2008

TR-2008009: Solving Homogeneous Linear Systems with Weakly Randomized Additive Preprocessing

Victor Y. Pan

Guoliang Qian

Solving Homogeneous Linear Systems with Weakly Randomized Additive Preprocessing *

Victor Y. Pan[1] and Guoliang Qian[2]

[1]Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/
[2] Ph.D. Program in Computer Science,
The City University of New York,
New York, NY 10036 USA
gqian@gc.cuny.edu

Abstract

By combining our weakly randomized preconditioning with aggregation and other known and novel techniques, we facilitate the solution of a homogeneous linear system of equations. We demonstrate the power of this approach and show some extensions.

Key words: Linear system of equations, Null space, Weakly random additive preprocessing

*Supported by PSC CUNY Awards 68291–0037 and 69330–0038. Some results of this paper have been presented at the International Conferences on the Matrix Methods and Operator Equations in Moscow, Russia, in June of 2005 and July 2007, on the Foundations of Computational Mathematics (FoCM'2005) in Santander, Spain, in July 2005, and on Industrial and Applied Mathematics, in Zürich, Switzerland, in July 2007, as well as at the SIAM Annual Meeting, in Boston, in July 2006, and at the International Workshop on Symbolic-Numeric Computation (SNC'07) in London, Ontario, Canada, in July 2007.

1 Introduction

1.1 Homogeneous linear systems of equations, null vectors, nmbs, and sample extensions

The solution vectors $\mathbf{y}$ of a homogeneous linear system of equations $A\mathbf{y} = \mathbf{0}$ are called (right) *null vectors* of the matrix A. They form its (right) *null space* $N(A)$. Given a matrix A, we seek its null vectors and its *null matrix basis*, that is, a matrix whose columns form a basis for the null space $N(A)$. Hereafter we use the abbreviations *nmb* and nmb(A) and let M^T denote the transpose of a matrix or a vector M. Our subject can be linked to some other fundamental matrix computations.

Example 1.1. *(Cf. Remark 3.1.) Solution* $\mathbf{y}$ *of a (real) non-homogeneous linear system of* n *equations* $A\mathbf{y} = \mathbf{b}$ *with* n *unknowns* $y_1, \dots, y_n$ *is a subvector of a scaled null vector* $\mathbf{z} = (z_i)_{i=0}^n$ *of the matrix* $(-\mathbf{b}, A)$ *formed by appending the column* $-\mathbf{b}$ *to the matrix* A*. Namely we have* $\tilde{\mathbf{z}} = \frac{\mathbf{b}^T A\tilde{\mathbf{z}}}{\mathbf{b}^T\mathbf{b}}\mathbf{y}$ *for* $\tilde{\mathbf{z}} = (z_i)_{i=1}^n$.

Example 1.2. *An eigenvector* $\mathbf{y}$ *associated with an eigenvalue* λ *of a matrix* M *is a null vector of a matrix* $A(\lambda) = \lambda I - M$*,* I *denoting the identity matrix.*

1.2 The customary and our approaches

SVD-based computation of nmbs and null vectors is most reliable, but most costly. The other customary algorithms employ LU and QR factorizations. They require pivoting, which "usually degrades the performance" [17, page 119], readily destroys matrix structure and sparseness and threatens or undermines block matrix algorithms.

Furthermore, the large scale sparse or structured linear systems of equations must be solved by iterative algorithms (such as the Conjugate Gradient algorithms), which recursively multiply the input matrix and its transpose by vectors. These algorithms, however, generally converge too slowly where the input matrix is ill conditioned.

Our alternative approach is less costly, readily preserves matrix structure and sparseness, can be naturally extended to preconditioning the input matrix, and supports rapid iterative refinement of nmbs and null vectors, so that we can obtain them with high accuracy at a lower cost.

Let us briefly introduce this approach. Assume a real $n \times n$ matrix A that has a rank $\rho < n$ and a positive nullity $r = n - \rho$. Let U and V be a pair of random real $n \times r$ matrices and let $P = UV^T$ and $C = A + P$. Then we can expect that the matrices U, V, and UV^T have rank r and that the matrix C is nonsingular. (Later we will formally support these claims.) If so, we can easily prove that the matrix $C^{-1}U$ is an nmb(A).

We call the matrix P *additive preprocessor* and call the transform $A \to A + P$ *additive preprocessing*. We use the respective abbreviations *APP* and *A-preprocessing*.

It is plausible that, while countering singularity, randomized A-preprocessing can turn a singular well conditioned matrix A into a nonsingular ill conditioned one, but the theoretical and experimental study in [34], [35] shows that this is unlikely wherever a random APP P has been scaled so that the ratio $\frac{||A||}{||P||}$ is neither large nor small. Moreover it is sufficient to use *weakly random* APPs $P = UU^T$ for structured and sparse generators U defined by fewer random parameters.

We can apply our approach numerically, with rounding. Instead of a singular input matrix A, we would assume a nearby ill conditioned one, $A + E$, and instead of null vectors and nmbs would seek singular vectors and bases for the singular spaces associated with the smallest singular values of the latter matrix. If the norm $||E||$ is a small fraction of the smallest singular value of a matrix A, then our algorithms approximate some bases for the respective spaces of singular vectors. In this case the transition $A \to C = A + P$ improves conditioning of such an ill conditioned matrix A, and our APPs turn into *A-preconditioners*. In the present paper we describe and analyze in some detail this approach to computing null vectors and nmbs. In the papers [31], [32], [33], [37], and [40] we cover its further variations and extensions.

We conclude with a simple demonstration of another useful feature of preconditioning.

Example 1.3. $A = \begin{pmatrix} 1+\delta & 1 & \delta \\ 1 & 1-\delta & \delta \\ 0 & 1 & -1 \end{pmatrix}$, $U = V = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$, $C = A + UV^T = \begin{pmatrix} 2+\delta & 0 & 1+\delta \\ 0 & 2-\delta & -1+\delta \\ 1 & 2 & 0 \end{pmatrix}$. *Here* $n = 3$, $\rho = 2$, *and the* 2×2 *leading principal (northwestern) submatrix of the matrix* A *is ill conditioned for smaller values* $|\delta|$, *whereas all* 2×2 *submatrices of the matrix* C *are well conditioned.*

The example demonstrates that to some extent random A-preconditioning can serve in lieu of pivoting. This is a general feature because random matrices tend to be well conditioned.

1.3 Organization of the paper

We organize our paper as follows. After presenting some definitions in the next section and some basic results in Section 3, we compute nmbs in Section 4. In Section 5 we extend our algorithms to approximate singular spaces and to strengthen A-preconditioning. In Section 6 we comment on preserving and exploiting matrix structure. Section 7 covers our numerical tests. They have been designed by the first author and performed by his coauthor. Otherwise the paper (including all typos and errors) is due to the first author.

For completeness we study the general case of a rectangular input matrix and supply the respective definitions and results, but Section 3.2 enables us to reduce most of our study to the simpler case of a square input matrix.

Acknowledgement. Valuable comments and criticism by the referee enabled us to improve our original draft substantially.

2 Basic definitions

We will reuse the definition and abbreviations from the Introduction, in particular nmbs, A-preprocessing, A-preconditioning, A-modification, and APPs.

We assume or slightly extend the customary definitions for matrix computations in [6], [7], [9], [17], [18]. This includes the definitions of the Hermitian, unitary (orthonormal), and singular matrices, full-rank and rank deficient matrices, the transpose $A^T$of a matrix or a vector A and the Hermitian, that is, complex conjugate transpose A^H, the $k \times k$ identity matrix $I = I_k$, the $k \times l$ matrix $0 = 0_{k,l}$ filled with zeros, the range of a matrix A (that is the span of its column vectors), denoted range A, the (right) null space $N(A) = \{\mathbf{y} : A\mathbf{y} = \mathbf{0}\}$, the left null space $LN(A) = \{\mathbf{x} : \mathbf{x}^H A = \mathbf{0}^H\}$, the rank $\rho = \text{rank}\, A$, the left nullity $\text{lnul}\, A = m - \rho$, the right nullity $\text{rnul}\, A = n - \rho$, the nullity $\text{nul}\, A = \min\{m, n\} - \rho$, and the thin QR and QRP factorizations. For $m \geq n$ and an $m \times n$ matrix A of full rank n, the Q-factor $Q(A)$ and the R-factor $R(A)$ in the thin QR factorization are unique provided the R-factor has positive diagonal entries [17, Theorem 5.2.2]. We write A^+ for the *Moore-Penrose generalized inverse* of an $m \times n$ matrix A of a rank ρ (also called *pseudo inverse*). We have $A^+ = (A^H A)^{-1} A^H$ if $m \geq n = \rho$, $A^+ = A^H (AA^H)^{-1}$ if $m = \rho \leq n$, and $A^+ = A^{-1}$ if $m = n = \rho$.

$\text{diag}(B_1, B_2)$ (resp. $\text{diag}(B_i)_{i=1}^k$) denotes the 2×2 (resp. $k \times k$) block diagonal matrix with diagonal blocks B_1 and B_2 (resp. $B_1, \ldots, B_k$), whereas (B, C) (resp. $(B_i)_{i=1}^k$) denotes the 1×2 (resp. $1 \times k$) block matrix with blocks B and C (resp. $B_1, \ldots, B_k$).

A matrix B is a *matrix basis* for its range if its column set is linearly independent. A null vector, a null basis, and a null matrix basis (nmb) for a matrix A is a vector in, a basis for, and a matrix basis for its (right) null space $N(A)$, respectively. Similar concepts are defined for the left null space $LN(A)$.

$|\Delta|$ is the cardinality of a set Δ. *Random sampling* of elements from a finite set Δ is their selection from the set Δ at random, independently of each other, and under the uniform probability distribution on Δ. A matrix is *random* if its entries are randomly sampled (from a fixed finite set Δ). A $k \times l$ *random unitary* matrix A for $k \geq l$ is the $k \times l$ Q-factor $Q(A)$ in the thin QR factorization of random $k \times l$ matrix A of full rank l where the square R-factor has only positive diagonal entries. (QR factorization reveals whether a matrix A has full rank, and if it does not, we can regenerate the matrix.)

In the rest of this section we cover some basic definitions for numerical matrix computations.

$||A||_l$ is the l-norms, for $l = 1, 2, \infty$. $||A||_F$ is the Frobenius norm. We write $||A|| = ||A||_2$, call a matrix A normalized if $||A|| = 1$, and recall (cf. [17, Sections 2.3.2 and 2.3.3]) that $||U|| = 1$ if $U^H U = I$, $||A|| = ||A^H||$, $||A||_F = ||A^H||_F$,

$||A||_1 = ||A^H||_\infty$, and furthermore
$\frac{1}{\sqrt{n}}||A||_\infty \le ||A|| \le \sqrt{m}||A||_\infty$, $\frac{1}{\sqrt{m}}||A||_1 \le ||A|| \le \sqrt{n}||A||_1$,
$||A|| \le \sqrt{||A||_1||A||_\infty}$, $||A|| \le ||A||_F \le \sqrt{n}||A||$ where A is an $m \times n$ matrix.

For an $m \times n$ matrix A of a rank ρ, its *Singular Value Decomposition* (hereafter *SVD*, also called the *full SVD*) is given by the equation $A = S\Sigma T^H$ where $S = (\mathbf{s}_j)_{j=1}^m$ and $T = (\mathbf{t}_j)_{j=1}^n$ are square unitary matrices; $\Sigma = \mathrm{diag}(\Sigma^{(\rho)}, 0_{l,r})$ is an $m \times n$ matrix; $l = \mathrm{lnul}\, A = m - \rho$, $r = \mathrm{rnul}\, A = n - \rho$, $\Sigma^{(\rho)} = \mathrm{diag}(\sigma_j)_{j=1}^\rho$ is a diagonal matrix; $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_\rho > 0$, and $\sigma_j = 0$ for $j > \rho$, and we write $\sigma_j = +\infty$ for $j < 1$. The scalars σ_j for $j \ge 1$ are the *singular values* of the matrix A, and $||A|| = \sigma_1$. The vectors $\mathbf{s}_j$ for $j = 1, \ldots, m$ and $\mathbf{t}_j$ for $j = 1, \ldots, n$ are the associated left and right *singular vectors*, respectively, so that the null vectors are the singular vectors associated with the singular value zero. The singular vectors associated with a fixed singular value or with a fixed set of them form the associated singular space. The decomposition $A = S^{(\rho)}\Sigma^{(\rho)}T^{(\rho)H} = \sum_{j=1}^\rho \sigma_j \mathbf{s}_j \mathbf{t}_j^H$ is the *compact SVD* of the matrix A. $A^+ = \sum_{j=1}^\rho \sigma_j^{-1} \mathbf{t}_j \mathbf{s}_j^H$.

$\mathrm{cond}_l\, A = ||A||_l||A^+||_l$ is the condition number of a matrix A under the matrix norm $||\cdot||_l$. We write $\mathrm{cond}\, A$ for $\mathrm{cond}_2\, A$. $\mathrm{cond}\, A = \frac{\sigma_1(A)}{\sigma_\rho(A)}$ for a matrix A of a rank ρ.

We write $n \gg d$ where the ratio $\frac{n}{d}$ is large. A matrix A of a rank ρ is *ill conditioned* if $\sigma_1 \gg \sigma_\rho$ and is *well conditioned* otherwise. The concepts "large", "ill" and "well conditioned" are quantified in the context of the computational task and computer environment. A matrix A of any rank $\rho > 1$ is ill conditioned if $\sigma_j(A) \gg \sigma_{j+1}(A)$ and if the ratios $\frac{\sigma_1(A)}{\sigma_j(A)}$ and $\frac{\sigma_{j+1}(A)}{\sigma_\rho(A)}$ are not large for some j, $1 \le j < \rho$, but such a matrix A of a larger rank can be ill conditioned even if $\frac{\sigma_j(A)}{\sigma_{j+1}(A)} \le c$ for all j and a smaller bound $c > 1$. E.g., we can have $\mathrm{cond}\, A = 2^{100}$ for $c = 2$ and $\mathrm{rank}\, A = 101$.

The norms $||A||_1 = \max_j \sum_i |a_{ij}|$ and $||A||_\infty = ||A^H||_1$ can be computed in about mn flops for an $m \times n$ matrix $A = (a_{ij})_{i,j}$, but one can yield estimates for these norms in expected linear time [43, Section 5.3.1]. See [17, Sections 2.3.2, 2.3.3, 3.5.4, and 12.5], [18, Chapter 15], and [43, Section 5.3] on other effective norm and condition estimators.

We call the (right) singular space associated with the r smallest singular values of a matrix A the (right) *r-tail* of the SVD of this matrix.

3 Some basic results

Next we sketch and then prove some basic results. In our sketch we write $C = A + UV^H$ and $r = \mathrm{rank}(UV^H)$ and let "$\Longrightarrow$" stand for "implies" and "$\Longleftrightarrow$" for "if and only if".

3.1 Determination of nmbs

Assuming that A and C are $m \times n$ matrices and $\operatorname{rank} C = n \le m$, we have

$$N(A) \subseteq \operatorname{range}(C^+U),$$

$$\{r = \operatorname{nul} A\} \Longleftrightarrow \{N(A) = \operatorname{range}(C^+U)\} \Longleftrightarrow \{AC^+U = 0\},$$

$$\{X \text{ is an nmb}(AC^+U)\} \Longleftrightarrow \{C^+UX \text{ is an nmb}(A)\}.$$

Theorem 3.1. *Suppose $m \ge n$ and for an $m \times n$ matrix A of a rank ρ and a pair of two matrices U of size $m \times r$ and V of size $n \times r$, the matrix $C = A + UV^H$ has full rank n. Then*

$$r \ge \operatorname{rank} U \ge n - \rho = \operatorname{nul} A, \tag{3.1}$$

$$N(A) \subseteq \operatorname{range}(C^+U). \tag{3.2}$$

Furthermore if

$$r = \operatorname{rank} U = n - \rho = \operatorname{nul} A, \tag{3.3}$$

then

$$C^+U \text{ is an nmb}(A), \tag{3.4}$$

$$V^HC^+U = I_r. \tag{3.5}$$

Proof. Bound (3.1) follows because $\operatorname{rank}(B + C) \le \operatorname{rank} B + \operatorname{rank} C$. If $\mathbf{y} \in N(A)$, then $C\mathbf{y} = (A + UV^H)\mathbf{y} = UV^H\mathbf{y}$, and therefore

$$\mathbf{y} = C^+U(V^H\mathbf{y}). \tag{3.6}$$

This proves (3.2).

(3.4) immediately follows from (3.2) and (3.3).

To prove (3.5), pre-multiply equation (3.6) by V^H, recall equation (3.4), and deduce that $(V^HC^+U - I_r)V^HC^+U = 0$. Now (3.5) follows unless the matrix V^HC^+U is singular, but if it is, then $V^HC^+U\mathbf{z} = \mathbf{0}$ for some nonzero vector $\mathbf{z}$. Let us write $\mathbf{w} = C^+U\mathbf{z}$, so that $V^H\mathbf{w} = \mathbf{0}$ and $\mathbf{w} \in \operatorname{range}(C^+U) = N(A)$. It follows that $A\mathbf{w} = \mathbf{0}$, and therefore, $C\mathbf{w} = A\mathbf{w} + UV^H\mathbf{w} = \mathbf{0}$. Now recall that the matrix C has full rank and conclude that $\mathbf{w} = \mathbf{0}$. Consequently, $\mathbf{z} = \mathbf{0}$ because the matrix C^+U has full rank. □

Corollary 3.1. *Under the assumptions of Theorem 3.1 let equations (3.1) and (3.2) hold. Then C^+UX is an nmb(A) if and only if X is an nmb(AC^+U).*

3.2 The right and left null spaces

Theorem 3.1 and Corollary 3.1 can be readily extended to the case of the left null space and left nmbs for an $m \times n$ matrix A where $m \le n$ because $LN(A) = N(A^T)$. In particular, $\operatorname{range}(V^HC^+) = LN(A)$ if A and $C = A + UV^H$ are $m \times n$ matrices, $m \le n$, U is an $m \times r$ matrix, V is an $n \times r$ matrix, the matrices C, U, and V have full rank, and $r = \operatorname{lnul} A$. Now instead of the latter equation

assume the bound $r > \operatorname{lnul} A$. Then YV^HC^+ is a left nmb(A) if and only if Y is a left nmb(V^HC^+A). If we seek both left and right nmbs for the matrix A, we can rely on the same factorization of the matrix C.

The following results relate the left and right nmbs to the *Schur aggregate* $G = I_r - V^HC^+U$ (also called Gauss transform) [31], [32], and [37].

Theorem 3.2. *For matrices A, U, and V of sizes $m \times n$, $m \times r$, and $n \times r$, respectively, such that $\min\{m, n\} > r$, let the matrix $C = A + UV^H$ have full rank and write $G = I_r - V^HC^+U$. Then $V^HC^+A = GV^H$ if $m \geq n$ and $AC^+U = UG$ if $m \leq n$.*

Proof.

$$V^HC^+A = V^HC^+(C - UV^H) = V^H - V^HC^+UV^H = GV^H \text{ if } m \geq n,$$

$$AC^+U = (C - UV^H)C^+U = U - UV^HC^+U = UG \text{ if } m \leq n.$$

□

Corollary 3.2. *Under the assumptions of Theorem 3.2, let $m = n$. Then $G = U^+AC^+U$ (and therefore $N(AC^+U) = N(G)$) if the matrix U has full rank, whereas $G = V^HC^+AV^{+H}$ (and therefore $LN(V^HC^+A) = LN(G)$) if the matrix V has full rank.*

3.3 From rectangular to square inputs

Our nmb algorithms are simpler and tend to be more stable numerically in the case of square input matrices. We can yield a Hermitian square input as long as we agree to square the condition number because $N(A) = N(A^HA)$ and $\operatorname{cond}(A^HA) = (\operatorname{cond} A)^2$.

Given an $m \times n$ matrix A for $m > n$, we can first represent it as the sum $A = \sum_{i=1}^{h} A_i$ where $A_i = (0, B_i, 0)^T$ and B_i are $m \times m$ matrices for $i = 1, \ldots, h$. Then we observe that $N(A) = \cap_{i=1}^{h} N(B_i)$. Furthermore, we can simplify the computation of the null space intersection due to the following result [17, Theorem 12.4.1].

Theorem 3.3. *Let Z be a unitary nmb for an $m \times n$ matrix A and let W be a unitary nmb(BZ) where B is a $p \times n$ matrix. Then ZW is a unitary matrix basis for the linear space $N(A) \cap N(B)$.*

If $m < n$ we can reduce our null space problem to the case of an $n \times n$ matrix based on the following simple fact.

Fact 3.1. *We have $N(A) = N(B^HA)$ for a pair of $m \times n$ matrices A and B where $m \leq n$ and B is a full rank matrix.*

For a unitary matrix B, the matrices A and B^HA share their SVDs. For $B = (I_m, 0)$, the transition $A \to B^TA$ means just appending the $n - m$ rows of zeros at the bottom of the matrix A.

3.4 Random APPs

We begin with a basic lemma and then recall some results from [34], [35] that show the power of random APPs.

Lemma 3.1. *[13] (cf. also [42], [51]). For a finite set Δ of cardinality $|\Delta|$, let a polynomial in m variables have total degree d, let it not vanish identically on the set Δ^m, and let the values of its variables be randomly sampled from the set Δ. Then the polynomial vanishes with a probability of at most $d/|\Delta|$.*

Theorem 3.4. *For positive integers m and $n \le m$, a finite set Δ of cardinality $|\Delta|$ in a ring $\mathbb{R}$ and four matrices $A \in \mathbb{R}^{m\times n}$ of a rank ρ, U in $\Delta^{r\times m}$, V in $\Delta^{r\times n}$, and $C = A + UV^T$, we have*

a) $\operatorname{rank} C \le r + \rho$,

b) $\operatorname{rank} C = n$ *with a probability of at least $1 - \frac{2r}{|\Delta|}$ if $r + \rho \ge n$ and either the entries of both matrices U and V have been randomly sampled from the set Δ or $U = V$ and the entries of the matrix U have been randomly sampled from this set,*

c) $\operatorname{rank} C = n$ *with a probability of at least $1 - \frac{r}{|\Delta|}$ if $r + \rho \ge n$, a fixed matrix U (respectively V) has full rank r, and the entries of the matrix V (respectively U) have been randomly sampled from the set Δ.*

The theorem is a simple corollary of Lemma 3.1 and shows that the transition $A \to C = A + UV^H$ is likely to fix the rank deficiency in the case of random generators U and V of rank $r = \operatorname{nul} A$ if the cardinality $|\Delta|$ is large enough.

Next we estimate the impact of such a transition onto $\operatorname{cond} A$.

Theorem 3.5. *[34], [35]. Let $A = S\Sigma T^H$ be the SVD of the matrix A, where S and T are $n \times n$ unitary matrices, so that $S^H S = T^H T = I_n$, $\Sigma = \operatorname{diag}(\Sigma_A, 0_r)$ is an $n \times n$ diagonal matrix of a rank $\rho = n - r$, and $\Sigma_A = \operatorname{diag}(\sigma_j)_{j=1}^{\rho}$ is the diagonal matrix of the positive singular values of the matrix A. Let U and V be $n \times r$ matrices such that the $n \times n$ matrix $C = A + UV^H$ is nonsingular. Write*

$$S^H U = \begin{pmatrix} U_\rho \\ U_r \end{pmatrix},\ T^H V = \begin{pmatrix} V_\rho \\ V_r \end{pmatrix},\ R_U = \begin{pmatrix} I_\rho & U_\rho \\ 0 & U_r \end{pmatrix},\ R_V = \begin{pmatrix} I_\rho & V_\rho \\ 0 & V_r \end{pmatrix}$$

where U_r and V_r are $r \times r$ block submatrices. Then
a) $C = SR_U \operatorname{diag}(\Sigma_A, I_r) R_V^H T^H$ and
b) the matrices R_U, R_V, U_r, and V_r are nonsingular.

Corollary 3.3. *Write $\theta = \frac{||UV^H||}{||A||}$, $q = ||R_U||\ ||R_V||$ and $p = ||R_U^{-1}||\ ||R_V^{-1}||$, so that*

$$\max\{1, ||U||, ||V||, ||U||\ ||V||\} \le q \le \sqrt{(1 + ||U||^2)(1 + ||V||^2)},$$

$$1 \le p^2 \le (1 + (1 + ||U||^2)||U_r^{-1}||^2)(1 + (1 + ||V||^2)||V_r^{-1}||^2).$$

Suppose $\sigma_{n-r} \le 1 \le \sigma_1$. Then under the assumptions of Theorem 3.5 we have

a) $\max\{|1-\theta|,\ \frac{1}{p}\} \le \frac{||C||}{||A||} \le \min\{1+\theta, q\}$,

b) $\frac{1}{q} \le \sigma_{n-r}||C^{-1}|| = ||\frac{C^{-1}||}{||A^+||} \le p$,

c) $\frac{1}{q}\max\{|1-\theta|,\ \frac{1}{p}\} \le \frac{\operatorname{cond} C}{\operatorname{cond} A} \le p\ \min\{1+\theta, q\}$.

Clearly we can nicely bound the parameters θ and q from above and below by properly scaling the matrices A, U and V. A little later we will argue that the upper bound p on the ratio $\frac{||C^{-1}||}{||A^+||}$ is itself expected to be reasonably bounded on the average pair of U and V.

Now suppose A is a singular matrix, $A + UV^H$ is its nonsingular A-modification, and both matrices A and $A + UV^H$ are well conditioned. Then, clearly, the A-modification with the same APP UV^H transforms all nonsingular ill conditioned matrices $A + E$ in a sufficiently small neighborhood of the matrix A into nonsingular well conditioned matrices $A + E + UV^H$. Let us supply the respective estimates.

Hereafter $M \ge 0$ means that M is a nonnegative definite Hermitian matrix. We extend the bounds of Corollary 3.3 in the cases where $||E||$ is small or $C \ge 0$ and $E \ge 0$.

Theorem 3.6. *Under the assumptions of Corollary 3.3 let the matrices C and $\tilde{C} = C + E$ be nonsingular. Write $\delta = ||E||$ and $\delta_C = \delta||C^{-1}||$. Then we have*

a) $||\tilde{C}|| \le \delta + ||C||$,

b) if $\delta_C < 1$, *then* $||\tilde{C}^{-1}|| \le \frac{||C^{-1}||}{1-\delta_C}$, *so that*
$\operatorname{cond} \tilde{C} \le \frac{\operatorname{cond} C + \delta}{1-\delta_C}$,

c) if $C \ge 0$ *and* $E \ge 0$, *then* $||\tilde{C}^{-1}|| \le ||C^{-1}||$, *so that*
$\operatorname{cond} \tilde{C} \le (1 + \frac{\delta}{||C||})\operatorname{cond} C$.

Now recall that a number of theoretical results (cf. [14], [45], [46], [48]) and huge empirical evidence show that random matrices tend to be well conditioned under the customary probability distributions. It follows that the upper bound $p = ||R_U^{-1}||\ ||R_V^{-1}||$ on the ratio $\frac{||A^{-1}||}{||C^{-1}||}$ is likely to be reasonable for random matrices U_r and V_r. The matrices $S^H U$ and $T^H V$ and consequently their blocks U_r and V_r can be viewed as random as long as we generate the matrices U and V independently of the matrices S and T (of the singular vectors of the matrix A) and use some randomization, even with a smaller number of random parameters.

These weak requirements seem to be met routinely. The paper [34] reports the results of extensive tests, performed for a variety of ill conditioned matrices A, where the randomness of the matrices U and V was restricted by various patterns of sparseness and structure. In these tests the resulting weakly random APPs UV^H regularly remained effective preconditioners.

In particular, this was the case for the sparse and structured generators and Hermitian APPs in Example 3.1 below (cf. [34, Example 4.6], [35, Example 6]), even where these APPs were further restricted to the primitive case in which $P = I$, $T_i = c_i I$ for all i, and random parameters c_i were sampled from the set $\{-2, -1, 1, 2\}$ with the subsequent normalization of the APP UU^T by scaling.

Example 3.1. Structured and sparse Hermitian APPs. *Let $k, n_1, \dots, n_k$ be positive integers (fixed or random) such that $kr+n_1+\cdots+n_k = n$. For $i = 1, \dots, k$, let $0_{r,n_i}$ denote the $r \times n_i$ matrices filled with zeros and let T_i denote some $r \times r$ (fixed or random) structured or sparse well conditioned matrices, e.g., the matrices of the discrete Fourier, sign or cosine transforms, matrices with a fixed displacement structure (e.g., Toeplitz, triangular Toeplitz, circulant, Hankel, or Cauchy matrices), sparse matrices with fixed patterns of sparseness, or in the simplest case just the scaled identity matrices $c_i I_r$ (cf. [10], [11], [24], [30], [38], and the bibliography therein). Let $U = P(T_1, 0_{r,n_1}, \dots, T_k, 0_{r,n_k})^T$. Choose an $n \times n$ permutation matrix P (in the simplest case let $P = I$) and define the APP UU^H.*

Remark 3.1. *Assume an $n \times n$ matrix A and an $n \times n$ weakly random APP P of a rank $r < n$ such that the ratio $\frac{||A||}{||P||}$ is neither large nor small. Define the matrix $C = A + P$. Then according to the above study and extensive tests in [34], $\operatorname{cond} C$ is likely to have the order $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$. In particular if in Example 1.1 the vector $\mathbf{b}$ is random and independent of the SVD of the matrix A and if the scaled ratio $\frac{||\mathbf{b}||}{||A||}$ is neither large nor small, then the condition number of the matrix $M = (-\mathbf{b}, A)$ is likely to be of the order of $\frac{\sigma_1(A)}{\sigma_{n-1}(A)}$.*

Lemma 3.2. *Let $C = A + UV^H$ where the matrices A, C, V, and U have sizes $m \times n$, $m \times n$, $n \times r$, and $m \times r$, respectively, $U \neq 0$, $m \geq n > r > 0$, $\rho = \operatorname{rank} A$, $n \leq r + \rho \leq m$, and the entries of the matrix U have been sampled at random from a finite set Δ. Then*

a) the matrix equation $CY = U$ has a solution Y with a probability of at most $\frac{r}{|\Delta|}$ and

b) the matrix C is rank deficient with a probability of at most $\frac{2r}{|\Delta|}$.

Proof. a) Clearly, the matrix A has an $m \times \rho$ submatrix A_ρ of the full rank ρ. The equation $CY = (A + UV^H)Y = U$ implies that $AY = U(I_r - V^H Y)$. Therefore the set $((\operatorname{range} A) \cap \operatorname{range} U) - \{0\}$ is not empty, and so the matrix (A_ρ, U) is rank deficient. Since $\rho + r \leq m$, it follows that the matrix U is rank deficient. Lemma 3.1 implies, however, that the determinant of any $r \times r$ submatrix of this matrix vanishes with a probability of at most $\frac{r}{|\Delta|}$, and so part a) follows.

b) Consider the entries of the matrix $C = A + UV^H$ as bilinear functions in the entries of the matrices U and V. Then, since $\rho + r \geq n$, the matrix C has full rank n. Let $C_n = C_n(U, V)$ denote its $n \times n$ submatrix that has full rank n. Then $\det C_n$ is a nonvanishing polynomial of the total degree of at most $2r$ in the entries of the matrices U and V. This polynomial vanishes where the matrix C is rank deficient. Now part b) follows from Lemma 3.1. □

3.5 Perturbation and error estimates

Next we assume computations in the field of real numbers and recall some estimates for the errors and perturbations (cf. [26] and [49, page 447] on the

extension to computations in the complex field). We list the estimates for the general rectangular input, but in view of Section 3.2 the reader may focus just on the simpler estimates for the square inputs and omit the respective results on the least squares computations.

Hereafter $fl(W) = fl_u(W)$ denotes the result of floating point computation of the matrix or vector W by a fixed algorithm assuming the unit roundoff u (also called machine epsilon), and we write $\gamma_n = \frac{nu}{1-nu}$.

For a matrix $A = (a_{ij})_{ij}$ we write $|A| = (|a_{ij}|)_{ij}$, so that $A \geq 0$ if $A = |A|$. We recall that $||A||_l = ||\ |A|\ ||_l$ for $l = 1, \infty, F$ and $|A| \leq B \rightarrow ||A|| \leq ||B||$.

Theorem 3.7. *(Cf. [18, Section 3.5].) For a pair of $n \times n$ matrices A and B and a vector $\mathbf{v}$ of dimension n, we have $||fl(A\mathbf{v}) - A\mathbf{v}||_l \leq \gamma_n ||A||_l ||\mathbf{v}||_l$, $l = 1, \infty$, and $||fl(AB) - AB||_l \leq \gamma_n ||A||_l ||B||_l$, $l = 1, 2, F$, assuming classical matrix multiplication.*

The following basic perturbation estimate enables standard extension of the backward error bounds to relative error bounds for the computed solution or least-squares solution of a linear system of equations. This bound is behind most of the error estimates in this subsection.

Theorem 3.8. *(Cf. [18, Section 7.1, page 121].) Let $A\mathbf{x} = \mathbf{v}$ and $(A + \Delta(A))\tilde{\mathbf{x}} = \mathbf{v} + \Delta(\mathbf{v})$ for a pair of nonsigular matrices A and $A + \Delta(A)$ and two vectors $\mathbf{v}$ and $\Delta(\mathbf{v})$ such that $||\Delta(A)||_l \leq \epsilon ||A||_l$, $||\Delta(\mathbf{v})||_l \leq \epsilon ||\mathbf{v}||_l$ for $l = 1, 2, \infty$ and $\epsilon \operatorname{cond}_l A < 1$. Then $\frac{||(\tilde{\mathbf{x}} - \mathbf{x})||_l}{||\tilde{\mathbf{x}}||_l} \leq 2\epsilon \frac{\operatorname{cond}_l A}{1 - \epsilon \operatorname{cond}_l A}$.*

We also recall some error estimates for the floating point comutation of the solutions and least-squares solutions to linear sytems of equations, which are more favorable in the former case of square input matrices.

Theorem 3.9. *(Cf. [18, Theorem 8.5], [43, Section 3.4.2, equation (4.5)].) For an $n \times n$ nonsingular triangular matrix T, let the floating point solution $\tilde{\mathbf{x}} = fl(\mathbf{x})$ to the linear system $T\mathbf{x} = \mathbf{v}$ be computed by means of substitution with any ordering (in n^2 flops). Then $(T + \Delta(T))\tilde{\mathbf{x}} = \mathbf{v}$, $|\Delta(T)| \leq \gamma_n |T|$. Furthermore we have $\frac{||\tilde{\mathbf{x}} - \mathbf{x}||_l}{||\tilde{\mathbf{x}}} ||_l \leq 1.12nu \operatorname{cond}_l T$ for $l = 1, \infty, F$.*

Theorem 3.10. *(Cf. [18, Theorems 9.3 and 9.4], [43, Theorem 3.4.9].)*

a) Suppose Gaussian elimination (with or without pivoting) applied to an $m \times n$ matrix A runs to completion (by using $(m - n/3)n^2 + O(mn)$ flops) and outputs triangular factors $\tilde{L}$ of size $m \times n$ and $\tilde{U}$ of size $n \times n$. Then $\tilde{L}\tilde{U} = A + \Delta(A)$, $|\Delta(A)| \leq \gamma_n |\tilde{L}|\ |\tilde{U}|$.

b) If $m = n$ and the computation produces floating point solution $\tilde{\mathbf{x}} = fl(\mathbf{x})$ to a linear system $A\mathbf{x} = \mathbf{v}$, then $(A + \Delta(A))\tilde{\mathbf{x}} = \mathbf{v}$, $|\Delta(A)| \leq 3\gamma_n |\tilde{L}|\ |\tilde{U}|$. Furthermore $\frac{||A(\tilde{\mathbf{x}} - \mathbf{x})||_l}{||\tilde{\mathbf{x}}||_l} \leq 1.12(3 + 1.12nu)nu||L||_l||U||_l$ for $l = 1, \infty, F$.

For Gaussian elimination with rook and complete pivoting (which use about $2n^2$ to $\frac{1}{3}n^3$ comparisons) the growth factor $g(A) = \frac{||L||_l ||U||_l}{||A||_l}$ is bounded by functions in n that grow not very rapidly. Even with partial pivoting (which uses $(n-1)n/2$ comparisons) we always have $||L||_l \leq 1$, whereas according to empirical evidence the norm $||U||_l$ usually has order 10 [17, page 116].

Theorem 3.11. *(Cf. [18, Theorems 10.3, 10.5, and 10.6].)*

a) Suppose Cholesky factorization applied to a real symmetric and positive definite $n \times n$ matrix $A = (a_{ij})_{i,j=1}^n$ runs to completion (by using $\frac{1}{3}n^3 + O(n^2)$ flops) and outputs $n \times n$ triangular factor $\tilde{R}$. Then $\tilde{R}\tilde{R}^T = A + \Delta(A)$ where $|\Delta(A)| \le \gamma_{n+1}|\tilde{R}^T|\ |\tilde{R}|$.

b) If the computation is extended to produce floating point solution $\tilde{\mathbf{x}} = fl(\mathbf{x})$ to a linear system $A\mathbf{x} = \mathbf{v}$, then $(A + \Delta(A))\tilde{\mathbf{x}} = \mathbf{v}$, $|\Delta(A)| \le \gamma_{3n+1}|\tilde{R}^T|\ |\tilde{R}|$ and $|\Delta(A)| \le \frac{\gamma_{n+1}}{1-\gamma_{n+1}} \sum_{i=1}^n a_{ii}$. Furthermore

$$\frac{||D(\tilde{\mathbf{x}} - \mathbf{x}||}{||D\mathbf{x}||} \le \frac{\epsilon \operatorname{cond} H}{1 - \epsilon \operatorname{cond} H}$$

where $D^2 = \operatorname{diag}(a_{ii})_i$, $A = DHD$, $\epsilon = n\frac{\gamma_{3n+1}}{1-\gamma_{n+1}}$, and $\epsilon \operatorname{cond} H < 1$.

Theorem 3.12. *a) Suppose the Householder QR algorithm applied numerically with rounding to a nonsingular $m \times n$ matrix $A = (\mathbf{a}_j)_{j=1}^n$, $m \ge n$ (performs $2(m - \frac{n}{3})n^2 + O(mn)$ flops and) outputs $m \times n$ trapezoidal factor $\tilde{R} = fl(R)$. Then there exists an $m \times m$ unitary matrix Q such that $Q\tilde{R} = A + \Delta_1(A)$ where $\Delta_1(A) = (\Delta(\mathbf{a}_j))_{j=1}^n$ and $||\Delta(\mathbf{a}_j)|| \le \gamma_{cn^2}||\mathbf{a}_j||$ for $j = 1, \ldots, n$ and a scalar c (cf. [18, Theorem 19.4]).*

b) If the latter computations with rounding are extended to computing floating point least squares solution $\tilde{\mathbf{x}} = fl(\mathbf{x})$ to a linear system $A\mathbf{x} = \mathbf{v}$, then this is the exact least squares solution to the linear system $(A + \Delta(A))\tilde{\mathbf{x}} = \mathbf{v} + \Delta(\mathbf{v})$ where $||\Delta(\mathbf{v})|| \le \gamma_{cn^2}||\mathbf{v}||$ and $||\Delta(\mathbf{a})_j|| \le \gamma_{cn^2}||\mathbf{a}_j||$ for $j = 1, \ldots, n$ and a scalar c (cf. [18, Theorem 19.5]), and furthermore

c) $||\Delta(A)||_F \le (6m - 3n + 41)nu||A||_F + O(u^2)$, $||\Delta(\mathbf{v})|| \le (6m - 3n + 40)nu||\mathbf{v}|| + O(u^2)$ (cf. [22, Chapter 16]).

Theorem 3.13. *(Cf. [18, Theorem 19.13].) Suppose the Modified Gram–Schmidt QR algorithm applied to an $m \times n$ matrix A of rank $n \le m$ (performs $2mn^2 + O(mn)$ flops and) computes with rounding the factors $\tilde{Q} = fl(Q)$ of the size $m \times n$ and $\tilde{R} = fl(R)$ of the size $n \times n$. Then there exists a unitary matrix Q and scalar parameters c_1, c_2, c_3, and c_4 depending on m and n such that $\tilde{Q}\tilde{R} = A + \Delta_1(A)$ and $Q\tilde{R} = A + \Delta_2(A)$, $||\Delta_1(A)|| \le c_1 u||A||$, $||\tilde{Q}^T\tilde{Q} - I|| \le c_2 u \operatorname{cond} A + O((u \operatorname{cond} A)^2)$, and $||\Delta_2(\mathbf{a}_j)|| \le c_3 u||\mathbf{a}_j||$, $j = 1, \ldots, n$.*

Theorem 3.14. *(Cf. [18, Section 20.2, page 385].) Suppose that least-squares solution $\tilde{\mathbf{x}} = fl(\mathbf{x})$ to a linear system $A\mathbf{x} = \mathbf{v}$ of m equations with n unknowns is computed (in $2(m - n/3)n^2 + O(mn)$ flops) by the Householder algorithm applied with rounding where A is an $m \times n$ matrix A of full rank $n \le m$. Then there is a constant c such that*

$$||A\tilde{\mathbf{x}} - \mathbf{v}|| \le m\gamma_{cmn}||\ \ |A\mathbf{x}| + |\mathbf{v}|\ \ || + (1 + m\gamma_{cmn} \operatorname{cond} A^T)||A\mathbf{x} - \mathbf{v}|| + O(u^2).$$

The latter estimate is readily extended to bound the error norm

$$||\tilde{\mathbf{x}} - \mathbf{x}|| = ||\tilde{\mathbf{x}} - A^+\mathbf{v}|| \le ||A^+||\ ||A\tilde{\mathbf{x}} - \mathbf{v}||. \tag{3.7}$$

In 1967 Å. Björk deduced similar estimates for the solution computed in $2mn^2 + O(mn)$ flops based on the Modified Gram-Schmidt algorithm instead of Householder's (cf. [3], [18, Section 20.3]).

By using the normal or corrected seminormal equations, one can obtain the solution in $(m + n/3)n^2 + O(mn)$ flops within the error norm bound $||\tilde{\mathbf{x}} - \mathbf{x}||$ of the orders

$$c_{m,n}(\operatorname{cond} A)^2 u||\mathbf{x}|| \tag{3.8}$$

for a scalar $c_{m,n}$ depending on m and n [18, Section 20.4] and

$$(2n^{1/2}(c_1 + 2n + m/2)(2n^{1/2}(c_1 + n)||\mathbf{x}|| + n^{1/2}m\frac{||\mathbf{v}||}{||A||})(\operatorname{cond} A)^3u^2 +$$

$$(n^{1/2}u \operatorname{cond} A)\ (n||\mathbf{x}|| + m \operatorname{cond} A\frac{||\mathbf{v} - A\mathbf{x}||}{||A||}) + n^{1/2}u||\mathbf{x}||)(1 + O(u \operatorname{cond} A))$$

provided $c_1 = (6m - 3n + 40)n + O(u)$ and $c_1 n^{1/2} u \operatorname{cond} A < 1$ (cf. [2]), respectively.

We also recall the following perturbation estimates by Wedin 1973 (cf. [18, Theorem 20.1]).

Theorem 3.15. *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *denote the least-squares solutions to two linear systems* $A\mathbf{x} = \mathbf{v}$ *and* $(A + \Delta(A))\mathbf{y} = \mathbf{v}$ *where* $||\Delta(A)|| \le \epsilon||A||$*, both* $m \times n$ *matrices* A *and* $A + \Delta(A)$ *have full rank* $n \le m$*, and* $\epsilon \operatorname{cond} A < 1$*. Then* $\frac{||\mathbf{x}-\mathbf{y}||}{||\mathbf{x}||} \le \frac{\epsilon \operatorname{cond} A}{1-\epsilon \operatorname{cond} A}(2 + (\operatorname{cond} A + 1)\frac{||\mathbf{v}-A\mathbf{x}||}{||A||\ ||\mathbf{x}||})$.

4 Null bases via A-preprocessing

4.1 The auxiliary least squares computations

One of our basic steps is the computation of an $m \times r$ least squares solution $Y = C^+U$ to the linear matrix equation $CY = U$ where $C = A + UV^H$ is an $m \times n$ matrix of full rank. There is a variety of effective numerical methods for this task [4], [17, Section 5.3], [18, Chapter 20], [22], [43, Section 4.2], but we can also simplify it by reducing it to the case where $m = n$, $C^+ = C^{-1}$ (see Section 3.3), and any method for linear systems of equations can be applied to computing the solution $Y = C^{-1}U$ of the matrix equation $CY = U$. In spite of the equations $\operatorname{cond}(C^HC) = \operatorname{cond}(CC^H) = (\operatorname{cond} C)^2$, the symmetrizations $C \to C^HC$ and $C \to CC^H$ (leading to normal and corrected seminormal equations) are quite competitive for general well conditioned matrices C. ("The most widely used method for solving the full rank LS problem is the method of normal equations" [17, page 238], and "it is safe to say that the majority – a great majority – of least squares problems are solved by forming and solving the normal equations" [43, page 298].)

4.2 Error-free computations of nmbs

We assume error-free computations in this subsection and cover the extension to numerical computations with rounding in the next subsection. We begin with a simpler case where we are given the nullity of the input matrix.

Algorithm 4.1. Computing an nmb given the nullity.

INPUT: *three integers m and n, a small positive δ, an $m \times n$ matrix A, and the integer $r = \operatorname{nul} A$ where $m \geq n > r \geq 0$.*

OUTPUT: *either FAILURE with a probability of at most δ or an* $\operatorname{nmb}(A)$.

INITIALIZATION: *Set $k \leftarrow 1$ and fix a smaller positive integer ν (say, $\nu = 1$ or $\nu = 2$) and a sufficiently large set Δ of real or complex numbers such that $2\nu/|\Delta| \leq \delta$.*

COMPUTATIONS:

1. *Randomly sample from the set Δ the entries of two matrices, U of the size $m \times r$ and V of the size $n \times r$.*
2. *Compute the matrix $C \leftarrow C + UV^H$. If this matrix C is rank deficient, then either output FAILURE and stop if $k > \nu$ or otherwise set $k \leftarrow k + 1$ and go to Stage 1.*
3. *Compute the matrix C^+U. Compute and output a matrix basis for its range and stop.*

Unless it fails, the algorithm produces an nmb due to Theorem 3.1. The algorithm invokes Stage 2 at most ν times. In each invocation, it fails with a probability of at most $\frac{2}{|\Delta|}$ due to Theorem 3.4b for $r = 1$, that is, the overall probability of failure is at most $\frac{2\nu}{|\Delta|} \leq \delta$. This proves correctness of the algorithm.

If the nullity $\operatorname{nul} A$ is unknown, we can recall Theorems 3.1 and 3.4 and compute it as

i) the minimum integer r such that the matrix $C = A + UV^H$ has full rank for some APP UV^H of rank r,

ii) the minimum integer r such that the matrix $C = A + UV^H$ is likely to have full rank for a random APP UV^H of rank r,

iii) the rank of an APP UV^H such that the matrix $C = A + UV^H$ has full rank and $AC^+U = 0$,

iv) the rank of an APP UV^H such that the matrix $C = A + UV^H$ has full rank and the matrix $AC^+U\mathbf{x}$ is likely to vanish for a random vector $\mathbf{x}$.

We can assume some initial range $[r_-, r_+]$ for the nullity $r = \operatorname{nul} A$ such that $0 \leq r_- \leq r \leq r_+ \leq n - 1$ and $r_- < r_+$. Then we can generate weakly random matrices U of the size $m \times i$ and V of the size $n \times i$ for i changing in this range until we arrive at a matrix $C = A + UV^H$ of full rank and such that $AC^+U = 0$. It remains to choose a policy of the search for the nullity

r in this range. We specify two algorithms that perform linear (sequential) search based on properties i) and iii) above, where we successively choose $i = r_-, r_- + 1, \dots, r$ or $i = r_+, r_+ - 1, \dots, r$, respectively (see Remark 4.2 and Algorithm 4.4 on the acceleration of the search). One can modify the algorithms by adding randomization and relying on properties ii) and iv), respectively.

Our next algorithms employ a black box Subroutine FULL·RANK that tests whether a given matrix has full rank (cf. Remark 4.6).

Algorithm 4.2. An nmb via the nullity search from below.

INPUT: *four integers m, n, r_-, and r_+ such that $m \geq n > r_+ > r_- \geq 0$, a small positive δ, an $m \times n$ matrix A such that $r_+ \geq r = \operatorname{nul} A \geq r_-$, and Subroutine FULL·RANK.*

OUTPUT: *either FAILURE with a probability of at most δ or the integer $r = \operatorname{nul} A$ and an $\operatorname{nmb}(A)$.*

INITIALIZATION: *Set $q \leftarrow r_-$. Fix a sufficiently large set Δ of real or complex numbers such that $2\frac{r_+ + 1}{|\Delta|} \leq \delta$. Sample from this set the entries of weakly random matrices U of the size $m \times q$ and V of the size $n \times q$. Compute the matrix $C = A + UV^H$. (If $r_- = 0$, then U and V are the dummy empty matrices of the sizes $m \times 0$ and $n \times 0$, respectively, and $C = A$.)*

COMPUTATIONS:

1. *Apply the Subroutine FULL·RANK to the matrix C.*
2. *If the matrix C is rank deficient, then either output FAILURE and stop if $q > r_+$ or otherwise randomly sample the entries of two vectors $\mathbf{u} = (u_i)_{i=1}^m$ and $\mathbf{v} = (v_i)_{i=1}^n$ from the set Δ, set $C \leftarrow C + \mathbf{u}\mathbf{v}^H$, $U \leftarrow (U, \mathbf{u})$, $V \leftarrow (V, \mathbf{v})$ (cf. Remark 4.1), and $q \leftarrow q + 1$, and go to Stage 1.*
3. *If the matrix C has full rank, compute the matrices C^+U and AC^+U. If $AC^+U \neq 0$, output FAILURE. Otherwise compute and output an $n \times r$ matrix basis B for the range of the matrix C^+U, where $r \leq q$. (B is the $n \times 0$ empty matrix if $r = 0$.) Output the integer r and stop.*

$r - r_- + 1$ times the algorithm invokes Stage 1 and computes the matrix C. It uses $(m+n)r$ random parameters for a general APP UV^H, but like all other our algorithms, it uses much fewer parameters for a sparse or structured APP (cf. Example 3.1).

$\operatorname{rank} C = \operatorname{rank} A + r_-$ at the initialization stage with a probability of at least $1 - 2\frac{r_-}{|\Delta|}$, due to Theorem 3.4b for $r = r_-$, whereas every recomputation of the matrix C (at Stage 2) increases its rank with a probability of at least $1 - \frac{2}{|\Delta|}$, due to Theorem 3.4b for $r = 1$. This means a probability of at least $(1 - \frac{2}{|\Delta|})^{r_+ - r_- + 1}(1 - 2\frac{r_-}{|\Delta|}) > 1 - 2\frac{r_+ + 1}{|\Delta|} \geq 1 - \delta$ that $\operatorname{rank} C = n$ after all updatings of the matrix C. In this case the algorithm produces correct output

in virtue of Theorem 3.1. Correctness of this output is certified at Stage 3, when we test whether $AC^+U = 0$. Otherwise, with a probability of at most δ, the algorithm outputs FAILURE (and so it works as we claimed).

Remark 4.1. *The computation of the matrix V at Stage 2 can be omitted, but if $m \le n$ and the matrices C, U and V have full ranks, we can compute the matrix V^HC^+ and test whether it is a left nmb(A) by testing whether $V^HC^+A = 0$. If $m = n$, we can recall Corollary 3.2 and modify Algorithm 4.2 and the latter extension to compute both left and right nmbs by using the matrix $G = I_r - V^HC^+U$ instead of the matrices AC^+U and V^HC^+A. This remark can be immediately extended to our next algorithms.*

Algorithm 4.2 tests property i) of the nullity nul A where $C = A+UV^H$ and the APPs UV^H have rank recursively increasing from r_-. Our next algorithm tests property iii) for the matrices $C = A + UV^H$ where the APPs UV^H have ranks recursively decreasing from r_+.

Algorithm 4.3. An nmb via the nullity search from above.

INPUT *and* OUTPUT *as in Algorithm 4.2.*

INITIALIZATION: *Set $q \leftarrow r_+$. Fix a sufficiently large set Δ of real or complex numbers such that $\frac{4r_+-2r_-}{|\Delta|} \le \delta$. Sample from this set the entries of (weakly) random matrices U of size $m \times q$ and V of size $n \times q$ and compute the matrix $C = A + UV^H$.*

COMPUTATIONS:

1. *Apply Subroutine FULL·RANK to the matrix C. If the matrix is rank deficient, output FAILURE and stop. Otherwise compute the matrices C^+U and AC^+U.*
2. *If $AC^+U = 0$, compute and output an $n \times r$ matrix basis B for the range of the matrix C^+U, output the integer r, and stop. (B is the $n \times 0$ empty matrix if $r = 0$.)*
3. *Otherwise if $q = r_-$, output FAILURE and stop. If $q > r_-$, update the matrices U and V by removing their last columns $\mathbf{u}$ and $\mathbf{v}$, respectively. Update the matrix C by setting $C \leftarrow C - \mathbf{u}\mathbf{v}^H$, set $q \leftarrow q-1$, and go to Stage 1.*

The algorithm uses $(m+n)r_+$ random parameters. $r_+ - r + 1$ times it invokes Stage 1 and thus $r_+ - r$ times updates the matrix C.

Unless it fails, the algorithm computes correct output due to Theorem 3.1. With a probability of at least $1 - \frac{2r_+}{|\Delta|}$ the initialization produces a matrix C of full rank n, due to Theorem 3.4b for $r = r_+$. With every update of the matrix C at Stage 3 its rank decreases with a probability of at least $\frac{2}{|\Delta|}$, due to Theorem 3.4b for $r = 1$. This means a probability of at least $(1 - \frac{2}{|\Delta|})^{r_+-r_-}$ that the rank decreases in each of the $r_+ - r$ updatings. Therefore the algorithm

produces correct output with a probability of at least $(1-\frac{2}{|\Delta|})^{r_+-r}(1-2\frac{r_+}{|\Delta|}) > 1-2\frac{2r_+-r_-}{|\Delta|} \geq 1-\delta$. (Correctness is certified at Stage 2 when we test whether $AC^+U = 0$.) Otherwise the algorithm outputs FAILURE (and so it works as we claimed). The algorithm outputs FAILURE only if it encounters a rank deficient matrix C, but the algorithm is likely to avoid dealing with rank deficient matrices at all where the cardinality $|\Delta|$ is large. The latter property also holds for our next algorithm.

Remark 4.2. *Both Algorithms 4.2 and 4.3 compute the nullity by means of the linear (sequential) search in the range $[r_-, r_+]$ based on properties i)–iv) of the nullity. We can achieve acceleration by applying binary search. Furthermore whenever we update the matrix C by adding a matrix of a rank h, we only need $O(mnh)$ flops to update also the inverse C^{-1} (by applying the Sherman–-Morrison–Woodbury formula [17, page 50]) as well as the QR factorization of the matrix C [17, Section 12.5.1]. Our next algorithm, based on Corollary 3.1, demonstrates yet another acceleration technique: it applies aggregation to compute the nullity and an nmb(A).*

Algorithm 4.4. An nmb via aggregation *(see Remark 4.3).*

INPUT, OUTPUT, INITIALIZATION *and Stages 1 and 2 of* COMPUTATIONS *are as in Algorithm 4.3, except that the* INPUT *includes an additional positive tolerance $\tilde{\delta}$ and at the* INITIALIZATION *we require that $2\frac{r_+}{|\Delta|} \leq \delta - \tilde{\delta}$.*

COMPUTATIONS:

3. Otherwise observe that $r_- \leq \text{nul}(AC^+U) < q$ and apply the algorithm to the $m \times q$ matrix AC^+U, allowing FAILURE with a probability of at most $\tilde{\delta}$. Unless the algorithm fails, it outputs an nmb(AC^+U) (of the size $q \times r$). Write X to denote this nmb, output the integer r, compute and output an $n \times r$ matrix C^+UX (being an nmb(A)), and stop.

Correctness of the algorithm follows from Corollary 3.1.

Remark 4.3. *Unless $AC^+U = 0$ at Stage 2, the latter algorithm is a new instance in the general class of* aggregation methods. *They successively a) aggregate an input $\mathbb{I}$ into an input $\mathbb{I}_1$ of a smaller size, b) compute the solution Y_1 for a given task, but for the aggregated input $\mathbb{I}_1$, and c) disaggregate the aggregated solution Y_1 producing the solution Y for the original input $\mathbb{I}$. At Stage b) one can recursively reapply aggregation. For examples recall Schur Aggregation in [31], [32], and [37], the hierarchial aggregation processes in [25], which in the 1980s served as the springboard for Algebraic Multigrid, and trilinear aggregating in [19], [20], [23], [28]. In our case $\mathbb{I} = A$, $\mathbb{I}_1 = AC^+U$, $Y_1 = X$, and $Y = C^+UX$, and we call this technique the Null Aggregation. If $m = n$, we can choose $\mathbb{I}_1 = G = I_r - V^HC^+U$ (cf. Remark 4.1).*

Remark 4.4. *Computing the residual matrix AC^+U takes $(2n-1)mr$ flops if we are given the matrix C^+U, but if $m = n > 2r$ (so that $C^+ = C^{-1}$), we have $U(V^HC^{-1}U - I_r) = -AC^{-1}U$, and so we can compute just the matrix $G = V^HC^{-1}U - I_r$ by using $2(2r-1)nr + r$ flops.*

Remark 4.5. *To compute a single null vector, rather than a null basis, we can fix a vector* $\mathbf{c}$ *and compute the vector* $C^+U\mathbf{c}$ *instead of the matrix* C^+U *in Algorithms 4.1–4.4, and similarly we can simplify Stage 3 of Algorithm 4.4.*

Remark 4.6. *We can modify Algorithms 4.2–4.4 as follows. Test whether the matrix* C *is rank deficient implicitly, by trying to compute the matrix* C^+U*. If the computation fails, then the matrix* C *is definitely rank deficient. Otherwise it is likely to have full rank if* $m = n$ *and if the set* Δ *has large cardinality (cf. Lemma 3.2). Thus it is plausible (although unlikely for* $m = n$ *and large values* $|\Delta|$*) that skipping our applications of the Subroutine FULL·RANK could imply that the matrix* C^+U *defines only a submatrix* Y *of an nmb*(A)*, but we can readily detect this deficiency and repair it. We would just extend the nmb by applying the same algorithms to the matrix* $\begin{pmatrix} A \\ Y^H \end{pmatrix}$*. We also recall that the Subroutine FULL·RANK is used in Algorithms 4.3 and 4.4 only as a stopping criterion at Stage 1.*

4.3 Numerical computation of nmbs: our initial comments

Suppose our algorithms have been performed numerically, with rounding errors. Let $\mathrm{nmb}(A) + E$ denote the output matrix, so that E denotes the error matrix. Then typically $A(C^+U + E) \neq 0$ and the computed matrix $\tilde{C} = \mathrm{fl}(A + UV^H)$ has full rank even where $\operatorname{rank} U = \operatorname{rank} V < \operatorname{nul} A$. Thus we must modify Algorithms 4.1–4.4. Instead of testing whether the matrix C is rank deficient and whether $AC^+U = 0$, we apply two Subroutines ILL·CONDITIONED and NORM, respectively. For two fixed small tolerance values τ and t, they test whether the matrix C is ill conditioned.

$$\operatorname{cond} C > \frac{1}{\tau} \tag{4.1}$$

(which means that no rank deficient matrix approximates the matrix C within the norm bound $\tau||C||$) and whether the residual norm $||AB||$ is small enough, namely whether

$$||AB|| \leq t||A||\ ||B||. \tag{4.2}$$

We apply this test for the matrix $B = C^+U$ or $B = Q(C^+U)$ and use the 2-norm $||\cdot|| = ||\cdot||_2$ and $\operatorname{cond} C = \operatorname{cond}_2 C$. (One can readily adjust the algorithms to using l-norms $||\cdot||_l$ and $\operatorname{cond}_l C$ for $l = 1$ and $l = \infty$ instead.) We further comment on these subroutines and the tolerance bounds in Section 4.5.

Instead of Theorem 3.4, we rely on Theorems 3.5 and 3.6 and Corollary 3.3 supported by the extensive tests in [34] (cf. Section 3.4).

4.4 Numerical computation of nmbs: algorithms

Let us specify numerical versions of Algorithms 4.1–4.4 assuming general matrices A and C (cf. Section 6 on the case of structured inputs A).

Algorithm 4.5. Computing a numerical nmb given the nullity *(cf. Algorithm 4.1).*

INPUT: *three integers m, n, and r, an $m \times n$ matrix A of rank $n - r$, such that $m \geq n > r = \operatorname{nul} A$, a small positive tolerance t, and a Subroutine NORM.*

OUTPUT: *either FAILURE with a low probability or an $n \times r$ unitary matrix B such that $||AB|| \leq t||A||\ ||B||$.*

INITIALIZATION: *Fix a sufficiently large set Δ of real or complex numbers.*

COMPUTATIONS:

1. *Generate two weakly random matrices, U of size $m \times r$ and V of size $n \times r$, with the entries from the set Δ. Compute the matrices $U \leftarrow ||C||Q(U)$, $V \leftarrow Q(V)$, and $C \leftarrow C + UV^H$.*
2. *Compute the matrices C^+U, $B = Q(C^+U)$, and AB.*
3. *Apply the Subroutine NORM (recalling that in our case $||B|| = 1$). If bound (4.2) holds, output the unitary matrix B and stop. Otherwise output FAILURE and stop.*

Unless it fails, the algorithm verifies correctness of its output at Stage 3. The failure can occur for two reasons: a) because of an unlucky choice of the APP UV^H or b) because the precision of computing was too low to ensure the selected tolerance bound t on the residual norm. In the latter case we just need to perform the same algorithm with a higher precision.

As in the previous section and as we discussed earlier, the cause a), that is the unlucky choice of APPs, is unlikely. If we rerun Stages 1 and 2 using the same precision and observe about the same ratio $\frac{||AC^+U||}{||A||\ ||C^+U||}$, then this is an additional pointer to the cause b). If, however, this ratio stays about the same when we increase the precision of computing, then we have a strong pointer to the cause a). The same comments apply to our next algorithms.

In Algorithm 4.5 we assume that we are given the nullity $r = \operatorname{nul} A$. In our next numerical counterparts of Algorithms 4.2–4.4 we compute the nullity. In Algorithm 4.6 at least $r - r_-$ times we handle ill conditioned matrices C that satisfy bound (4.1), whereas Algorithm 4.7 is likely to involve no such matrices. In Algorithms 4.6 and 4.7 we can update the matrix C and its inverse at a lower cost based on the Sherman–Morrison–Woodbury formula or on updating the QR factorization (cf. Remark 4.2)).

In both algorithms we must solve the matrix equations $CW = U$ where $\operatorname{cond} C$ is expected to be of the order $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$ for $r = \operatorname{nul} A$. In our Algorithm 4.8, which extends Algorithm 4.4, we solve this equation only once, where $\operatorname{cond} C$ has the order $\frac{\sigma_1(A)}{\sigma_{n-r_+}(A)}$ and where we choose $r_+ \geq r = \operatorname{nul} A$.

Algorithm 4.6. A numerical nmb via the nullity search from below.

INPUT: *four integers m, n, r_-, and r_+ such that $m \geq n > r_+ \geq r_- \geq 0$, an $m \times n$ matrix A such that $r_+ \geq \operatorname{nul} A \geq r_-$, two small positive values t and τ, and two Subroutines NORM and ILL·CONDITIONED.*

OUTPUT: *either FAILURE or an integer r such that $r_+ \geq r \geq r_-$ and an $n \times r$ unitary matrix B such that $||AB|| \leq t||A||$.*

INITIALIZATION: *Set $r \leftarrow r_-$. Fix a sufficiently large set Δ of real or complex numbers. Sample from this set the entries of (weakly) random matrices U of size $m \times r$ and V of size $n \times r$, and compute the matrices $U \leftarrow ||C||Q(U)$, $V \leftarrow Q(V)$, and $C \leftarrow C + UV^H$. (If $r_- = 0$, then U and V are the empty matrices of the sizes $m \times 0$ and $n \times 0$, respectively, and $C = A$.)*

COMPUTATIONS:

1. *Apply the Subroutine ILL·CONDITIONED to the matrix C.*
2. *If $\operatorname{cond} C > \frac{1}{\tau}$, then either output FAILURE and stop if $r > r_+$ or, otherwise, randomly sample the entries of two vectors $\mathbf{u} = (u_i)_{i=1}^m$ and $\mathbf{v} = (v_i)_{i=1}^n$ from the set Δ, set $U \leftarrow (U, \mathbf{u})$ and $V \leftarrow (V, \mathbf{v})$ (cf. Remark 4.1), compute the matrices $U \leftarrow ||C||Q(U)$, $V \leftarrow Q(V)$, and $C + \mathbf{uv}^H$, set $C \rightarrow C + \mathbf{uv}^H$ and $r \leftarrow r + 1$, and go to Stage 1.*
3. *If $\operatorname{cond} C \leq \frac{1}{\tau}$, then compute the matrices C^+U, $B = Q(C^+U)$, and AB.*
4. *Apply the Subroutine NORM to the matrix AB. If bound (4.2) holds, output the integer r and the matrix B and stop. Otherwise output FAILURE and stop.*

Algorithm 4.7. A numerical nmb via the nullity search from above.

INPUT *and* OUTPUT *are as in Algorithm 4.6.*

INITIALIZATION: *Set $r \leftarrow r_+$, fix a sufficiently large set Δ of numbers, sample from this set the entries of (weakly) random matrices U of size $m \times r$ and V of size $n \times r$, and compute the matrices $U \leftarrow ||C||Q(U)$, $V \leftarrow Q(V)$, and $C \leftarrow C + UV^H$. (If $r_+ = 0$, then U and V are the empty matrices of the sizes $m \times 0$ and $n \times 0$, respectively, and $C = A$.)*

COMPUTATIONS:

1. *Apply the Subroutine ILL·CONDITIONED to the matrix C.*
2. *If $\operatorname{cond} C > \frac{1}{\tau}$, then output FAILURE and stop. If $\operatorname{cond} C \leq \frac{1}{\tau}$, compute the matrices C^+U, $B = Q(C^+U)$, and AB.*
3. *Apply the Subroutine NORM to the matrix AB. If bound (4.2) holds, output the integer r and the matrix B and stop. (B is the $m \times 0$ empty matrix if $r = 0$.)*

4. Otherwise if $r = r_-$, output FAILURE and stop. If $r > r_-$, update the matrices U and V by removing their last columns $\mathbf{u}$ and $\mathbf{v}$, respectively; update the matrix C by setting $C \leftarrow C - \mathbf{u}\mathbf{v}^H$, set $r \leftarrow r - 1$, and go to Stage 1.

Algorithm 4.8. A numerical nmb via aggregation.

INPUT, OUTPUT, INITIALIZATION *and Stages 1, 2 and 3 of* COMPUTATIONS *are as in Algorithm 4.7.*

COMPUTATIONS:

4. Otherwise observe that $r_- \le \operatorname{nul} G < r$ for the computed matrix $G = I_r - V^H C^+ U$ and apply the algorithm to this matrix. Unless the algorithm fails, it approximates an $r \times q$ nmb(G). Denote this nmb by X and compute the matrix BX. Set $r \leftarrow q$ and $B \leftarrow Q(BX)$, output the integer r and the matrix B, and stop.

4.5 Numerical computation of nmbs: the error and tolerance bounds

Bound (4.1) holds if and only if our A-modification does not improve conditioning to the desired level. Bound (4.2) shows us that the matrix B approximates an nmb(A) within a fixed tolerance to the residual norm. The two bounds enable us to extend rules i)–iv) in Section 4.2 to numerical computations.

We can test bound (4.1) by applying the effective condition estimators in [17, Section 3.5.4], [18, Chapter 15], and [43, Section 5.3], but our comments in Remark 4.6 on relaxing the application of the Subroutine FULL·RANK can be extended to the application of the Subroutine ILL·CONDITIONED, which verifies the bound (4.1).

Let us link bounds (4.1) and (4.2) to the respective error estimates based on the results in Section 3.5. Let $\Delta(M) = \mathrm{fl}(M) - M$ denote the error matrix in floating-point computation of a matrix M with rounding to a fixed (e.g., the IEEE standard double) precision. Assume that the matrices A, U, and V have been normalized by scaling so that $||A|| = ||U|| = ||V|| = 1$ and therefore $||C|| \le 2$. Further assume that $\Delta(C) = 0$, thus ignoring the smaller errors in computing the matrix C (cf. Remark 4.8). Write $\kappa_- = ||C^+||$. To simplify the estimates, ignore the terms of higher orders in the unit roundoff u and write $c_{m,n}$ for the bounds that depend on the dimensions m and n, but otherwise are independent of the matrix C.

By combining the estimates in Section 3.5 for the errors in matrix multiplication and in computing the solutions and least squares solutions to linear systems of equations, we obtain that $||\Delta(C^+)|| \le c_{m,n} u \kappa_-$ and $||\Delta(AC^+U|| \le c_{m,n} u ||A|| \kappa_-$. These estimates can guide our choice of tolerances t, τ and $\tilde{\tau}$.

We can decrease the value u and therefore the output residual norm bounds if we increase the precision of computing. We can also do this implicitly, staying with the double precision computations, but applying the fast advanced

algorithms in [12], [18], [21], [27], [36], [41] (which compute sums and products error-free or with the desired high accuracy) and the extended iterative refinement for the solution of linear systems of equations in [31], [32], and [37].

Remark 4.7. *The error estimates are more favorable in the case of square (nonsingular) matrices C, and this can motivate using the respective techniques in Section 3.2.*

Remark 4.8. *By carefully performing A-preprocessing, we can control or even exclude the errors in computing the matrix C. E.g., we can fill the weakly random matrices U or V with shorter numbers. We can make the computation of the A-modification C multiplication-free by filling the matrices U or V with short integers (say just with -2, -1, 0, 1, and 2). The power of A-preconditioning is still preserved for such APPs according to our analysis and extensive experiments. Also see [32, Section 12] on an alternative technique of error-free preprocessing.*

5 From the null spaces to the invariant spaces

Suppose a computational problem is defined by an unknown well conditioned but rank deficient matrix $\tilde{A}$ of the size $m \times n$ where $m \geq n$. Further assume that we are given its close approximation with an ill conditioned matrix $A = \tilde{A} - E$ of full rank. Finally suppose an APP UV^H has the minimum rank r such that the A-modification $C = A + UV^H$ is well conditioned. Then the matrix $\tilde{C}^+U = (C - E)^+U$ is an nmb($\tilde{A}$) for an unknown matrix $\tilde{A}$ of our interest, and the matrix C^+U approximates this nmb. In Section 5.2 we supply some relevant quantitative estimates. Next we reexamine the same argument relating it to the approximation of the (right) r-tails of the SVD of the matrices $\tilde{A}$ and A.

5.1 From the null spaces to the tails of the SVDs

The null space $N(\tilde{A})$ of an $m \times n$ matrix $\tilde{A}$ such that $m \geq n > \operatorname{nul} \tilde{A} = r > 0$ is also the (right) r-tail of this matrix (that is, the singular space associated with its r smallest singular values, equal to zero). According to Theorem 3.1, this space is also given by range($\tilde{C}^+U$) provided the matrix $\tilde{C} = \tilde{A} + UV^H$ has full rank n for an APP UV^H of rank r. For a matrix E of a smaller norm, the perturbation $A = \tilde{A} - E$ has exactly r vanishing or small positive singular values. If the matrix $\tilde{C}$ is well conditioned, then according to Theorem 3.15, the matrix $\tilde{C}^+$ and consequently the space range($\tilde{C}^+U$) changes little in the transition $\tilde{A} \to A$ to a nearby matrix A. We recall the following formal estimates supporting this claim.

Lemma 5.1. *Let C and $C - E$ be two matrices of full rank. Then*

$$||(C-E)^+ - C^+|| \leq ||(C-E)^+ - C^+||_F \leq 2||E||_F \max\{||C^+||^2,\ ||(C-E)^+||^2\}.$$

Proof. See [17, Section 5.5.5] for $\delta A = -E$. □

Lemma 5.2. *For two square matrices C and E of the same size such that the matrix C is nonsingular and $||C^{-1}E|| = \theta < 1$, we have $||I - (C-E)^{-1}C|| \le \frac{\theta}{1-\theta}$.*

Proof. See [43, Theorem 1.4.18] for $P = C^{-1}E$. □

In the transition $\tilde{A} \to A = \tilde{A} - E$ the singular values satisfy the bounds $|\sigma_j(\tilde{A}) - \sigma_j(A)| \le ||E||$ for all i [44, Theorem 3.3.3]. Therefore, the space range(C^+U) closely approximates the (right) r-tail of the matrix A if the perturbation norm $|E||$ is small. In this case we call the computation of the matrix C^+U the *Tail Approximation* for the matrix $\tilde{A}$ (cf. some estimates for the residual norms $||AC^+U||$ in the next subsection).

Let us define the relevant concepts of δ-nullity and *numerical nullity* nnul A. For a fixed small positive tolerance δ we say that r is the δ-nullity of an $m \times n$ matrix A if there are r singular values (counting multiplicity) that are less than $\delta\sigma_1(A)$, whereas the ratio $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$ is not very small, that is if $\sigma_{n-r}(A) \gg \delta\sigma_1(A) \ge \sigma_{n-r+1}(A)$. In this definition we assume a gap in the spectrum of the singular values of the input matrix A. We define the *numerical nullity* nnul A as the δ-nullity for a small positive value δ. One can employ the parameter δ instead of the nullity in the extension of our null basis algorithms to the Tail Approximation. We refer the reader to [39], [40] on the incorporation of the Tail Approximation into the inverse power iteration for eigen-solving.

5.2 Residual norm estimates for the Tail Approximation

Theorem 5.1. *Assume an $m \times n$ matrix A for $m \ge n$ and an APP UV^H such that the A-modification $C = A + UV^H$ has full rank n. Then the vector $\mathbf{y} - C^+A\mathbf{y}$ lies in the space* range(C^+U).

Proof. Postmultiply the matrix equation $C = A+UV^H$ by $\mathbf{y}$, pre-multiply it by C^+, substitute $C^+C = I_n$, and obtain that $\mathbf{y} = C^+A\mathbf{y}+C^+U\mathbf{z}$ for $\mathbf{z} = V\mathbf{y}$. □

The theorem implies that for a matrix A and a well conditioned matrix $C = A + UV^H$ of full rank, a vector $\mathbf{y}$ lies near the space range(C^+U) provided the vector $A\mathbf{y}$ has a small norm (and the norm $||C^+||$ is not very large). Conversely, our next theorem bounds the norm $||A\mathbf{y}||$ for the vectors $\mathbf{y}$ from the space range(C^+U).

Theorem 5.2. *For positive integers m, n, and r where $m \ge n$, a pair of $m \times n$ matrices A and E, and a pair of unitary matrices U of size $m \times r$ and V of size $n \times r$, write $C = A + UV^H$ and assume that $r =$ nul$(A - E)$, the matrix C has full rank,*

$$||A|| = 1,\ \delta = ||E||_F < \sigma_- = \sigma_n(C) = \frac{1}{||C^+||} \le ||C|| \le 2,$$

and $\mathbf{y} = C^+U\mathbf{x}$ *for a normalized vector* $\mathbf{x}$. *Then* $||A\mathbf{y}|| \le \tau||A||||\mathbf{y}||$ *where* $\tau \le \delta + (4+4\delta)\frac{\delta}{(\sigma_- - \delta)^2}$ *(for* $m \ge n$*), and if* $m = n$, *then* $\tau \le \delta + (1+\delta)\frac{\delta}{\sigma_- - \delta}$.

Proof. We have $(A-E)(C-E)^+U\mathbf{x} = \mathbf{0}$ in virtue of Theorem 3.1 (cf. (3.4)). Therefore, $A\mathbf{y} = E\mathbf{y} + \mathbf{z}$ where

$$\mathbf{z} = (A-E)\mathbf{y} = (A-E)C^+U\mathbf{x} = (A-E)(C^+ - (C-E)^+)U\mathbf{x}.$$

It follows that $||\mathbf{z}|| \le ||A-E||\ ||C^+ - (C-E)^+|| \le (1+\delta)||C^+ - (C-E)^+||$ because $||E|| \le ||E||_F = \delta$, $||\tilde{A}|| = 1$, and consequently $||A-E|| \le ||A|| + ||E|| \le 1+\delta$.

Moreover, $2||\mathbf{y}|| \ge ||\mathbf{y}||||C|| \ge ||C\mathbf{y}||$, and since $C\mathbf{y} = U\mathbf{x}$ for $m \ge n$, we obtain that $2||\mathbf{y}|| \ge ||U\mathbf{x}|| = 1$. Furthermore, $||(C-E)^+|| \le \frac{1}{\sigma_- - \delta}$, $||C^+|| = \frac{1}{\sigma_-}$. Combine all these estimates with Lemma 5.1 and obtain the claimed bound on τ for $m \ge n$.

For $m = n$ we have $C^+ - (C-E)^+ = (I - (C-E)^{-1}C)C^{-1}$, and therefore $\mathbf{z} = (A-E)(I-(C-E)^{-1}C)C^{-1}U\mathbf{x}$. Substitute $\mathbf{y} = C^+U\mathbf{x}$ and obtain $\mathbf{z} = (A-E)(I-(C-E)^{-1}C)\mathbf{y}$. Consequently $||\mathbf{z}|| \le ||A-E||\ ||I-(C-E)^{-1}C||\ ||\mathbf{y}||$. To estimate the norm $||I-(C-E)^{-1}C||$, apply Lemma 5.2 and substitute the bound $||C^{-1}E|| \le ||C^{-1}||||E|| \le \frac{\delta}{\sigma_-}$. Combine the resulting estimate for the norm $||\mathbf{z}||$ with the bound $||A|| \le 1+\delta$ and the equation $A\mathbf{y} = E\mathbf{y} + \mathbf{z}$ and obtain the theorem for $m = n$. □

5.3 Numerical nmbs and A-preconditioning

Now suppose an input matrix A is both rank deficient and ill conditioned. Let exactly $r = \operatorname{nnul} A$ singular values of the matrix A (counting them with their multiplicities and including the zero singular values) be small relatively to the norm $||A||$ and let two matrices U and V be properly scaled, weakly random, and have rank r. Then we can expect that the matrix $C = A + UV^H$ is well conditioned and has full rank. In this case property (3.4) holds and we can recall Theorem 3.2 and apply Algorithm 4.8 to compute an nmb X for the matrix $G = I_r - V^HC^+U$ and then the nmb C^+UX for the matrix A. Then our numerical problems would be confined to the computation of and with the Schur aggregate G of a smaller size. These stages require higher accuracy, so that we need either high precision computations or their emulation with the cited double precision algorithms for sums and products in [12], [18], [21], [27], [36], [41] and for iterative refinement in [31], [32], and [37].

5.4 The Head Approximation

By applying the Tail Approximation to the $n \times m$ matrix A^+ with q small positive singular values and to a dual APP VU^H, we define the *Head Approximation*. In this case the matrix $C_- = A^+ + VU^H$ plays the role of the A-modification C used in the Tail Approximation, but it is more efficient to operate with the

matrix $(C_-)^+$ and to adjust the algorithms respectively. For full rank matrices A and C_- we have the expression

$$(C_-)^+ = (A^+ + VU^H)^+ = A - AVH^{-1}U^HA, \quad H = I_q + U^HAV,$$

which we call the Sherman–Morrison–Woodbury dual formula (cf. [31], [37]). In this case we invert only the $q \times q$ *dual aggregate* H.

5.5 Improving A-preconditioners via orthogonalization

Suppose for an ill conditioned matrix A of full rank we obtain a crude A-preconditioner and the integer $\operatorname{nnul} A$, e.g., by extending the algorithms in Section 4 to the Tail Approximation. In this section we refine such an A-preconditioner. Due to the results in [31], [37], the refined preconditioners can serve, e.g., as pointers to the gaps in the spectrum of the singular values.

At first let $\tilde{A}$ denote a rank deficient matrix with nullity r and let UV^H denote an APP of the rank r such that the A-modification $\tilde{C} = \tilde{A} + UV^H$ has full rank. We may have $\operatorname{cond}\tilde{C} > \operatorname{cond}\tilde{A}$ and even $\operatorname{cond}\tilde{C} \gg \operatorname{cond}\tilde{A}$, but the following transform serves as a remedy,

$$(U \leftarrow Q(\tilde{C}^+U), \quad V \leftarrow Q(\tilde{C}^{+H}V)). \tag{5.1}$$

Clearly, with the new APP UV^H the A-modification $C = A + UV^H$ still has full rank, and our next theorem shows that A-preprocessing with this APP preserves the condition number of the matrix $\tilde{A}$.

Theorem 5.3. *Let $\tilde{A}$ be a normalized $n \times n$ matrix of a rank $\rho < n$ and let U and V be a pair of $n \times r$ unitary matrices such that $r = n - \rho = \operatorname{nul}\tilde{A}$ and the matrix $\tilde{C} = \tilde{A} + UV^H$ is nonsingular. Let $U_1 = Q(\tilde{C}^+U)$ and $V_1 = Q(\tilde{C}^{+H}V)$ denote the respective updates of the matrices U and V according to policy (5.1). Then the matrix $\tilde{A} + U_1V_1^H$ is nonsingular and* $\operatorname{cond}(\tilde{A} + U_1V_1^H) = \operatorname{cond}\tilde{A}$.

Proof. Due to Theorem 3.1, the updated matrices U_1 and V_1 are the right and left nmbs for the matrix $\tilde{A}$, respectively. Let $\tilde{A} = \sum_{j=1}^{\rho}\sigma_j\mathbf{s}_j\mathbf{t}_j^H$ be the SVD of the matrix $\tilde{A}$. Write $U_1 = (\mathbf{u}_j)_{j=1}^r$ and $V_1 = (\mathbf{v}_j)_{j=1}^r$ and obtain the SVD of the matrix $\tilde{A} + U_1V_1^H = \sum_{j=1}^{r}\mathbf{u}_j\mathbf{v}_j^H + \sum_{j=1}^{\rho}\sigma_j\mathbf{s}_j\mathbf{t}_j^H$. Theorem 5.3 follows because $r = n - \rho$ and $\sigma_1 = 1$. □

Suppose the matrix $\tilde{A}$ in this theorem is well conditioned. Then so is the matrix $\tilde{A}+U_1V_1^H$ as well as all nearby matrices. Therefore, the APP $U_1V_1^H$ preconditions any (ill conditioned) matrix $A = \tilde{A} - E$ with $\operatorname{nnul} A = \operatorname{rank}(U_1V_1^H)$ lying near the matrix $\tilde{A}$. According to our extensive tests (cf. [34, Table 7.2]) the transformation (5.1) substantially increases the preconditioning power of a crude preconditioner UV^H for such a matrix A.

5.6 Generating and improving A-preconditioners via inflation and compression

Suppose we have an upper bound h on the unknown number r of small (positive and zero) singular values of an $m \times n$ input matrix A for $m \geq n$. To approximate an nmb(A), we can generate a scaled random APP UV^H of rank h, compute the A-modification $C = A + UV^H$, approximate the matrix C^+U, and test whether $AC^+U = 0$. If not so, we can choose a candidate integer $r < h$ and approximate the singular space associated with the r smallest singular values of the matrix A by extending transform (5.1) to the compression of the APP as follows.

Flowchart 5.1. Inflation/Compression of an APP *(cf. [50]).*

1. (Generation of an inflated APP.) *Generate an APP UV^H of rank h.*
2. (The Tail Approximation.) *Compute two unitary or well conditioned matrix bases $T(U)$ and $T(V)$ for the r-trailing right singular spaces of the matrices AC^+U and $A^HC^{+H}V$, respectively. (If $m = n$ and the matrices U and V are unitary, we can apply Theorem 3.2 and compute just the r-tail of the matrix $G = I_h - V^HC^+U$.)*
3. (Compression.) *Compute and output the new generators $U \leftarrow Q(C^+UT(U))$ and $V \leftarrow Q(C^{+H}VT(V))$ and the new APP UV^H.*

If we have no targit integer r, we can apply the flowchart recursively, say for $r = 1, 2, \ldots$, until the matrix AC^+U vanishes or nearly vanishes.

X. Wang in [50] has applied an algorithm similar to Flowchart 5.1 to 10×10 Hilbert input matrices $A = (\frac{1}{i+j-1})_{i,j=1}^{10}$ and has consistently arrived at $\operatorname{cond} C \approx \frac{\sigma_1(A)}{\sigma_{10-h}(A)}$ in his extensive tests for various choices of positive $h \leq 10$ and $r < h$.

(Weakly) random APPs UV^H whose rank exceeds $\operatorname{nnul} A$ is a safe initial choice for obtaining a well conditioned matrix $C = A + UV^H$ according to our tests. Flowchart 5.1 complements this choice to yield effective A-preconditioners of the rank $\operatorname{nnul} A$.

Here is a natural extension of our policy (5.1) to dual APPs VU^H,

$$V \leftarrow Q((\tilde{C}_-)^+V), \quad U \leftarrow Q((\tilde{C}_-)^{+H}U)). \tag{5.2}$$

6 A-preprocessing and matrix structure

If an input matrix A can be multiplied by a vector fast, then we can choose APPs with the same property, to have it also for the A-modifications. For such scaled weakly random APPs of sufficiently large ranks, we are likely to arrive at well conditioned matrices C, and if so, it can be effective to compute the matrices C^+U or V^HC^+ by applying the Conjugate Gradient algorithms.

Furthermore if an input matrix A has the displacement structure, then we can choose a pair of generators U and V with consistent structure, and as we

already recalled from [34] and [35], this turns out to be compatible with the power of A-preprocessing (see Example 3.1 in Section 3.4, [34, Examples 4.1–4.5], [35, Examples 1–5]). Only a few matrix additions, multiplications, and inversions are required for the computation of the APP UV^H, the A-modification $C = A + UV^H$, and the matrices $V^H C^+$, C^+U, G and G^{-1}, and so we can preserve and employ the structure to perform these operations fast [30].

Dealing with structured matrices A and C, we should avoid involving QR factorization in our computations, not to destroy the structure. In particular in Algorithms 4.5–4.8 we should set $B \leftarrow \frac{C^+U}{||C^+U||}$ rather than $B \leftarrow Q(C^+U)$, $U \leftarrow \frac{||C||U}{||U||}$ rather than $U \leftarrow ||C||Q(U)$, and $V \leftarrow \frac{V}{||V||}$ rather than $V \leftarrow Q(V)$, whereas in Algorithms 4.6 and 4.7 we should apply the Sherman–Morrison–Woodbury formula for updating the inverse C^{-1} (cf. Remark 4.2).

[34, Examples 4.1–4.6], [35, Examples 1–6] show APPs having various most popular matrix structures. Furthermore, the *method of displacement transformation* in [29] and [30] enables us to extend the power of these APPs to other classes of structured matrices, even to the classes that contain no well conditioned matrices and thus contain no well conditioned APPs [16], [47].

Assume an $n \times n$ structured ill conditioned matrix A with exactly r singular values that are small relatively to the norm $||A||$ (we count every singular value with its multiplicity). Then the structured matrices $V^H C^+$ of size $r \times n$ and C^+U of size $n \times r$ approximate some matrix bases for the left and right singular spaces associated with the r smallest singular values of the matrix A (cf. Section 5.1). This holds even where the singular spaces have no structured matrix bases, that is no structured matrices of full rank whose rows (resp. columns) span these spaces.

7 Numerical experiments

7.1 Generation of input matrices and APPs

In a series of numerical experiments in [34], scaled weakly random APPs UV^T (including the APPs with various patterns of sparseness and structure of generators U and V) were generated for various matrix classes. In good accordance with the theoretical estimates in Section 3.4, the tests in [34] (and similarly our present tests) showed that adding scaled weakly random APP of rank r to an $n \times n$ ill conditioned matrix A has regularly decreased its condition number roughly to the level $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$.

For $n = 64$ as well as for $n = 128$, we generated 1000 instances of the $n \times n$ input matrices A computed numerically, with double precision, as the products $S\Sigma T^T$ (cf. [18, Section 28.3]). Here S and T were generated as random real orthonormal matrices, that is, the Q-factors in the thin QR factorization of matrices with random integer entries from the range $(-10^4, 10^4)$ and with positive diagonal entries of the R-factors, whereas $\Sigma = \mathrm{diag}(\sigma_i)_{i=1}^n$ were diagonal matrices with the diagonal entries $\sigma_1, \dots, \sigma_1$ from one of the four following classes.

Class 1. $\sigma_i = \frac{1}{i}$ for $i = 1, \ldots, n-k$, $\sigma_i = 0$ for $i > n-k$,

Class 2. $\sigma_i = \frac{1}{i}$ for $i = 1, \ldots, n-k$, $\sigma_i = \frac{10^{-14}}{i-n+k}$ for $i > n-k$,

Class 3. $\sigma_i = \frac{1}{i}$ for $i = 1, \ldots, n-k-l$, $\sigma_i = \frac{10^{-9}}{i-n+k+l}$ for $i = n-k-l+1, \ldots, n-k$, $\sigma_i = 0$ for $i > n-k$,

Class 4. $\sigma_i = \frac{1}{i}$ for $i = 1, \ldots, n-k-l$, $\sigma_i = \frac{10^{-9}}{i-n+k+l}$ for $i = n-k-l+1, \ldots, n-k$, $\sigma_i = \frac{10^{-14}}{i-n+k}$ for $i > n-k$.

For each of these classes, besides generating random orthonormal matrices T independently of the matrices S, we defined T by setting $T = S$. Respectively we defined Classes 1n, 1s, 2n, 2s, 3n, 3s, 4n, and 4s where "n" stood for "nonsymmetric" and "s" for "symmetric".

In our tests we selected $k = 24$ and $l = 20$ for $n = 64$ and selected $k = 48$ and $l = 40$ for $n = 128$.

7.2 Computation of nmbs

For every instance of the input matrix A we generated the A-modification matrix $C = A + UV^T$ for random $n \times r$ generators U and $V = U$ scaled so that $||UV^T|| = 1$ and $r = k$ for Classes 1 and 2 or $r = k + l$ for Classes 3 and 4.

Then we computed approximate nmbs by applying Algorithm 4.5 for Classes 1 and 2 and Algorithm 4.8 for Classes 3 and 4. In the latter case we successively computed the matrices $C^{-1}U$, $G = I_r - V^T C^{-1}U$ for $r = k+l$, an approximate nmb X for the matrix G, and finally the approximate nmb $C^{-1}UX$ for the input matrix A.

In all cases we estimated the ratios $\frac{||AC^{-1}U||}{||A||\ ||C^{-1}U||}$ and $\frac{||AC^{-1}UX||}{||A||\ ||C^{-1}UX||}$, which are the relative residual norms for the matrices A in Classes 1 and 2 and in Classes 3 and 4, respectively. We output their maximum, minimum, and average values as well as the standard deviations for each algorithm and each case. Tables 7.1 and 7.2 show the results of our tests performed with double precision and without using the extended iterative refinement.

We have also run 100 tests for each of $n = 64$ and $n = 128$ and for the input matrices A where we computed these matrices as the error-free products $A = S\Sigma T^T$ and applied the extended iterative refinement at the stage of computing the matrices $C^{-1}U$ and G^{-1}. Tables 7.3 and 7.4 display the results of these tests. As we expected, in the case of matrices A of Classes 2 and 4, the residual norms decreases only to the level about the smallest positive singular value σ_n, whereas in the case of matrices A of Classes 1 and 3 these norms immediately went below the level achieved with the costly SVD-based algorithms and then kept decreasing rapidly towards zero until we stopped the iterative refinement process with the ratios at the level below 10^{-40}.

Table 7.1: residual norms for 64×64 matrices

Class	**Type**	**min**	**max**	**mean**	**std**
1	n	9.6×10^{-16}	3.0×10^{-11}	6.6×10^{-14}	9.8×10^{-13}
1	s	8.7×10^{-16}	2.8×10^{-12}	2.1×10^{-14}	1.1×10^{-13}
2	n	3.8×10^{-15}	7.8×10^{-12}	1.0×10^{-13}	4.1×10^{-13}
2	s	3.8×10^{-15}	5.7×10^{-12}	9.7×10^{-14}	3.9×10^{-13}
3	n	1.1×10^{-13}	1.6×10^{-10}	8.5×10^{-12}	1.4×10^{-11}
3	s	1.2×10^{-14}	2.9×10^{-10}	1.6×10^{-12}	1.3×10^{-11}
4	n	9.7×10^{-14}	1.8×10^{-10}	8.9×10^{-12}	1.5×10^{-11}
4	s	1.4×10^{-14}	3.8×10^{-10}	2.0×10^{-12}	1.5×10^{-11}

Table 7.2: residual norms for 128×128 matrices

Class	**Type**	**min**	**max**	**mean**	**std**
1	n	5.9×10^{-15}	1.2×10^{-11}	1.1×10^{-13}	5.7×10^{-13}
1	s	1.9×10^{-15}	8.1×10^{-12}	5.6×10^{-14}	3.6×10^{-13}
2	n	5.9×10^{-15}	7.5×10^{-11}	2.1×10^{-13}	2.4×10^{-12}
2	s	4.6×10^{-15}	8.0×10^{-12}	1.1×10^{-13}	4.5×10^{-13}
3	n	1.0×10^{-12}	2.4×10^{-10}	1.6×10^{-11}	1.7×10^{-11}
3	s	6.1×10^{-14}	3.0×10^{-10}	2.9×10^{-12}	1.3×10^{-11}
4	n	1.2×10^{-12}	2.4×10^{-10}	1.7×10^{-11}	1.8×10^{-11}
4	s	8.1×10^{-14}	2.9×10^{-10}	4.2×10^{-12}	1.5×10^{-11}

References

[1] O. Axelsson, *Iterative Solution Methiods*, Cambridge University Press, Cambridge, England, 1994.

[2] Å. Björck, Stability Analysis of the Method of Seminormal Equations, *Linear Algebra and Its Applications*, **88/89**, 31–48, 1987.

[3] Å. Björck, Numerics of Gram-Schmidt Orthogonalization, *Linear Algebra and Its Applications*, **197/198**, 297–316, 1994.

[4] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[5] M. Benzi, Preconditioning Techniques for Large Linear Systems: a Survey, *J. of Computational Physics*, **182**, 418–477, 2002.

[6] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, *Templates for the*

Table 7.3: residual norms for 64×64 matrices (in computations with iterative refinement and extended precision)

Class	**Type**	**min**	**max**	**mean**	**std**
1	n	4.0×10^{-53}	5.2×10^{-49}	6.0×10^{-50}	1.6×10^{-49}
1	s	1.9×10^{-59}	6.3×10^{-47}	6.3×10^{-48}	2.0×10^{-47}
2	n	1.0×10^{-14}	1.5×10^{-13}	5.2×10^{-14}	4.6×10^{-14}
2	s	4.1×10^{-14}	3.5×10^{-12}	4.9×10^{-13}	1.0×10^{-12}
3	n	2.4×10^{-50}	8.9×10^{-43}	9.9×10^{-44}	3.0×10^{-43}
3	s	2.8×10^{-55}	3.0×10^{-43}	3.0×10^{-44}	9.4×10^{-44}
4	n	2.9×10^{-13}	1.6×10^{-12}	6.4×10^{-13}	4.0×10^{-13}
4	s	9.7×10^{-13}	9.4×10^{-11}	1.7×10^{-11}	2.9×10^{-11}

Table 7.4: residual norms for 128×128 matrices (in computations with iterative refinement and extended precision

Class	**Type**	**min**	**max**	**mean**	**std**
1	n	1.8×10^{-56}	2.3×10^{-45}	2.3×10^{-46}	7.3×10^{-46}
1	s	6.9×10^{-57}	3.9×10^{-44}	4.9×10^{-45}	1.4×10^{-44}
2	n	2.0×10^{-14}	4.2×10^{-12}	5.9×10^{-13}	1.3×10^{-12}
2	s	4.9×10^{-14}	1.8×10^{-11}	3.3×10^{-12}	6.4×10^{-12}
3	n	2.4×10^{-55}	7.9×10^{-49}	1.1×10^{-49}	2.5×10^{-49}
3	s	1.6×10^{-52}	3.9×10^{-47}	5.7×10^{-48}	1.4×10^{-47}
4	n	1.7×10^{-13}	2.0×10^{-11}	4.0×10^{-12}	6.3×10^{-12}
4	s	3.2×10^{-13}	1.3×10^{-11}	3.3×10^{-12}	4.6×10^{-12}

Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, Philadelphia, 1993.

[7] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[8] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, Cambridge, England, 2005.

[9] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[10] J. J. Dongarra, I. S. Duff, D. C. Sorensen, H. A. van Der Vorst, *Numerical Linear Algebra for High-Performance Computers*, SIAM, Philadelphia, 1998.

[11] I. S. Duff, A. M. Erisman, J. K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, England, 1986.

[12] J. Demmel, Y. Hida, Accurate and Efficient Floating Point Summation, *SIAM J. on Scientific Computing*, **25**, 1214–1248, 2003.

[13] R. A. Demillo, R. J. Lipton, A Probabilistic Remark on Algebraic Program Testing, *Information Processing Letters*, **7**, **4**, 193–195, 1978.

[14] A. Edelman, Eigenvalues and Condition Numbers of Random Matrices, *SIAM J. on Matrix Analysis and Applications*, **9**, **4**, 543–560, 1988.

[15] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[16] W. Gautschi, G. Inglese, Lower Bounds for the Condition Number of Vandermonde Matrices, *Numerische Math.*, **52**, 241–250, 1988.

[17] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 3rd edition, The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[18] N. J. Higham, *Accuracy and Stability in Numerical Analysis*, SIAM, Philadelphia, 2002 (second edition).

[19] I. Kaporin, A Practical Algorithm for Faster Matrix Multiplication, *Numerical Linear Algebra with Applications*, **6, 8**, 687-700, 1999.

[20] I. Kaporin, The Aggregation and Cancellation Techniques As a Practical Tool for Faster Matrix Multiplication, *Theoretical Computer Science*, **315**, **2–3**, 469–510, 2004.

[21] X. Li, J. Demmel, D. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Kang, A. Kapur, M. Martin, B. Thompson, T. Tung, D. Yoo, Design, Implementation and Testing of Extended and Mixed Precision BLAS, *ACM Transactions on Mathematical Software*, **28**, 152–205, 2002. http //crd.lbl.gov/x̃iaoye/XBLAS/.

[22] C. L. Lawson, R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1974. Reissued with a survey of recent debvelopments by SIAM, Philadelphia, 1995.

[23] J. Laderman, V. Y. Pan, H. X. Sha, On Practical Algorithms for Accelerated Matrix Multiplication, *Linear Algebra and Its Applications*, **162–164**, 557–588, 1992.

[24] R. J. Lipton, D. Rose, R. E. Tarjan, Generalized Nested Dissection, *SIAM J. on Numerical Analysis*, **16, 2**, 346–358, 1979.

[25] W. L. Miranker, V. Y. Pan, Methods of Aggregations, *Linear Algebra and Its Applications*, **29**, 231–257, 1980.

[26] F. W. J. Olver, Error Analysis of Complex Arithmetic, in *Computational Aspect of Complex Analysis*, Volume 102 of *NATO Advanced Study Institute Series C* (edited by D. Reidel), 279–292. Dordrecht, Holland, 1983.

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

[28] V. Y. Pan, How Can We Speed up Matrix Multiplication? *SIAM Review,* **26**, **3**, 393–415, 1984.

[29] V. Y. Pan, On Computations with Dense Structured Matrices, *Math. of Computation*, **55, 191**, 179–190, 1990.

[30] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

[31] V. Y. Pan, The Schur Aggregation and Extended Iterative Refinement, Technical Report TR 2007, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2007. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[32] V. Y. Pan, D. Grady, B. Murphy, G. Qian, R. E. Rosholt, A. Ruslanov, Schur Aggregation for Linear Systems and Determinants, *Theoretical Computer Science*, *Special Issue on Symbolic–Numerical Algorithms* (D. A. Bini, V. Y. Pan, and J. Verschelde editors), accepted. Also Technical Report TR 2008008, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2008. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[33] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, I. Taj-Eddin, Y. Tang, X. Yan, Additive Preconditioning and Aggregation in Matrix Computations, *Computers and Mathematics (with Applications)*, **55**, **8**, 1870–1886, 2008.

[34] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, Technical Report TR 2008004, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2008. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[35] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, *Proc. of the Third International Computer Science Symposium in Russia (CSR 2008)*, *Lecture Notes in Computer Science (LNCS)*, **5010**, 372–383, 2008.

[36] V. Y. Pan, B. Murphy, G. Qian, R. E. Rosholt, Error-free Computations via Floating-Point Operations, *Computers and Mathematics (with Applications)*, in press, and Technical Report TR 2007010, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2007. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[37] V. Y. Pan, B. Murphy, R. E. Rosholt, M. Tabanjeh, The Schur Aggregation for Solving Linear Systems of Equations, *Proceedings of the Third International Workshop on Symbolic–Numeric Computation (SNC 2007)*, 142–151, July 2007, London, Ontario, Canada, (Jan Verschelde and Stephen Watt editors), ACM Press, New York, 2007.

[38] V. Y. Pan, J. Reif, Fast and Efficient Parallel Solution of Sparse Linear Systems, *SIAM J. on Computing*, **22**, **6**, 1227–1250, 1993.

[39] V. Y. Pan, X. Yan, Null Space and Eigenspace Computations with Additive Preprocessing, *Proceedings of the Third International Workshop on Symbolic–Numeric Computation (SNC 2007)*, 152–160, July 2007, London, Ontario, Canada (Jan Verschelde and Stephen Watt editors), ACM Press, New York, 2007.

[40] V. Y. Pan, X. Yan, Additive Preconditioning, Eigenspaces, and the Inverse Iteration, Technical Report TR 2008006, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2008. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[41] S. M. Rump, T. Ogita, S. Oishi, Accurate Floating-Point Summation, Tech. Report 05.12, *Faculty for Information and Communication Sciences*, Hamburg University of Technology, November, 2005.

[42] J. T. Schwartz, Fast Probabilistic Algorithms for Verification of Polynomial Identities, *Journal of ACM*, **27**, **4**, 701–717, 1980.

[43] G. W. Stewart, *Matrix Algorithms, Vol I: Basic Decompositions*, SIAM, Philadelphia, 1998.

[44] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, 1998.

[45] D. Spielman, S.-H. Teng, Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time, *Proc. 33rd Annual ACM Symposium on the Theory of Computing (STOC 2001)*, 296–305, ACM Press, New York, 2001. Full version available at http://math.mit.edu/ spielman/SmoothedAnalysis.

[46] D. Spielman, S.-H. Teng, Smoothed Analysis of Algorithms, *Proc. of the International Congress of Mathematicians* (Beijing 2002), Vol. I, 597–606, Higher ED. Press, Beijing, 2002.

[47] E. E. Tyrtyshnikov, How Bad Are Hankel Matrices? *Numerische Math.*, **67, 2**, 261–269, 1994.

[48] Van H. Vu, Terence Tao, The Condition Number of a Randomly Perturbed Matrix, *Proc. Annual Symposium on Theory of Computing (STOC 2007)*, 248–255, ACM Press, New York, 2007.

[49] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.

[50] X. Wang, Affect of Small Rank Modification on the Condition Number of a Matrix, *Computer and Math. (with Applications)*, **54**, 819–825, 2007.

[51] R. E. Zippel, Probabilistic Algorithms for Sparse Polynomials, *Proceedings of EUROSAM'79, Lecture Notes in Computer Science*, **72**, 216–226, Springer, Berlin, 1979.