^{10}	1.00	1.00	1.00	15.6	1.00	1.00	–	–
50	200	10^{14}	3.92	1.00	1.00	15.9	–	1.07	–	–
50	500	10^5	1.00	1.00	1.00	23.0	1.00	1.00	46.2	1.00
50	500	10^{10}	1.00	1.00	1.00	23.6	1.00	1.00	–	–
50	500	10^{14}	–	1.00	1.00	24.3	–	1.09	–	–
50	1000	10^5	1.00	1.00	1.00	30.7	1.00	1.00	–	–
50	1000	10^{10}	1.00	1.00	1.00	30.4	1.00	1.00	–	–
50	1000	10^{14}	–	1.00	1.00	30.8	–	1.10	–	–
100	200	10^5	1.00	1.00	1.00	19.7	1.00	1.00	26.4	1.00
100	200	10^{10}	1.00	1.00	1.00	19.3	1.00	1.00	–	–
100	200	10^{14}	10.25	1.00	1.00	19.5	–	1.07	–	–
100	500	10^5	1.00	1.00	1.00	30.6	1.00	1.00	55.2	1.00
100	500	10^{10}	1.00	1.00	1.00	30.8	1.00	1.00	–	–
100	500	10^{14}	–	1.00	1.00	32.2	–	1.12	–	–
100	1000	10^5	1.00	1.00	1.00	41.7	1.00	1.00	–	–
100	1000	10^{10}	1.00	1.00	1.00	43.3	1.00	1.00	–	–
100	1000	10^{14}	–	1.00	1.00	42.4	–	1.15	–	–
200	500	10^5	1.00	1.00	1.00	39.7	1.00	1.00	59.9	1.00
200	500	10^{10}	1.00	1.00	1.00	40.3	1.00	1.00	–	–
200	500	10^{14}	–	1.00	1.00	41.4	–	1.41	–	–
200	1000	10^5	1.00	1.00	1.00	59.2	1.00	1.00	99.3	1.00
200	1000	10^{10}	1.00	1.00	1.00	56.4	1.01	1.00	–	–
200	1000	10^{14}	–	1.01	1.00	61.6	–	1.73	–	–

Table 5: Condition numbers of obtained inclusions

Since all the singular values are clustered around 1, the computed bounds for $\frac{\sigma_1(\mathbf{X})}{\sigma_{n-m}(\mathbf{X})}$ are hardly better than $\frac{\sigma_1(\mathrm{m}(()\mathbf{X}))+\|\mathbf{X}-\mathrm{m}(()\mathbf{X})\|_2}{\sigma_{n-m}(\mathrm{m}(()\mathbf{X}))-\|\mathbf{X}-\mathrm{m}(()\mathbf{X})\|_2} \approx \frac{1+0.049}{1-0.049} \approx 1.1$. This corresponds to the value 1.07 in Table 5. The other outliers and also the values very close to 1 can be explained in a similar way.

It is noteworthy that none of the tested methods produce ill-conditioned enclosures $\mathbf{X}$. Nevertheless, in terms of orthogonality of the basis, the results obtained by Method 3 or 5 are not desirable. The other methods produce enclosures for a nearly orthogonal basis of the null space if the verification was successful.

5 Conclusion

We presented four verification methods to compute an inclusion of a basis of the null space of a matrix. As illustrated in some numerical results, the presented methods are applicable to matrices of large size and with large condition number. We clarified that Methods 2 and 4' are best for obtaining tight enclosures for a nearly orthogonal basis of the null space; the former providing a higher accuracy and the latter involving a smaller computational footprint. The fastest and still accurate verification method for computing some basis of the null space of a matrix is Method 3. The inclusions obtained by this method are not orthogonal but typically well-conditioned.

Furthermore, we showed techniques to improve the accuracy of our presented methods. The accurate preconditioning approach yields a particularly high accuracy of the basis enclosures. Nevertheless, the involved computational efforts make it suitable only for extremely ill-conditioned problems where the standard methods fail.

The presented methods could be useful to study verification methods based on null space methods for saddle point linear systems arising from the KKT condition in optimization (cf. [1, 2]).

Acknowledgements

The authors wish to thank Florian Bünger for many fruitful remarks. We are also indebted to the anonymous referees for their constructive reports that gave this paper a much better structure.

References

[1] M. Benzi, G.H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta numerica*, 14:1–137, 2005.

[2] R. Fletcher and T. Johnson. On the stability of null-space methods for kkt systems. *SIAM Journal on Matrix Analysis and Applications*, 18(4):938–958, 1997.

[3] T.N.E. Greville. The pseudoinverse of a rectangular or singular matrix and its application to the solution of systems of linear equations. *SIAM review*, 1(1):38–43, 1959.

[4] W.M. Kahan. A more complete interval arithmetic. *Lecture notes for a summer course at the University of Michigan*, 1968.

[5] B. Kearfott, M. Nakao, A. Neumaier, S.M. Rump, S.P. Shary, and P. Van Hentenryck. Standardized notation in interval analysis. *Computational Technologies*, 15(1):7–13, 2010.

[6] R. Krawczyk. Newton-algorithmen zur bestimmung von nullstellen mit fehlerschranken. *Computing*, 4(3):187–201, 1969.

[7] A. Minamihata, K. Sekine, T. Ogita, S.M. Rump, and S. Oishi. Improved error bounds for linear systems with h-matrices. *Nonlinear Theory and Its Applications, IEICE*, 6(3):377–382, 2015.

[8] A. Neumaier. *Interval methods for systems of equations*, volume 37. Cambridge university press, 1990.

[9] T. Ogita, S.M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6):1955–1988, 2005.

[10] S. Oishi, K. Ichihara, M. Kashiwagi, K. Kimura, X. Liu, H. Masai, Y. Morikura, T. Ogita, K. Ozaki, S.M. Rump, K. Sekine, A. Takayasu, and N. Yamanaka. *Principle of Verified Numerical Computations.* Corona publisher, Tokyo, Japan, 2018.

[11] J. Rohn. Verification of linear (in) dependence in finite precision arithmetic. *Mathematics in Computer Science*, 8(3-4):323–328, 2014.

[12] S.M. Rump. Intlab-interval laboratory. In *Tibor Csendes, editor, Developments in reliable computing*, pages 77–104. Kluwer Academic Publishers, 1999. http://www.ti3.tuhh.de/intlab.

[13] S.M. Rump. Inversion of extremely ill-conditioned matrices in floating-point. *Japan Journal of Industrial and Applied Mathematics*, 26(2-3):249–277, 2009.

[14] S.M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, 2010.

[15] S.M. Rump. Accurate solution of dense linear systems, part ii: Algorithms using directed rounding. *Journal of computational and applied mathematics*, 242:185–212, 2013.

[16] S.M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part i: Faithful rounding. *SIAM Journal on Scientific Computing*, 31(1):189–224, 2008.

[17] P. Wolfe. The reduced gradient method. *Technical report, The RAND Corporation, Santa Monica, CA. Unpublished.*, 1962.

[18] T. Yamamoto. Error bounds for approximate solutions of systems of equations. *Japan Journal of Applied Mathematics*, 1(1):157, 1984.