2009

TR-2009014: Randomized Preprocessing of Homogeneous Linear Systems

Victor Y. Pan

Guoliang Qian

Randomized Preprocessing of Homogeneous Linear Systems *

Victor Y. Pan[1,2],[a] and Guoliang Qian[2],[b]

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

Abstract

Our randomized preprocessing enables pivoting-free and orthogonalization-free solution of homogeneous linear systems of equations. In the case of Toeplitz inputs, we decrease the solution time from quadratic to nearly linear, and our tests show dramatic decrease of the CPU time as well. We prove numerical stability of our randomized algorithms and extend our approach to solving nonsingular linear systems, inversion and generalized (Moore–Penrose) inversion of general and structured matrices by means of Newton's iteration, approximation of a matrix by a nearby matrix that has a smaller rank or a smaller displacement rank, matrix eigen-solving, and root-finding for polynomial and secular equations. Some by-products and extensions of our study can be of independent technical intersest, e.g., our extensions of the Sherman–Morrison–Woodbury formula for matrix inversion, our estimates for the condition number of randomized matrix products, preprocessing via augmentation, and the link of preprocessing to aggregation.

Key words: Linear systems of equations, Randomized preprocessing, Conditioning

1 Introduction

1.1 Null vectors, nmbs, and linear systems of equations

Given an $n \times n$ matrix A, we seek its *null vector* $\mathbf{y}$ (that is a solution of the homogeneous linear system of equations $A\mathbf{y} = \mathbf{0}$) and its *null matrix basis* (hereafter we write *nmb* or nmb(A) for short), that is a matrix whose columns form a basis for the null space $N(A) = \{\mathbf{y} : A\mathbf{y} = \mathbf{0}\}$. These tasks are linked to other fundamental matrix computations in Sections 1.4, 7.2, 8, and 11.

*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, at the International Workshop on Symbolic-Numeric Computation (SNC'07) in London, Ontario, Canada, in July 2007, at the XIXth International Workshop on Operator Theory and its Applications (IWOTA'08), Williamsburg, Virginia, in July 2008, and at the Second International Conference on Structured Linear Algebra Problems: Analysis, Algorithms, and Applications, Cortona, Italy, in September, 2008.

1.2 The customary solution algorithms

Practically, to compute null vectors and nmbs, one employs pivoting (that is row or column interchange), orthogonalization, or the Singular Value Decomposition (SVD). Orthogonalization and particularly SVD are more costly (and more reliable), but even pivoting "usually degrades the performance" [25, page 119], readily destroys matrix structure and sparseness, and threatens or undermines application of block matrix algorithms.

The resulting slowdown of the computations can be significant or even dramatic. E.g., pivoting-free superfast algorithms solve nonsingular Toeplitz or Hankel linear systems of n equations in nearly linear arithmetic time in $O(n \log^2 n)$ [4], [29], [34], [79], [80], whereas the known solution algorithms with pivoting run either in cubic time based on Gaussian elimination or SVD) or in quadratic time in [24], based on the displacement transformation method, proposed in [44] (cf. our Remark 2.1 and [48, Sections 4.8, 4.9, and 5.6]).

1.3 Our acceleration and its technical support

Our alternative is the structure preserving, pivoting-free and orthogonalization-free randomization. Hereafter A^T and A^H denote the transpose and Hermitian (that is complex conjugate) transpose of a matrix A, respectively, and we write "A-" for "additive". Given an $n \times n$ matrix A of a rank $\rho < n$, we generate a pair of $n \times r$ matrices U and V for $r = n - \rho$ and compute the A-preprocessor UV^T and A-modification $C = A + UV^T$. If $\operatorname{rank} UV^T = r$ and the matrix C is nonsingular, then $C^{-1}U = \operatorname{nmb}(A)$, and in the case of Toeplitz or Hankel input the known superfast algorithms yield this nmb in nearly linear arithmetic time versus cubic time based on Gaussian elimination with pivoting, orthogonalization, or SVD.

The acceleration of the known algorithms is dramatic also in terms of the Boolean cost (that is the number of bit-operations involved) [59, Section 9] and, according to our extensive tests, in terms of the CPU time as well. For $n \times n$ Toeplitz inputs our tests showed the decrease of the average CPU time by the factor $a(n)$ where $a(512) > 20$, $a(1024) > 90$, and $a(2048) > 300$ (see Table 10.1).

If the two matrices U and V are random, then $\operatorname{rank} U = \operatorname{rank} V = r$ and $\operatorname{rank} C = n$ with a probability close to one (see Lemma 3.2). This motivates symbolic implementation of our algorithms. To motivate their numerical implementation, one should also prove that with a probability close to one our A-preprocessing $A \Longrightarrow C = A + UV^T$ keeps the matrix well conditioned. Generally, however, the ratio $\frac{\operatorname{cond} C}{\operatorname{cond} A}$ is large if the matrices A and UV^T are singular, the matrix C is not, and the ratio $\frac{||A||}{||UV^T||}$ is large or small. So, having generated random general, Toeplitz or Hankel matrices U and V, we scale them or the matrix A to make the latter ratio neither large nor small. We prove in Section 3.6 and confirm experimentally in [54] that in this case the ratio $\frac{\operatorname{cond} C}{\operatorname{cond} A}$ tends to be neither very large nor very small as well. Our auxiliary estimates for the condition numbers of circulant matrices (cf. Theorem 3.9) and for the product of a fixed well conditioned matrix and a random matrix may be of independent interest (cf. Theorems 3.7 and 3.8).

Empirically, very *weak randomization* is sufficient to support our estimates, e.g., we can set $U = V$ and represent the matrix U as a block vector with the scaled identity blocks $\pm aI$ for random choice of $+$ and $-$ and for a constant $a \approx ||A||^{1/2}$. Our analysis explains this phenomenon as well (see Remark 3.6 in Section 3.6).

Our study of A-preprocessing naturally involves the Sherman–Morrison–Woodbury formula for matrix inversion [25, page 50]. Our modifications and new applications of this formula can be of independent interest (see Sections 2.5, 11.2, and 11.4).

Preprocessing by augmentation $A \Longrightarrow K = \begin{pmatrix} A & U \\ S & W \end{pmatrix}$ for an $r \times r$ block W and $r = n - \rho$ (also studied in [52, Section 12] and [59]) is closely linked to A-preprocessing (cf. Theorem 4.3), but is superior in preserving the structure of the input matrix. The augmentation less than doubles the input dimension and increases it insignificantly if the nullity r is small.

The input matrices A are kept Hermitian in A-preprocessing for $U = V$ and in preprocessing by augmentation for $U = S$ and $W = W^H$, whereas these restrictions on A-preprocessors UV^H and augmentation do not destroy preprocessing power according to our formal and experimental study

(cf. Remark 3.7, Theorem 4.3, and tests in [54]).

1.4 Approximate nmbs and applications

To model numerical application of our approach, assume that an $n \times n$ ill conditioned input matrix $\tilde{A} = A + E$ of full rank closely approximates an unknown well conditioned singular matrix A of a rank $\rho < n$. Wherever the norm $||E||$ is a small fraction of the smallest positive singular value of the matrix A, the output $\tilde{C}^{-1}U$ of our nmb algorithms closely approximates a nmb(A). Consequently its range closely approximates the *r-tail of the SVD* of the matrix $\tilde{A}$, that is the singular space associated with its r smallest singular values.

With a high probability a properly scaled random A-preprocessor UV^T of rank $r \geq n - \rho$ maps the matrix $\tilde{A}$ into the matrix $\tilde{C} = \tilde{A} + UV^T$ with $\operatorname{cond} \tilde{C}$ of the order of $\operatorname{cond} A \approx \frac{||E||}{||A||} \operatorname{cond} \tilde{A}$, that is these A-preprocessors are likely to work as *A-preconditioners* for computing approximate nmbs. In Section 8 we employ our approximate nmb algorithms to improve such A-preconditioners.

Let us point out some further applications of our approximate nmb algorithms.

- Nonhomogeneous linear systems can be readily reduced to homogeneous ones, and we extend our algorithms respectively in Section 11.5. In the case of an ill conditioned input, the transition to homogeneous linear systems typically improves conditioning but requires to output null vectors with high accuracy. The tradeoff can be attractive because we can apply more steps of iterative refinement but with double rather than extended precison. In this variation of our approach we can fully preserve the Toeplitz matrix structure and as a result simplify the solution of Toeplitz linear systems of equations (see Section 11.5).

- We apply our preconditioning to improve the initialization of Newton's iteration for matrix inversion, which is critical for its acceleration (see Section 11.4).

- If $m \geq n$ and if the $n \times r$ factor Q in the QR factorization of the matrix $\tilde{C}^{-1}U$ closely approximates a nmb(A), then the matrix $\tilde{A} - \tilde{A}QQ^H$ of rank $n - r$ closely approximates the $m \times n$ matrix $\tilde{A}$ (see Section 7.2). By applying these techniques to the Stein (resp. Sylvester) displacement $\tilde{A} - B\tilde{A}C$ (resp. $B\tilde{A} - \tilde{A}C$) for appropriate matrices B and C (such as the matrices of shifts or diagonal scaling), we approximate the displacement with a matrix of a smaller rank with the extension to approximating the matrix $\tilde{A}$ with a structured matrix of a smaller displacement rank (e.g., with a Toeplitz-like or Hankel-like matrix).

- For every eigenpair $(\lambda, \mathbf{y})$ of a matrix A, the eigenvector $\mathbf{y}$ is the null vector of $A - \lambda I$. Having an approximation $\tilde{\lambda}$ to a simple and isolated eigenvalue λ available, we can apply our techniques to approximate the eigenvector $\mathbf{y}$ by the vector $\tilde{C}^{-1}\mathbf{u}$ for the matrix $\tilde{C} = A - \tilde{\lambda}I + \mathbf{v}^T\mathbf{u}$. In this case, the matrix $A - \tilde{\lambda}I$ is ill conditioned. but we can expect that for scaled random vectors $\mathbf{u}$ and $\mathbf{v}$ the matrix $\tilde{C}$ is well conditioned. An alternative way is to choose the vectors $\mathbf{u}$ and $\mathbf{v}$ that make the matrix $\tilde{C}$ more readily invertible than the matrix $A - \lambda I$. E.g., for a tridiagonal matrix M we can yield a 2×2 block diagonal matrix $\tilde{C}$ with tridiagonalstill blocks of half size on its diagonal. These observations serve as the basis for designing effective eigen-solvers in [61] and [68], and we prove that local quadratic convergenece is still preserved in these variations.

- In Section 11.3 we extend them to accelerate root-finding for polynomial and secular equations reduced to eigen-solving for auxiliary structured matrices. We plan to work out similar extension to solving polynomial systems of equations.

1.5 Contents and organization of the paper and its selective reading

We present and analyze in some detail our approach to symbolic and numerical computation of null vectors and nmbs. In Sections 7.2, 8, and 11 and papers [52], [59], [60], [68], [61], and [69] we cover its further variations, extensions, and applications. Otherwise we organize our paper as follows. We cover definitions and some auxiliary results in the next section. In Section 3 we extensively study

randomized A-preprocessing. Section 4 is devoted to randomized preprocessing by augmentation. In Section 5 we recall the known estimates (mostly from [27]) for the perturbation and errors in numerical matrix computations. (We use these results briefly, only to complete our error estimates in Section 6.6. Otherwise the reader can skip them.) Section 6 covers symbolic and numerical computation of nmbs. We present the respective algorithms at length, and an advanced reader may prefer to skip many details. In Section 7 we extend our nmb algorithms to approximate singular spaces. In Section 9 we comment on preserving and exploiting matrix structure in our computations. Section 10 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 and should be cited as his work.

For completeness we study the general case of a rectangular input matrix, but some readers may prefer to examine just the case of a square input matrix, to which Section 3.3 reduces most of our study. A large part of the paper (in particular all our study of the errors and conditioning) can be omitted by those readers who are only interested in the symbolic versions of our algorithms, whereas those readers who are not interested in these versions can skip many details in Section 6.2.

Acknowledgements. We have substantially improved our original draft in response to valuable comments and criticism by the referee. Marc Van Barel's pointers and comments to his Toeplitz solver in [78] were most helpful for our tests covered in Section 10.

2 Definitions and auxiliary results

Hereafter $\mathbb{R}$ (resp. $\mathbb{C}$) denotes the field of real (resp. complex) numbers and "flop" stands for "arithmetic operation".

2.1 General matrices and additive preprocessing

We assume or slightly extend the customary definitions for matrix computations (cf. [25], [27], [73], [74]). This includes the definitions of the Hermitian, unitary (orthonormal), and singular matrices, full-rank and rank deficient matrices, the $k \times k$ identity matrix $I = I_k$, its ith column vectors $\mathbf{e}_i$, $i = 0, \dots, k-1$, the $k \times l$ matrix $0 = 0_{k,l}$ filled with zeros, the transpose A^T of a matrix A and its Hermitian, that is complex conjugate transpose A^H, its rank $\rho = \operatorname{rank} A$, nullity $\operatorname{nul} A = n - \rho$, left nullity $\operatorname{lnul} A = m - \rho = \operatorname{nul}(A^T)$, range $\operatorname{range} A = \{\mathbf{y} : \mathbf{y} = A\mathbf{z}\}$, left range $\{\mathbf{x} : \mathbf{x}^T = \mathbf{w}^T A\}$, null space $N(A) = \{\mathbf{y} : A\mathbf{y} = \mathbf{0}\}$, left null space $LN(A) = \{\mathbf{x} : \mathbf{x}^H A = \mathbf{0}^H\}$, and QR and QRP factorizations. QR factorization of a matrix is unique if the R-factor is a square matrix whose diagonal entries are positive [25, Theorem 5.2.2]. $\operatorname{diag}(B_1, B_2)$ (resp. $\operatorname{diag}(B_i)_{i=1}^k$) denotes the 2×2 (resp. $k \times k$) block diagonal matrix with diagonal blocks B_1 and B_2 (resp. $B_1, \dots, B_k$), whereas (B, C) (resp. $(B_j)_{j=1}^k$) denotes the 1×2 (resp. $1 \times k$) block matrix with blocks B and C (resp. $B_1, \dots, B_k$).

The map $A \Longrightarrow C = A + P$ is A-preprocessing of a matrix A, the matrix C is its A-modification, and the matrix P is its *A-preprocessor* or *APP* (cf. Section 1.3).

2.2 Sparse and structured matrices

Definition 2.1. *A matrix is* sparse *if its entries are mostly zeros according to a fixed criterion specifying the informal notion "mostly" (cf. [14], [15], [33], [62], and the bibliography therein).*

In the rest of this subsection we cover structured matrices of two groups, involved into Example 3.1 and Sections 3.5, 6.1, and 9.

a) **Matrices with small displacement ranks**. $M - AMB$ is the Stein displacement of a matrix M for two fixed operator matrices A and B. (One can similarly work with the Sylvester displacement $AM - MB$, closely linked to the Stein displacement.) $d = \operatorname{dr}_{A,B}(M)$ denotes $\operatorname{rank}(M - AMB)$, called the displacement rank of the matrix M. Next we sketch some relevant definitions and basic

properties assuming operator matrices A and B of shifts and diagonal scaling (cf. [48] and the bibliography therein on elaboration, proofs and further study).

(i) Matrices of Shifts, Diagonal Scaling, and Reflection. We have

$$Z_f^n = fI,\ Z_f \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{pmatrix} = \begin{pmatrix} v_n \\ v_{n-1} \\ \vdots \\ v_2 \\ v_1 \end{pmatrix} \text{ for } Z_f = \begin{pmatrix} 0 & & & & f \\ 1 & \ddots & & & 0 \\ & \ddots & \ddots & & \vdots \\ & & \ddots & 0 & 0 \\ & & & 1 & 0 \end{pmatrix} \tag{2.1}$$

and any scalars f, $v_1, \dots, v_n$. Z_f is the $n \times n$ unit f-circulant matrix, $Z = Z_0 = Z_f - f\mathbf{e}_0\mathbf{e}_{n-1}^T$ is the down shift matrix, and Z_1 is the cyclic shift matrix.

$D_{\mathbf{t}} = \mathrm{diag}(t_i)_{i=1}^n$ is a matrix of diagonal scaling defined by a vector $\mathbf{t} = (t_i)_{i=1}^n$.

$J = (j_{i,k})_{i,k=0}^{n-1}$ is the reflection matrix, $j_{i,k} = 1$ if $i + k = n - 1$, $j_{i,k} = 0$ otherwise, so that $J(v_i)_{i=1}^n = (v_{n+1-i})_{i=1}^n$, $J^2 = I$.

(ii) Four Basic Matrix Structures. The operator matrices A and B define the type of the matrix structure. We have $\mathrm{dr}_{Z_e,Z_f}(T) < 3$, $ef \neq 1$, for Toeplitz matrices $T = (t_{i-j})_{i=0,j=0}^{m-1,n-1}$; $\mathrm{dr}_{Z_e,Z_f^T}(H) < 3$, $ef \neq 1$, for Hankel matrices $H = (h_{i+j})_{i=0,j=0}^{m-1,n-1}$; $\mathrm{dr}_{D_{\mathbf{t}},Z_f}(V(\mathbf{t})) = 1$ and $\mathrm{dr}_{Z_f^T,D_{\mathbf{t}}}(V(\mathbf{t})^T) = 1$, $f \neq t_i^n$, for nonzero Vandermonde matrices $V(\mathbf{t}) = (t_i^j)_{i=0,j=0}^{m-1,n-1}$, and $\mathrm{dr}_{D_{\mathbf{s}},D_{\mathbf{t}}}(C(\mathbf{s},\mathbf{t})) = 1$ for Cauchy matrices $C(\mathbf{s},\mathbf{t}) = (\frac{1}{s_i - t_j})_{i=0,j=0}^{m-1,n-1}$, $s_i t_j \neq 0$ for all pairs (i,j). These are the four most popular classes of matrices with small displacement ranks. Matrices are said to have structures of Toeplitz, Hankel, Vandermonde and Cauchy types, respectively, if their displacement ranks are small assuming the above operator matrices. Hereafter we write $\mathcal{T}_d$, $\mathcal{H}_d$, $\mathcal{V}_d(\mathbf{t})$, and $\mathcal{C}_d(\mathbf{s},\mathbf{t})$ to denote the respective classes of matrices having small displacement ranks d. They include Sylvester, companion, Loewner, Pick, and other celebrated classes of structured matrices. We also call the matrices in $\mathcal{T}_d$ Toeplitz-like and in $\mathcal{H}_d$ Hankel-like matrices.

(iii) Structured Matrix Computations via Operations with Displacements. The displacement $M - AMB$ of rank d can be represented as the sum of d outer products $\mathbf{g}_k\mathbf{h}_k^T$, $k = 1, \dots, d$,

$$M - AMB = \sum_{k=1}^{d} \mathbf{g}_k\mathbf{h}_k^T. \tag{2.2}$$

Unless the linear operator $M \Longrightarrow M - AMB$ degenerates, one can express the original matrix M through the entries of these vectors, so that an $n \times n$ matrix M can be readily expressed via $2dn$ parameters. For $d \ll n$ this is more economical than its representation with the n^2 entries. Furthermore matrix transposition, addition, multiplication by a constant and a vector, pairwise multiplication and inversion can be expressed economically in terms of the respective operations with their displacements. In particular one can multiply an $n \times n$ matrix M by a vector in $O(dn \log n)$ flops for M in $\mathcal{T}_d$ and $\mathcal{H}_d$ and in $O(dn \log^2 n)$ flops for M in $\mathcal{V}_d(\mathbf{t})$ and $\mathcal{C}_d(\mathbf{s},\mathbf{t})$, and one can solve a nonsingular linear system of n equations with a matrix from any of the four classes by using $O(d^2 n \log^2 n)$ flops.

(iv) Transformation of Matrix Structure via Transformation of Displacements. Multiplication by the reflection matrix J and appropriate Vandermonde matrices and their transposes transforms the classes $\mathcal{T}_d$, $\mathcal{H}_d$, $\mathcal{V}_d(\mathbf{t})$, and $\mathcal{C}_d(\mathbf{s},\mathbf{t})$ into each other all ways. Here are some respective maps, $J\mathcal{T}_d = \mathcal{T}_d J = \mathcal{H}_d$, $J\mathcal{H}_d = \mathcal{H}_d J = \mathcal{T}_d$, $V(\mathbf{t})\mathcal{T}_d \subset \mathcal{V}_{d+1}(\mathbf{t})$, $V(\mathbf{t})^T\mathcal{V}_d(\mathbf{t}) \subset \mathcal{T}_{d+1}$, $\mathcal{C}_d(\mathbf{s},\mathbf{t})V(\mathbf{t}) \subset \mathcal{V}_{d+1}(\mathbf{s})$, $\mathcal{V}_d(\mathbf{s})V(\mathbf{t})^T \subset \mathcal{C}_{d+1}(\mathbf{s},\mathbf{t})$, and $\mathcal{C}_d(\mathbf{s},\mathbf{t})C(\mathbf{t},\mathbf{u}) \subset \mathcal{C}_{d+1}(\mathbf{s},\mathbf{u})$. (We also have $J\mathbb{T} = \mathbb{T}J = \mathbb{H}$ and $J\mathbb{H} = \mathbb{T}J = \mathbb{T}$ for the classes $\mathbb{T}$ of Toeplitz and $\mathbb{H}$ of Hankel matrices.) It follows that any algorithm for the inversion of the matrices in one of these classes or for solving linear systems with the matrices

of this class can be immediately extended to the same operations with the inputs in the three other classes.

Remark 2.1. The Method of Displacement Transformation. *The above techniques of displacement transformation for algorithm design was proposed in [44] and was refined in [22], [24], and [26] to devise practical algorithms for Toeplitz, Hankel, Toeplitz-like, and Hankel-like linear systems of equations. The algorithms first map an input from the class* $\mathbb{T}$, $\mathcal{T}_d$, $\mathbb{H}$ *or* $\mathcal{H}_d$ *into a matrix with the structure of Vandermonde or Cauchy type. Then they apply fast Gaussian elimination with partial pivoting (GEPP) by exploiting the input structure. Direct application of GEPP to the four former matrix classes destroys the structure and is as slow as in the case of general input matrices, whereas one can run GEPP fast on the input classes* $\mathcal{V}_d(\mathbf{t})$ *and* $\mathcal{C}_d(\mathbf{s},\mathbf{t})$. *In other words, the above map extends the fast and numerically stable solution with GEPP from inputs in* $\mathcal{V}_d(\mathbf{t})$ *and* $\mathcal{C}_d(\mathbf{s},\mathbf{t})$ *to the highly important input classes* $\mathbb{T}$, $\mathcal{T}_d$, $\mathbb{H}$, *and* $\mathcal{H}_d$. *In the algorithms in [22], [24], and [26] Vandermonde multipliers are specialized to the matrices of Discrete Fourier transform, and then the map is slightly further refined versus the original maps in [44]. The approach has been further advanced in [9], [50], and [70].*

(v) Toeplitz-like, Toeplitz, and f*-Circulant Matrices.* Let us supply more details on the basic classes of Toeplitz-like, Toeplitz, and f-circulant matrices.

For a scalar f and a column vector $\mathbf{v} = (v_i)_{i=0}^{n-1}$ the matrix polynomials $Z_f(\mathbf{v}) = \sum_{i=0}^{n-1} v_i Z_f^i$ form the algebra of *f-circulant matrices* in the class of Toeplitz matrices. For $f = 0$ this is the algebra of lower triangular Toeplitz matrices for $f = 0$. In both cases the matrix $Z_f(\mathbf{v})$ is defined by its first column vector $\mathbf{v}$. Due to the factorization of f-circulant matrices in [10], they can be multiplied and inverted in $O(n \log n)$ flops based on FFT. These estimates also hold for triangular Toeplitz matrices $Z_0(\mathbf{v})$.

f-circulant matrices are called *circulant* for $f = 1$ and *skew circulant* for $f = -1$.

Now fix two scalars e and f, $ef \neq 1$, e.g., $e = f = 0$ or $f = -f = 1$. Then M is an $n \times n$ Toeplitz matrix (resp. Toeplitz-like matrix of displacement rank d for the operator matrices $A = Z_e$ and $B = Z_f^T$) if and only if it can be nonuniquely represented as the sum $Z_e(\mathbf{u}) + Z_f(\mathbf{v})^T$ for two vectors $\mathbf{u}$ and $\mathbf{v}$ (resp. as the sum

$$(1 - ef)M = \sum_{k=1}^{d} Z_e(\mathbf{g}_k) Z_f(\mathbf{h}_k)^T \tag{2.3}$$

for d pairs of vectors $(\mathbf{g}_k, \mathbf{h}_k)$, which are precisely the vectors in (2.2) for $A = Z_e$ and $B = Z_f^T$). There are similar expressions for the matrix classes $\mathcal{H}_d = J\mathcal{T}_d$, $\mathcal{V}_d(\mathbf{t})$, and $C_d(\mathbf{s},\mathbf{t})$ (cf. [48, Sections 4.3 and 4.4]).

b) **Banded matrices** $(b_{i,j})_{i,j}$ with a *lower bandwidth* l and an *upper bandwidth* u, that is such that $b_{i,j} = 0$ where $i - j > l$ or $j - i > u$ (cf. [25, Section 1.2.1]), their inverses, and more generally the **rank structured matrices**.

Definition 2.2. *(Cf. [20, Section 1], [81, Definition 3], [82, Definition 2], [83], [84].) A lower (resp. upper) rank* $l = l_M$ *(resp.* $u = u_M$*) of a matrix* M *is the maximal rank of any its submatrix lying entirely below (resp. above) its diagonal. Its rank pair* $(l,u)_M = (l_M, u_M)$ *is the pair of its lower rank* l *and upper rank* u. *An* $m \times n$ *matrix* M *is rank structured if* $\max\{l(M), u(M)\} \ll \min\{m,n\}$.

Such an $n \times n$ matrix can be generated with $O((l+u+1)n)$ parameters and multiplied by a vector in $O((l+u+1)n)$ flops. One can solve a nonsingular linear system of equations with such a matrix in $O((l+u+1)^2 n)$ flops (see [20]).

2.3 Null vectors, nmbs, and annihilators

A matrix B of full column rank is a *matrix basis* for its range. A null vector, a null basis, and a null matrix basis for a matrix A are a vector in, a basis for, and a matrix basis for its nonempty

null space $N(A)$, respectively (cf. Section 1.1). For an $m \times n$ matrix A with a positive nullity r, its $\mathrm{nmb}(A)$ is an $n \times r$ matrix where $r < n$ unless $A = 0$. A matrix H is a *complete annihilator* of a matrix A if $\mathrm{range}\, H = N(A)$. We use the abbreviations nmb, $\mathrm{nmb}(A)$, and $\mathrm{ca}(A)$ and define similar concepts for the left null space $LN(A)$. Clearly, every $\mathrm{nmb}(A)$ is a $\mathrm{ca}(A)$. Conversely, given a $\mathrm{ca}(A)$, we can compute a $\mathrm{nmb}(A)$ from LUP or QR factorization of the matrix A, but the following fact can be preferred as the basis in the case of structured matrices A.

Fact 2.1. *Suppose H is a $\mathrm{ca}(A)$. Then*
(a) H is a $\mathrm{nmb}(A)$ if and only if $\mathrm{nul}\, H = 0$ and
(b) HY is a $\mathrm{nmb}(A)$ if X is a $\mathrm{ca}(H)$ and if (X, Y) is a nonsingular matrix.

Proof. Part (a) is trivial, and clearly $AHY = 0$ because H is a $\mathrm{ca}(A)$. Now suppose $HY\mathbf{x} = \mathbf{0}$ for a nonzero vector $\mathbf{x}$. Then $Y\mathbf{x} = X\mathbf{y}$ for some vector $\mathbf{y}$ because X is a $\mathrm{ca}(H)$. Therefore $(X, Y)\mathbf{w} = \mathbf{0}$ for $\mathbf{w}^T = (\mathbf{x}^T, -\mathbf{y}^T)$. It follows that $\mathbf{w} = \mathbf{0}$ (and thus $\mathbf{x} = \mathbf{0}$) because (X, Y) is a nonsingular matrix. Contradiction to the assumption that $\mathbf{x} \neq \mathbf{0}$ implies part (b). □

2.4 SVDs and generalized inverses

For an $m \times n$ matrix A of a rank ρ, its *full Singular Value Decomposition* (hereafter *SVD* or *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, $S^H S = S^H S = I_m$, $T^H T = TT^H = I_n$; $\Sigma = \mathrm{diag}(\Sigma^{(\rho)}, 0_{m-\rho,n-\rho})$ is an $m \times n$ matrix; $\Sigma^{(\rho)} = \mathrm{diag}(\sigma_j)_{j=1}^{\rho}$; $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\rho > 0$, $\sigma_j = 0$ for $j > \rho$ and $\sigma_j = +\infty$ for $j < 1$. The scalars $\sigma_j = \sigma_j(A)$ for $j \geq 1$ are the *singular values* of the matrix A. The vectors $\mathbf{s}_j^H$ for $j = 1, \ldots, m$ and $\mathbf{t}_j$ for $j = 1, \ldots, n$ are the associated left and right *singular vectors*, respectively. $\mathbb{S}_{k,r} = \mathrm{range}((s_j)_{j=k+1}^{k+r})$ (resp. $\mathbb{T}_{k,r} = \mathrm{range}((t_j)_{j=k+1}^{k+r})$) is the left (resp. right) singular space associated with the singular values $\sigma_{k+1}, \ldots, \sigma_{k+r}$ provided $k > 0$, $\sigma_k > \sigma_{k+1}$, and either $\sigma_{k+r} > \sigma_{k+r+1}$ or $k + r = m$ (resp. $k + r = n$). If $k + r = n \leq m$, then $\mathbb{T}_{k,r}$ is the (right) *r-tail* and $\mathbb{T}_{0,n-r}$ is the (right) $(n - r)$-*head* of the SVD.

Fact 2.2. *$\sigma_j(W^H) = \sigma_j(W)$ for all matrices W and integers j.*

A matrix $X = A^{(I)}$ is a left (resp. right) inverse of a matrix A if $XA = I$ (resp. $AX = I$). An $m \times n$ matrix of a rank ρ has a left (resp. right) inverse if and only if $\rho = n$ (resp. $\rho = m$). Such an inverse is unique if and only if the matrix A is nonsingular, and then $A^{(I)} = A^{-1}$.

$A^+ = \sum_{j=1}^{\rho} \sigma_j^{-1} \mathbf{t}_j \mathbf{s}_j^H$ denotes the Moore–Penrose generalized inverse (also called pseudo inverse) of an $m \times n$ matrix A of a rank ρ. $A^+ = (A^H A)^{-1} A^H$ is a left inverse $A^{(I)}$ if $m \geq n = \rho$, $A^+ = A^H (AA^H)^{-1}$ is a right inverse $A^{(I)}$ if $m = \rho \leq n$, and $A^+ = A^{-1}$ if $m = n = \rho$. A^{+H} (resp. A^{-H}) denotes the matrix $(A^+)^H = (A^H)^+$ (resp. $(A^{-1})^H = (A^H)^{-1}$).

2.5 The Sherman–Morrison–Woodbury formula and its modifications

Theorem 2.1. *Assume that $A \in \mathbb{C}^{m \times n}$, $U \in \mathbb{C}^{m \times r}$, $V \in \mathbb{C}^{n \times r}$, $C = A + UV^H \in \mathbb{C}^{m \times n}$, and the matrices A and C have full rank. Then the matrix $G = I_r - V^H C^+ U$ is nonsingular and*

$$A^+ = C^+ + C^+ U G^{-1} V^H C^+. \tag{2.4}$$

Proof. See [52, Section 4.2]. □

In the case where $m = n$ and A is a square matrix, we arrive at the Sherman–Morrison–Woodbury classical formula [25, page 50],

$$A^{-1} = C^{-1} + C^{-1} U G^{-1} V^H C^{-1}. \tag{2.5}$$

Corollary 2.1. *Under the assumptions of Theorem 2.1, we have* $\mathrm{range}(C^+ U) = \mathrm{range}(A^+ U)$.

(Hereafter we use the abbreviation *SMW formulae* for both expressions (2.5) and (2.4).)

Next we keep dealing with the matrices $U \in \mathbb{C}^{m\times r}$, $V \in \mathbb{C}^{n\times r}$, and $A \in \mathbb{C}^{m\times n}$ and assume that the matrices A and $C_- = A^+ + UV^H$ have full rank. Apply Theorem 2.1 to the matrices A and C_- (replacing the matrix C). Obtain that the matrix $H = I_r + V^H AU \in \mathbb{C}^{r\times r}$ (replacing the matrix G in Theorem 2.1) is nonsingular and arrive at the *dual SMW formula* (cf. [52])

$$(C_-)^+ = A - AUH^{-1}V^H A. \tag{2.6}$$

It follows that

$$A^+ = C_- - UV^H = (A - AUH^{-1}V^H A)^+ - UV^H. \tag{2.7}$$

For $m = n$ we have the expressions

$$(C_-)^{-1} = A - AUH^{-1}V^H A, \quad A^{-1} = (A - AUH^{-1}V^H A)^{-1} - UV^H. \tag{2.8}$$

2.6 Norms, condition numbers, and singular values

$||A||$ is the 2-norm of a matrix A, which is normalized if $||A|| = 1$. $\operatorname{cond} A = ||A||\,||A^+|| = \frac{\sigma_1(A)}{\sigma_\rho(A)}$ is the condition number of a matrix A of a rank ρ. Effective norm and condition estimators can be found in [12], [25, Sections 2.3.2, 2.3.3, 3.5.4, and 12.5], [27, Chapter 15], and [73, Section 5.3].

Fact 2.3. *For $A = (a_{i,j})_{i,j=1}^{m,n}$ we have $||A||/\sqrt{mn} \le \max_{i,j=1}^{m,n} |a_{i,j}| \le ||A|| = ||A^H||$.*

The minimax characterization [25, Theorem 8.6.1] relates the norms and singular values,

$$\sigma_j = \max_{\dim(\mathbb{S})=j} \min_{\mathbf{x}\in\mathbb{S},\ ||\mathbf{x}||=1} ||A\mathbf{x}|| \tag{2.9}$$

for $j = 1, 2, \dots, n$ and for linear spaces $\mathbb{S}$. Hence for an $m \times n$ matrix A of a rank ρ we have $\sigma_\rho = 1/||A^+||$, $\sigma_1 = \max_{||\mathbf{x}||=1} ||A\mathbf{x}|| = ||A||$, $\sigma_n = \min_{||\mathbf{x}||=1} ||A\mathbf{x}||$.

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.

An APP P is an *A-preconditioner* of a matrix A if $\operatorname{cond}(A + P) \ll \operatorname{cond} A$.

An $m \times n$ matrix A has *numerical rank* $\rho = \operatorname{nnul} A$ and *numerical nullity* $\operatorname{nnul} A = l - \rho$ for $l = \min\{m, n\}$ if this matrix has exactly ρ singular values (counting their multiplicities) that exceed $\delta\sigma_1(A)$ for a fixed small positive δ and if either $\rho < l$ and the ratio $\frac{\sigma_\rho(A)}{\sigma_{\rho+1}(A)}$ is small or $\rho = l$.

2.7 The singular values of submatrices and matrix products

The two following theorems are used in the proofs of Theorems 3.7 and 3.8 in Section 3.5. In fact parts b) and c) of Theorems 2.2 are not used there but naturally complement part a).

Theorem 2.2. *Let p, q, m, and n be four positive integers such that $1 \le p \le m$ and $1 \le q \le n$ and let A_0 be a $p \times q$ submatrix of an $m \times n$ matrix A. Then*

a) $\sigma_j(A) \ge \sigma_j(A_0)$ for $j = 1, 2, \dots, \min\{p, q\}$,

b) there is a $p \times n$ block submatrix $\widehat{A}$ of the matrix A such that $\lceil m/p\rceil^{1/2}||\widehat{A}|| \ge ||A||$, and

c) there is a $p \times q$ block submatrix $\tilde{A}$ of the matrix A such that $\lceil m/p\rceil^{1/2}\lceil n/q\rceil^{1/2}||\tilde{A}|| \ge ||A||$.

Proof. a) A_0 is a submatrix of a certain $p \times n$ submatrix $\bar{A}_0$ of the matrix A. Then (2.9) implies that $\sigma_j(A) \ge \sigma_j(\bar{A}_0)$ for all j and similarly $\sigma_j(\bar{A}_0^H) \ge \sigma_j(A_0^H)$, whereas $\sigma_j(\bar{A}_0) = \sigma_j(\bar{A}_0^H)$ and $\sigma_j(A_0) = \sigma_j(A_0^H)$ for all j, due to Fact 2.2.

b) Recall that $\sigma_j(A) = ||A\mathbf{x}^{(j)}||$ for some vectors $\mathbf{x}^{(j)}$ such that $||\mathbf{x}^{(j)}|| = 1$ and for $j = 1, 2, \dots$ due to minimax characterization (2.9). Write $\mathbf{a}_i^T = \mathbf{e}_i^T A$ for $i = 0, \dots, m-1$, $\mathbf{a}_i^T = \mathbf{0}_i^T$ for $i \ge m$, and $\tilde{A}_j = (\mathbf{a}_i^T)_{i=jl}^{jl+l-1}$ for $j = 0, \dots, r-1$ and $r = \lceil m/p\rceil$. Observe that $||A\mathbf{x}||^2 = \sum_{i=0}^{r-1} ||\tilde{A}_j\mathbf{x}||^2 = \sum_{i=0}^{m-1} |\mathbf{a}_j^T\mathbf{x}|^2$ for all vectors $\mathbf{x}$. Suppose the norm $||\tilde{A}_j\mathbf{x}^{(1)}||$ is maximum for $j = h$ and deduce that $||A|| = \sigma_1(A) \le \lceil m/p\rceil^{1/2}||\tilde{A}_h\mathbf{x}^{(1)}|| \le \lceil m/p\rceil^{1/2}||\tilde{A}_h||$. Write $\widehat{A} = \tilde{A}_h$ and obtain part b).

c) Now apply part b) to the $n \times p$ matrix $\tilde{A}_h^T$ replacing the $m \times n$ matrix A and obtain $q \times p$ block submatrix B such that $\lceil n/q \rceil^{1/2} ||B|| \geq ||\tilde{A}_h^T|| = ||\tilde{A}_h||$. Therefore $\lceil m/p \rceil^{1/2} \lceil n/q \rceil^{1/2} ||B|| \geq ||A||$. Write $\tilde{A} = B^T$ and obtain part c) because B^T is a $p \times q$ block submatrix of the matrix A and $||B|| = ||B^T||$. □

Theorem 2.3. *[60]. Let $A \in \mathbb{C}^{m \times r}$ and $B \in \mathbb{C}^{r \times n}$ and write $r_A = \operatorname{rank} A$, $r_B = \operatorname{rank} B$, $r_- = \min\{r_A, r_B\}$ and $r_+ = \max\{r_A, r_B\}$. Let $r_+ = r$. (In particular this holds if at least one of the matrices A and B is nonsingular.) Then* $\operatorname{rank}(AB) = r_-$, $\sigma_{r_-}(AB) \geq \sigma_{r_A}(A)\sigma_{r_B}(B)$ *and* $\operatorname{cond}(AB) \leq (\operatorname{cond} A) \operatorname{cond} B$.

Remark 2.2. $\operatorname{cond}(AB)$ *can be arbitrarily large even for $m \times r$ unitary matrices A and B^H if $m > r$.*

2.8 Random sampling, random matrices, and Gaussian random variables

$|\Delta|$ is the cardinality of a set Δ. *Random sampling* of elements from a set Δ is their selection from this set at random, independently of each other, and under the uniform probability distribution on the set. A matrix is *random* if its entries are randomly sampled from a fixed set Δ, e.g., the set of all double precision numbers with the exponents in a fixed range, for numerical computations.

The next definition and lemma are only used in Section 3.5 and Theorem 3.12 in Section 3.6.

Definition 2.3. *$F_X(y) = Probability\{X \leq y\}$ for a real random variable X is the* cumulative distribution function (CDF) *of X evaluated at y. $F_A(y) = F_{\sigma_l(A)}(y)$ for an $m \times n$ matrix A and an integer $l = \min\{m, n\}$. A matrix (resp. vector) is a* Gaussian random matrix *(resp. vector) with a mean μ and a variance σ^2 if it is filled with independent Gaussian random variables, all having the same mean μ and variance σ^2. If $\mu = 0$ and $\sigma^2 = 1$, this is a* standard Gaussian random matrix *(resp. vector). $F_{\mu,\sigma}(y) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{y} \exp(-\frac{(x-\mu)^2}{2\sigma^2})dx$ is the CDF for a Gaussian random variable with a mean μ and a variance σ^2. $\Phi_{\mu,\sigma}(y) = F_{\mu,\sigma}(y) - F_{\mu,\sigma}(-y)$ for $y \geq 0$.*

Lemma 2.1. *For positive scalars y, y_1, and y_2 we have*
$F_X(y) \leq F_{X_1}(y_1) + F_{X_2}(y_2)$ if $X \leq \min\{X_1 y/y_1, X_2 y/y_2\}$,
$1 - F_X(y) \leq 2 - F_{X_1}(y_1) - F_{X_2}(y_2)$ if $X \geq \max\{X_1 y/y_1, X_2 y/y_2\}$.

3 A-preprocessing and randomization

3.1 APPs and nmbs

Theorem 3.1. *For three integers m, n and r, $m \geq n \geq r > 0$, and for a pair of matrices U of size $m \times r$ and V of size $n \times r$, assume that (a) A is an $m \times n$ matrix of a rank ρ, (b) the matrix $C = A + UV^H$ has full rank n, and (c) $C^{(I)}$ is a left inverse of C, that is $C^{(I)}C = I$. Then*

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

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

Furthermore if

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

then (cf. Corollary 2.1)

$$B = C^{(I)}U \text{ is a } \operatorname{nmb}(A), \tag{3.4}$$

$$V^H C^{(I)} U = I_r. \tag{3.5}$$

Proof. Bound (3.1) follows because $\operatorname{rank}(A + UV^H) \leq \operatorname{rank} A + \operatorname{rank}(UV^H)$, $\operatorname{rank} A = \rho$, and $\operatorname{rank}(UV^H) \leq \operatorname{rank} U$.

Now let $\mathbf{y} \in N(A)$. Then $C\mathbf{y} = (A + UV^H)\mathbf{y} = UV^H\mathbf{y}$, and therefore

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

This proves (3.2).

(3.4) follows from (3.2) and (3.3) because $\text{rank}(C^{(I)}U) \le r$.

To prove (3.5), first premultiply by V^H equation (3.6) and obtain that $(V^HC^{(I)}U - I_r)V^H\mathbf{y}$ for any vector $\mathbf{y}$ in $N(A)$. Next recall equation (3.4) and deduce that $(V^HC^{(I)}U - I_r)V^HC^{(I)}U = 0$. Now (3.5) follows unless the matrix $V^HC^{(I)}U$ is singular, but if it is, then $V^HC^{(I)}U\mathbf{z} = \mathbf{0}$ for some nonzero vector $\mathbf{z}$. Let us write $\mathbf{w} = C^{(I)}U\mathbf{z}$, so that $V^H\mathbf{w} = \mathbf{0}$ and $\mathbf{w} \in \text{range}(C^{(I)}U) = N(A)$. 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 since $m \ge n$ conclude that $\mathbf{w} = \mathbf{0}$. Consequently, $\mathbf{z} = \mathbf{0}$ because the matrix $C^{(I)}U$ has full rank. □

Corollary 3.1. *Under assumptions (a)–(c) of Theorem 3.1, let $B = C^{(I)}U$. (i) Then BX is a* $\text{ca}(A)$ *if X is a* $\text{ca}(AB)$. *(ii) Furthermore X is a* $\text{ca}(AB)$ *if BX is a* $\text{ca}(A)$ *and if* $\text{rank}\, B = r$.

Proof. (i) $A(BX) = (AB)X = 0$ if X is a $\text{ca}(AB)$. Conversely, let $A\mathbf{u} = \mathbf{0}$. Then $\mathbf{u} = B\mathbf{v}$ for some vector $\mathbf{v}$ in virtue of (3.2). Therefore $AB\mathbf{v} = A\mathbf{u} = \mathbf{0}$. It follows that $\mathbf{v} = X\mathbf{z}$ for some vector $\mathbf{z}$ because X is a $\text{ca}(AB)$. Consequently $\mathbf{u} = B\mathbf{v} = BX\mathbf{z}$.

(ii) $(AB)X = A(BX) = 0$ if BX is a $\text{ca}(A)$. Conversely, let $AB\mathbf{u} = A(B\mathbf{u}) = \mathbf{0}$. Then $B\mathbf{u} = BX\mathbf{v}$ for some vector $\mathbf{v}$ because BX is a $\text{ca}(A)$. Therefore $\mathbf{u} = B\mathbf{v}$ since $\text{rank}\, B = r$. □

3.2 The 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 $LN(A) = N(A^T)$. Specifically, assume that $U \in \mathbb{C}^{m\times r}$, $V \in \mathbb{C}^{n\times r}$, $C = A + UV^H$, $r \ge \text{rank}\, U \ge \text{lnul}\, A$, and $\text{rank}\, C = m$ and write $B_{\text{left}} = V^HC^{(I)}$. Then $LN(A)$ is a subspace of the left range of the matrix B_{left} and furthermore YB_{left} is a left $\text{ca}(A)$ if Y is a left $\text{ca}(B_{\text{left}}A)$, whereas Y is a left $\text{ca}(B_{\text{left}}A)$ if YB_{left} is a left $\text{ca}(A)$ and if $\text{rank}\, B_{\text{left}} = r$. If $\text{rank}\, V = r = \text{lnul}\, A$, then $LN(A)$ is precisely the left range of the matrix B_{left}.

Seeking a pair of 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 $G = I_r - V^HC^{(I)}U$, also called Gauss transform [52] (cf. Section 6.3 on aggregation techniques).

Theorem 3.2. *Suppose $A \in \mathbb{C}^{m\times n}$, $C = A + UV^H \in \mathbb{C}^{m\times n}$, $U \in \mathbb{C}^{m\times r}$, $V \in \mathbb{C}^{n\times r}$,* $\text{rank}\, C = \min\{m, n\} > r$, *and $C^{(I)}$ is a right or left inverse of C, that is $CC^{(I)} = I$ or $C^{(I)}C = I$. Write $G = I_r - V^HC^{(I)}U$. Then $V^HC^{(I)}A = GV^H$ (and therefore $G = V^HC^{(I)}A(V^H)^{(I)}$ and $LN(G) \subseteq LN(V^HC^{(I)}A)$) if $m \ge n$, whereas $AC^{(I)}U = UG$ (and therefore $G = U^{(I)}AC^{(I)}U$ and $N(G) \subseteq N(AC^{(I)}U)$) if $m \le n$.*

Proof.

$$V^HC^{(I)}A = V^HC^{(I)}(C - UV^H) = V^H - V^HC^{(I)}UV^H = GV^H \text{ if } m \ge n,$$
$$AC^{(I)}U = (C - UV^H)C^{(I)}U = U - UV^HC^{(I)}U = UG \text{ if } m \le n.$$

□

Corollary 3.2. *Under the assumptions of Theorem 3.2, let $m = n$. Then $N(AC^{-1}U) = N(G)$ if the matrix U has full rank, whereas $LN(V^HC^{-1}A) = LN(G)$ if the matrix V has full rank.*

Proof. If $G\mathbf{x} = \mathbf{0}$, then $UG\mathbf{x} = AC^{(I)}U\mathbf{x} = \mathbf{0}$. Conversely, suppose $AC^{(I)}U\mathbf{x} = UG\mathbf{x} = \mathbf{0}$. Then $G\mathbf{x} = \mathbf{0}$ because U is a matrix of full rank. This proves that $N(AC^{-1}U) = N(G)$. Equation $LN(V^HC^{-1}A) = LN(G)$ is proved similarly. □

3.3 From rectangular to square inputs

Our nmb algorithms are simpler and more stable numerically in the case of square input matrices. We can always shift to the Hermitian square input A^HA satisfying $N(A) = N(A^HA)$, but $\text{cond}(A^HA) = (\text{cond}\, A)^2$, and so next we show some alternatives.

Given an $m \times n$ matrix A for $m > n$, we can represent it as the sum $A = \sum_{i=1}^h A_i$ where $A_i = (0, B_i^T, 0)^T$ and B_i are $k_i \times n$ matrices for $i = 1, \dots, h$, $\sum_{i=1}^h k_i \ge m$. Then $N(A) = \cap_{i=1}^h N(B_i)$, and we can also employ the following result [25, Theorem 12.4.1].

Lemma 3.1. *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 $N(A) = N(B^H A)$ for a pair of $m \times n$ matrices A and B where $m \le n$ and B is a matrix of full rank.*

The matrices A and BA share their singular values and right singular spaces if $B^H B = I$.

We have $B^T A = \begin{pmatrix} A \\ 0 \end{pmatrix}$ for $B = (I_m, 0)$, whereas $B^T A = \begin{pmatrix} 0 \\ A \end{pmatrix}$ for $B = (0, I_m)$).

3.4 Random APPs against rank deficiency

Theorem 3.1 defines a nmb$(A) = C^{(I)}U$ if an A-modification $C = A + UV^H$ has full rank for the matrices U and V of the minimum rank. Next we show that the full rank property holds with a probability close to one if U and V are random matrices of sufficiently large sizes. We rely on the following lemma.

Lemma 3.2. *[17] (cf. also [72], [88]). For a 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 $d/|\Delta|$.*

Corollary 3.3. *Let an $m \times n$ matrix M be filled with random entries sampled from a set Δ. Let $l = \min\{m, n\}$. Then the matrix has full rank with a probability of at least $1 - \frac{l}{|\Delta|}$.*

Proof. Clearly an $m \times n$ matrix M has full rank if its entries are indeterminates. The matrix is rank deficient if and only if $\det M_l = 0$ for all the $l \times l$ sumbatrices M_l. Such determinants are polynomials of degrees at most l in the entries, and so the corollary follows from Lemma 3.2. □

Theorem 3.3. *Assume five positive integers m, n, q, ρ, and r such that $\rho \le n \le m$ and $q = \min\{n, r + \rho\}$, a set Δ of cardinality $|\Delta|$ in a ring $\mathcal{R}$ with at least $|\Delta|$ elements, and four matrices, $A \in \mathcal{R}^{m \times n}$ of rank ρ, U in $\mathcal{R}^{m \times r}$, V in $\mathcal{R}^{n \times r}$, and $C = A + UV^T$. Then*

a) $\operatorname{rank} C \le q$,

b) *Probability*$\{\operatorname{rank} C = q\} \ge 1 - \frac{2r}{|\Delta|}$ *where the entries of both matrices U and V have been randomly sampled from the set Δ as well as where the entries of the matrix U have been randomly sampled from this set and $V = U$,*

c) *Probability*$\{\operatorname{rank} C = q\} \ge 1 - \frac{r}{|\Delta|}$ *if the matrix U (resp. V) is fixed and has full rank r and if the entries of the matrix V (resp. U) have been randomly sampled from the set Δ.*

Proof. a) Combine the relationships $\operatorname{rank} C \le \operatorname{rank} A + \operatorname{rank}(UV^T)$, $\operatorname{rank} A = \rho$, and $\operatorname{rank}(UV^T) \le \operatorname{rank} U \le r$.

b) Clearly $\operatorname{rank} C = q$ provided the entries of the matrix $C = A + UV^H$ are bilinear functions in the indeterminate entries of the matrices U and V. Let $C_q = C_q(U, V)$ denote its $q \times q$ submatrix of rank q. Then $\det C_q$ is a nonvanishing polynomial of a total degree of at most $2r$ in the entries of the matrices U and V. The matrix C is rank deficient if and only if this polynomial vanishes, which occurs on a variety of a lower dimension. Now part b) follows from Lemma 3.2.

Part c) is proved similarly to part b). □

Remark 3.1. *Theorem 3.3 implies that the matrix C has rank q with a high probability provided the matrices U and V have random entries sampled from a set of a large cardinality.*

Remark 3.2. *Theorem 3.3 and Corollary 3.3 can be extended to the case where the matrices M, U and V are endowed with the displacement or rank structure as long as the determinants involved in the proofs of the theorem and the corollary do not vanish identically. See [67] on the respective study.*

3.5 Extremal singular values of random matrices and of their products with fixed matrices

Gaussian random matrices (cf. Definition 2.3) are well conditioned with a high probability [13], [19], and even perturbations by such a random matrix A is expected to make a matrix M well conditioned unless the ratio $||M||/||A||$ is small or large [75]. Next we specify and then extend the respective known estimates using a constant $c \le 2.35$ from [75].

Theorem 3.4. *Assume an $m \times n$ matrix A filled with d random variables $X_1, \dots, X_d$. ($d = m+n-1$ for random Toeplitz and Hankel matrices A, $d = mn$ for random general matrix A.) Write $F_-(y) = \min_{i=1}^{d} F_{|X_i|}(y)$ for $y \ge 0$. Then a) $F_-(y) = \Phi_{\mu,\sigma}(y)$ where $X_1, \dots, X_d$ are Gaussian random variables with a mean μ and a variance σ, whereas b) $F_-(y) = y/a$ (resp. $F_-(y) = (y/a)^2$) where $0 \le y \le a$ and the random variables $X_1, \dots, X_d$ are uniformly distributed on the real line segments $[-a, a]$ or $[0, a]$ (resp. the circle $\{x : |x| \le a\}$ on the complex plane or a sector of this circle). Furthermore we have c) $1 - F_{||A||}(y) \le (1 - F_-(y/\sqrt{mn}\,))d$, which is a trivial bound unless $F_-(y/\sqrt{mn}\,) > 1 - 1/d$, and d) $F_{||A||}(y) \ge (F_-(y/\sqrt{mn}\,))^d$ if the d random variables are independent of each other.*

Proof. Parts a) and b) are immediately verified. Part d) follows because Fact 2.3 implies that $||A|| \le y$ if $|X_i| \le y/\sqrt{mn}$ for all i. Apply Lemma 2.1 to deduce part c). □

Quite good estimates are known for the CDFs $F_{||A||}(y)$ and $F_{A+M}(y)$ for a fixed matrix M and a Gaussian random matrix A with the mean zero and a variance σ^2. Moreover these estimates can be extended to the matrices $\bar{U}(A+M)\bar{V}$ where $\bar{U}$ and $\bar{V}$ are unitary matrices (because $\sigma_j(\bar{U}(A+M)\bar{V}) = \sigma_j(A+M)$ for all j) and even where $\bar{U}$ and $\bar{V}$ are just well conditioned matrices of full rank (in virtue of Theorem 2.3).

Theorem 3.5. *(See [18, Theorem II.7].) Suppose $A \in \mathbb{R}^{n\times n}$ is a Gaussian random matrix with the mean zero and a variance σ^2. Then $F_{||A||}(y) \ge 1 - e^{-x^2/2}$ for all nonnegative $x = y/\sigma - 2\sqrt{n}$.*

Theorem 3.6. *(See [75, Theorem 3.3].) Suppose $M \in \mathbb{R}^{m\times n}$, $\bar{U} \in \mathbb{R}^{m\times m}$, and $\bar{V} \in \mathbb{R}^{n\times n}$ are three fixed matrices, $\bar{U}$ and $\bar{V}$ are unitary matrices, $A \in \mathbb{R}^{m\times n}$ is a Gaussian random matrix independent of the matrix M and having mean zero and a variance σ^2, $W = \bar{U}(A+M)\bar{V}$, l denotes $\min\{m, n\}$, and $y \ge 0$. Then $F_W(y) \le cy\sqrt{l}/\sigma$.*

Combining the two latter theorems implies the following result.

Corollary 3.4. *(See [75, Theorem 3.1].) Under the assumptions of Theorem 3.6, let $||M|| \le \sqrt{l}$. Then $F_{\mathrm{cond}\, W}(y) \ge 1 - (c_1 + c_2\sqrt{(\ln y)/n})n/(y\sigma)$ for $c_1 = 14.1$, $c_2 = 4.7\sqrt{2}$, and all $y \ge 1$.*

On a further improvement of this bound by the factor of $\sqrt{\log n}$, see [86].

Let us combine Theorem 3.6 with our results in Section 2.7 to estimate the functions $F_{GW}(y)$ and $F_{WH}(y)$ for fixed matrices G and H and Gaussian random matrix W.

Theorem 3.7. *Suppose that $\mathbb{T} \in \mathbb{C}$, $G \in \mathbb{T}^{q\times m}$, $H \in \mathbb{T}^{n\times r}$, and a random matrix $W \in \mathbb{T}^{m\times n}$ is independent of the matrices G and H and has full rank ρ with probability one. Write $r_G = \operatorname{rank} G$ and $r_H = \operatorname{rank} H$. Then $F_{GW}(y) \le F_W(y/\sigma_{r_G}(G))$ if $r_G = m$, whereas $F_{WH}(y) \le F_W(y/\sigma_{r_G}(H))$ if $r_H = n$.*

Proof. The theorem follows from Theorem 2.3. □

The theorem probabilistically bounds from below the smallest singular value of the product of a fixed matrix G or H and a random matrix W. In view of Remark 2.2, we cannot merely drop the above assumptions that $r_G \ge m$ and $r_H \ge n$, but the next theorem (employing Theorem 3.7 and employed in the next subsection) circumvents the problem. We use this theorem only for $\bar{U} = 0$ and $\bar{V} = 0$.

Theorem 3.8. *[60]. Suppose $G \in \mathbb{R}^{r_G\times m}$, $H \in \mathbb{R}^{n\times r_H}$, $X \in \mathbb{R}^{m\times n}$, $\tilde{U} \in \mathbb{R}^{r_G\times n}$, $\tilde{V} \in \mathbb{R}^{m\times r_H}$, $\operatorname{rank} G = r_G < m$, $\operatorname{rank} H = r_H < n$, and the assumptions of Theorem 3.6 hold for the matrix X replacing W. Then a) $F_{GX+\tilde{U}}(y) \le cy\sqrt{l}/(\sigma_{r_G}(G)\sigma)$ and b) $F_{XH+\tilde{V}}(y) \le cy\sqrt{l}/(\sigma_{r_H}(H)\sigma)$.*

Remark 3.3. *The estimates in [75] and consequently in our Theorem 3.8 are stated assuming real inputs, but the underlying geometric properties of random matrices (see, e.g., [76, Lemma 4.5 and Theorem 4.3]) and thus apparently the resulting probabilistic estimates in our Theorem 3.8 as well can be extended to the case of complex inputs.*

The norm estimates in Theorems 3.4 and 3.5 hold or can be readily extended to the cases of random sparse and structured matrices, but apparently the extension of the estimates of this section for the smallest positive singular values has not been elaborated upon the cases of banded, Toeplitz, and Hankel matrices (see, however, Remark 3.5 below and the Appendix), whereas random Vandermonde and Cauchy matrices are ill conditioned with some exceptions such as the matrices of discrete Fourier transforms.

Here are the estimates from [60] for the CDFs $F_{||A||}(y)$ and $F_A(y)$ for f-circulant matrices A.

Theorem 3.9. *[60]. Let an $n \times n$ circulant matrix $A = Z_1(\mathbf{v}) = \sum_{i=0}^{n-1} v_i Z_f^i$ be defined by a Gaussian random vector $\mathbf{v} = (v_i)_{i=0}^{n-1}$ having a mean μ and a variance σ^2. Then we have*

$$1 - F_{||A||}(y) \le 2n - 1 - (2n-2)\Phi_{0,\widehat{\sigma}\sqrt{n}}(y) - \Phi_{\mu n, \sigma\sqrt{n}}(y),$$

$$F_A(y) \le 2(n-1)\Phi_{0,\widehat{\sigma}\sqrt{n}}(y/\sqrt{2}) + \Phi_{\mu n, \sigma\sqrt{n}}(y)$$

for all nonnegative y, the function $\Phi_{\mu,\sigma}(y)$ defined in Section 2.2, and $2^{1/4}\sigma \le \widehat{\sigma} \le \sigma$.

Remark 3.4. *Factorizations in [10] imply that $\frac{1}{g}\sigma_j(Z_1(\mathbf{v})) \le \sigma_j(Z_f(\mathbf{v})) \le g\sigma_j(Z_1(\mathbf{v}))$ for all vectors $\mathbf{v}$, scalars f, $j = 1, 2, \dots, n$, and $g = \max\{1, |f|^2\}\ \max\{1, \frac{1}{|f|}\}$. This enables us to extend the estimates of Theorem 3.9 to f-circulant matrices for all $f \neq 0$. In particular these estimates do not change in the case of skew circulant matrices (for which $f = -1$).*

Remark 3.5. *The experiments in [60] suggest that the estimates of Theorem 3.9 are rather crude. This is probably because they rely on the bounds of Lemma 2.1, which are crude in this application. The experiments also show that as n grows large the values* cond A *for an $n \times n$ random Toeplitz matrices A tend to grow rather slowly (although much faster than in the case of circulant matrices).*

3.6 Preconditioning with random APPs

Theorem 3.3 shows that with a high probability close to one the transition $A \Longrightarrow C = A + UV^H$ fixes the rank deficiency in the case of random generators U and V of rank $r = \text{nul}\, A$ if the cardinality $|\Delta|$ is large enough. Next we recall the estimates from [54], [55] for the impact of such a transition onto cond A. For simplicity we only recall the estimates in the case of a square input.

Theorem 3.10. *Let $A = S\Sigma T^H$ be full SVD of an $n \times n$ matrix A of a rank ρ where $\rho < n$, S and T are unitary matrices, S, $T \in \mathbb{C}^{n\times n}$, $\Sigma = \text{diag}(\Sigma_A, 0_{r,r})$ is an $n \times n$ diagonal matrix, $r = n - \rho$, and $\Sigma_A = \text{diag}(\sigma_j)_{j=1}^{\rho}$ is the $\rho \times \rho$ diagonal matrix of the positive singular values of the matrix A. Suppose $U \in \mathbb{C}^{m\times r}$, $V \in \mathbb{C}^{n\times r}$, and let the $n \times n$ matrix $C = A + UV^H$ be 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 nonsingular $r \times r$ matrices. Then $R_U \Sigma R_V^H = \Sigma$, $R_U\, \text{diag}(0_{\rho,\rho}, I_r) R_V^H = S^H U V^H T$, so that $C = S R_U\, \text{diag}(\Sigma_A, I_r) R_V^H T^H$.

Corollary 3.5. *Write $\theta = \frac{||UV^H||}{||A||}$, $q = ||R_U||\ ||R_V||$ and $p = ||R_U^{-1}||\ ||R_V^{-1}||$. Suppose $\sigma_{n-r} \le 1 \le \sigma_1$. Then under the assumptions of Theorem 3.10 we have*

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

b) $\frac{1}{q} \le \frac{||C^+||}{||A^+||} \le p$, *and therefore*

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

Theorem 3.11. *Under the assumptions of Corollary 3.5, we have*

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

Proof. Combine the equations $R_U^{-1} = \begin{pmatrix} I_\rho & -U_\rho U_r^{-1} \\ 0 & U_r^{-1} \end{pmatrix}$ and $R_V^{-1} = \begin{pmatrix} I_\rho & -V_\rho V_r^{-1} \\ 0 & V_r^{-1} \end{pmatrix}$ with the bounds $\max\{||X||, ||Y||\} \le ||(X,Y)|| \le \sqrt{||X||^2+||Y||^2}$, which hold for all matrices (X,Y). □

We can readily bound the parameters θ and q from above and below by properly scaling the matrices A, U and V, e.g., $||C||/||A|| \le 2$ if $||A|| = ||UV^H||$ and $1/2 \le ||C||/||A|| \le 3/2$ if $2||A|| = ||UV^H||$.

Let us apply Theorem 3.8a to complement these estimates from [54], [55] with probabilistic upper bounds on the norms $||U_r^{-1}|| = 1/\sigma_r(U_r)$ and $||V_r^{-1}|| = 1/\sigma_r(V_r)$ and thus on the product $p = ||R_U^{-1}||\,||R_V^{-1}|| \ge \frac{||C^+||}{||A^+||}$ for a pair of random U and V.

Theorem 3.12. *Let U, V, U_r, and V_r denote the four matrices in Theorem 3.10 and let Theorem 3.8 hold (a) for $r_G = r$, $G = (0, I_r)S^H$, $X = U$, and $\tilde{U} = 0$ as well as (b) for $r_G = r$, $m = n$, $G = (0, I_r)T^H$, $X = V$, and $\tilde{V} = 0$. Then (a) $F_{U_r}(y) \le cy\sqrt{r}/\sigma$ and (b) $F_{V_r}(y) \le cy\sqrt{r}/\sigma$, respectively, for $c = 2.35$ and $F_A(y)$ in Definition 2.3.*

Proof. Apply Theorem 3.8a, at first for $r_G = r$, $G = (0, I_r)S^H$, $X = U$, and $\tilde{U} = 0$ to obtain that $F_{U_r}(y) \le cy\sqrt{r}/(\sigma_r((0, I_r)S^H)\sigma)$, and then for $r_G = r$, $m = n$, $G = (0, I_r)T^H$, $X = V$, and $\tilde{V} = 0$ to obtain that $F_{V_r}(y) \le cy\sqrt{r}/(\sigma_r((0, I_r)T^H)\sigma)$. Observe that $\sigma_r((0, I_r)S^H) = \sigma_r((0, I_r)T^H) = 1$ because $(0, I_r)S^H$ and $(0, I_r)T^H$ are unitary matrices, substitute these equations into the above bounds, and obtain the theorem. □

Now suppose A is a rank deficient matrix, $C = A + UV^H$ is its A-modification of full rank, and both matrices A and C are well conditioned. Then the A-modification with the APP UV^H transforms all ill conditioned matrices $\tilde{A} = A + E$ of full rank in a sufficiently small neighborhood of the matrix A into well conditioned matrices $\tilde{C} = C + E$ of full rank because $|\sigma_j(A+E) - \sigma_j(A)| \le ||E||$ for $j = 1, 2, \ldots, \operatorname{rank} A$ [25, Corollary 8.6.2].

In view of Corollary 3.5 and Theorems 3.11 and 3.12, we can expect that $\operatorname{cond} \tilde{C} \approx \operatorname{cond} C$ has the order of $\operatorname{cond} A = \sigma_1(A)/\sigma_{n-r}(A) \approx \sigma_1(\tilde{A})/\sigma_{n-r}(\tilde{A})$ provided U and V are Gaussian random matrices scaled so that the ratio $||UV^H||/||A||$ is neither large nor small.

Remark 3.6. *Theorem 3.12 can be readily extended to the case where the matrices U and V are fixed and well conditioned provided S^H and T^H are Gaussian random matrices, and* it is realistic to assume that the matrices S^H and T^H are random where the matrices U and V are defined independently of them. *Furthermore the estimates in the theorem for* $\operatorname{cond} A$ *were in quite good accordance with the extensive tests performed for a variety of ill conditioned matrices A in [54], even though randomness of the matrices U and V was restricted by various patterns of sparseness and structure. In spite of these restrictions, the resulting APPs UV^H regularly remained effective preconditioners. In particular this was the case for the sparse and structured generators and Hermitian APPs UU^H in Example 3.1 below (cf. [54, Example 4.6], [55, Example 6]), even where we further restricted the APPs to the primitive case $\widehat{P} = I$, $T_i = c_i I$ for all i (see below) and sampled the random parameters c_i from the sets $\{-2, -1, 1, 2\}$ or just $\{-1, 1\}$ with the subsequent scaling of the APP UU^T.*

Example 3.1. Sparse and structured Hermitian APPs. *Let k, n_1, …, n_k be positive integers (fixed or random) such that $kr + n_1 + \cdots + n_k = n$. For $i = 1, \ldots, k$, let $0_{r,n_i}$ denote the $r \times n_i$ matrices filled with zeros, and generate some $r \times r$ (fixed or random) structured or sparse well conditioned matrices T_i, 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 structured matrices with fixed patterns of sparseness, or in the simplest case just the scaled identity matrices $c_i I_r$. Define a block vector $T = (T_1, 0_{r,n_1}, \ldots, T_k, 0_{r,n_k})^T$ of block dimension $2k$ with the k nonzero block coordinates $T_1, \ldots, T_k$. Fix an $n \times n$ permutation matrix $\widehat{P}$ and define the generator $U = \widehat{P}T$ and the APP UU^H.*

Remark 3.7. *We proved Theorem 3.12 assuming that the matrices* $(0, I_r)S^H$ *and* $(0, I_r)T^H$ *are randomized by two independent random multipliers* U *and* V*, but the same proof implies the same estimates if the same random multiplier* $U = V$ *is applied to randomize both matrices, in which case our A-preprocessing keeps a Hermitian input matrix A Hermitian.*

Remark 3.8. *Assume an* $n \times n$ *matrix* A *and an* $n \times n$ *(sparse or structured) random APP* P *of a rank* $r \ll n$ *such that the ratio* $\frac{||A||}{||P||}$ *is neither large nor small. Define the matrix* $C = A + P$*. Then according to the above analysis and extensive tests in [54], the random value* $\operatorname{cond} C$ *tends to have roughly the order of* $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$*. A particular case is the matrix* $M = (A, -\mathbf{b})$ *of the linear system* $A\mathbf{y} - z\mathbf{b} = \mathbf{0}$ *in Section 1.1 provided the vector* $\mathbf{b}$ *can be viewed as random and independent of the SVD of the matrix A and the scaled ratio* $\frac{||\mathbf{b}||}{||A||}$ *is neither large nor small. Then the value cond M is expected to be of the order of* $\frac{\sigma_1(A)}{\sigma_{n-1}(A)}$*.*

4 Preprocessing via augmentation

Next we extend Theorem 3.1 to the map $A \Longrightarrow \begin{pmatrix} A & U \\ S & W \end{pmatrix}$ rather than $A \Longrightarrow A + UV^H$.

Theorem 4.1. *For three integers* m*,* n *and* r*,* $m \geq n \geq r > 0$*, and three matrices* U *of size* $m \times r$*,* S *of size* $r \times n$*, and nonsingular* W *of size* $r \times r$*, let* A *be an* $m \times n$ *matrix of a rank* ρ *and let the* $(m+r) \times (n+r)$ *matrix* $K = \begin{pmatrix} A & U \\ S & W \end{pmatrix}$ *have full rank* $n+r$*. Let* $K^{(I)}$ *be a left inverse of the matrix* K*, so that* $K^{(I)}K = I$*. Then relationships (3.1) hold, that is* $r \geq \operatorname{rank} U \geq n - \rho = \operatorname{nul} A$*, and*

$$N(A) \subseteq \operatorname{range}(I_n, 0)K^{(I)} \begin{pmatrix} U \\ 0 \end{pmatrix}. \tag{4.1}$$

Furthermore if equations (3.3) hold, that is if $r = \operatorname{rank} U = n - \rho = \operatorname{nul} A$*, then*

$$B = (I_n, 0)K^{(I)} \begin{pmatrix} U \\ 0 \end{pmatrix} \text{ is a } \operatorname{nmb}(A). \tag{4.2}$$

Proof. $\operatorname{rank} U \geq n - \rho$ because $\operatorname{rank} K = n + r \leq \operatorname{rank} A + \operatorname{rank} U + \operatorname{rank}(S, W)$, $\operatorname{rank} A = \rho$, and $\operatorname{rank}(S, W) \leq r$. This proves bound (3.1) because clearly $\operatorname{rank} U \leq r$.

Now let $\mathbf{y} \in N(A)$ and $\mathbf{z} = -W^{-1}S\mathbf{y}$. Then $K \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} A\mathbf{y} + U\mathbf{z} \\ S\mathbf{y} + W\mathbf{z} \end{pmatrix} = \begin{pmatrix} U\mathbf{z} \\ \mathbf{0} \end{pmatrix}$, and so $\mathbf{y} = (I_n, 0)K^{(I)} \begin{pmatrix} U\mathbf{z} \\ \mathbf{0} \end{pmatrix}$. This proves property (4.1).

Property (4.2) follows from (3.3) and (4.1) because $\operatorname{rank}((I_n, 0)K^{(I)} \begin{pmatrix} U \\ 0 \end{pmatrix}) \leq \operatorname{rank} U$. □

Corollary 4.1. *Let the assumptions of Theorem 4.1 hold, except possibly for equations (3.3), and let* $B = (I_n, 0)K^{(I)} \begin{pmatrix} U \\ 0 \end{pmatrix}$*. Then* BX *is a* $\operatorname{ca}(A)$ *if* X *is a* $\operatorname{ca}(AB)$*. Furthermore* X *is a* $\operatorname{ca}(AB)$ *if* BX *is a* $\operatorname{ca}(A)$ *and if* $\operatorname{rank} B = r$*.*

The proof of this result mimics the proof of Corollary 3.1, whereas Theorem 3.3 is extended as follows.

Theorem 4.2. *Assume five positive integers* m*,* n*,* q*,* r *and* ρ *such that* $\rho \leq n \leq m$ *and* $q = \min\{n, r + \rho\}$*, a set* Δ *of cardinality* $|\Delta|$ *in a ring* $\mathcal{R}$*, and five matrices* $A \in \mathcal{R}^{m \times n}$ *of rank* ρ*,* U *in* $\mathcal{R}^{m \times r}$*,* S *in* $\mathcal{R}^{r \times n}$*,* W *in* $\mathcal{R}^{r \times r}$*, and* $K = \begin{pmatrix} A & U \\ S & W \end{pmatrix}$*. Then*

a) $\operatorname{rank} K \leq q + r$*,*

b) $\operatorname{rank} K = q + r$ *with a probability of at least* $1 - \frac{2r}{|\Delta|}$ *if either the entries of the three matrices* U, S *and* W *have been randomly sampled from the set* Δ *or the entries of the matrices* U *and* W *have been randomly sampled from this set and* $S = U$,

c) $\operatorname{rank} K = q + r$ *with a probability of at least* $1 - \frac{r}{|\Delta|}$ *provided the matrix* (A, U) *(resp.* $\begin{pmatrix} A \\ S \end{pmatrix}$*) for a fixed matrix* U *(resp.* S*) has a rank of at least* q *and the entries of the matrices* S *and* W *(resp.* U *and* W*) have been randomly sampled from the set* Δ.

Proof. Part a) follows from the simple bounds $\operatorname{rank}(A, U) \le q$ and $\operatorname{rank} K \le r + \operatorname{rank}(A, U)$. Clearly both parts b) and c) hold where all the random entries are replaced with indeterminates. It remains to apply Lemma 3.2 to complete the proof. □

We omit the straightforward extensions of Remarks 3.1 and 3.2.

The following factorizations are readily verified and enable extension of a large part of our study from A-preprocessing to preprocessing by augmentation (cf. [59] on related study).

Theorem 4.3. *Assume that* $K = \begin{pmatrix} A & U \\ S & W \end{pmatrix}$ *is an* $(m + r) \times (n + r)$ *matrix,* W *is a nonsingular* $r \times r$ *matrix,* $V^H = -W^{-1}S$, $C = A + UV^H$. *Then*

$$K = \begin{pmatrix} A & U \\ -WV^H & W \end{pmatrix} = \operatorname{diag}(I_m, W)\widehat{U} \operatorname{diag}(C, I_r)\widehat{V} \tag{4.3}$$

and consequently

$$\operatorname{diag}(C, I_r) = \tilde{U} \operatorname{diag}(I_m, W^{-1})K\tilde{V}, \tag{4.4}$$

$$C = (I_m, 0)\tilde{U} \operatorname{diag}(I_m, W^{-1})K\tilde{V} \begin{pmatrix} I_n \\ 0 \end{pmatrix} \tag{4.5}$$

where $\widehat{U} = \begin{pmatrix} I_m & U \\ 0 & I_r \end{pmatrix}$, $\tilde{U} = \widehat{U}^{-1} = \begin{pmatrix} I_m & -U \\ 0 & I_r \end{pmatrix}$, $\widehat{V} = \begin{pmatrix} I_n & 0 \\ -V^H & I_r \end{pmatrix}$, *and* $\tilde{V} = \widehat{V}^{-1} = \begin{pmatrix} I_n & 0 \\ V^H & I_r \end{pmatrix}$, *and the matrix* K *has full rank if and only if the matrix* C *has full rank.*

Corollary 4.2. *Suppose the assumptions of Theorem 4.3 hold and the matrices* C *and* K *have full rank. Then*

$$K^+ = \tilde{V} \operatorname{diag}(C^+, I_r)\tilde{U} \operatorname{diag}(I_m, W^{-1}), \tag{4.6}$$

$$\operatorname{diag}(C^+, I_r) = \widehat{V}K^+ \operatorname{diag}(I_m, W)\widehat{V}, \tag{4.7}$$

$$C^+ = (I_m, 0)\widehat{V}K^+ \operatorname{diag}(I_m, W)\widehat{V} \begin{pmatrix} I_n \\ 0 \end{pmatrix}. \tag{4.8}$$

Under the assumptions of Corollary 4.2, we can express the matrix A^+ via the matrix K^+ by combining equations (2.4) and (4.8).

Corollary 4.3. *Under the assumptions of Corollary 4.2 let* $||C|| = ||W|| = 1$ *and* $||U|| = ||V|| \le \gamma$ *for a scalar* γ. *Then we have*

a) $\operatorname{rank} K = r + \operatorname{rank} C$,

b) $\operatorname{cond} K \le (1 + \gamma)^4 (\operatorname{cond} W) \operatorname{cond} C$, *and*

c) $\operatorname{cond} C \le (1 + \gamma)^4 (\operatorname{cond} W) \operatorname{cond} K$.

Note that $(1 + \gamma)^4 = 16$ for $\gamma = 1$, $(1 + \gamma)^4 < 4$ for $\gamma = 0.4$.

Proof. Part a) immediately follow from equation (4.3).

b) We have $\operatorname{cond}(\operatorname{diag}(C, I_r)) = \operatorname{cond} C$ and $\operatorname{cond}(\operatorname{diag}(I_n, W)) = \operatorname{cond} W$ because $||C|| = ||W|| = 1$. Furthermore we have $||\widehat{U}|| = ||\widehat{U}^{-1}|| \le 1 + \gamma$, $||\widehat{V}|| = ||\widehat{V}^{-1}|| \le 1 + \gamma$. Therefore $\operatorname{cond} \widehat{U} \le (1 + \gamma)^2$, $\operatorname{cond} \widehat{V} \le (1 + \gamma)^2$. Now part b) follows from equations (4.3) and Theorem 2.3.

Part c) follows from equation (4.4) because $\operatorname{cond}(\operatorname{diag}(C, I_r)) = \operatorname{cond} C$. □

By combining Theorem 3.12 and Corollary 4.3, we deduce that for Gaussian random and properly scaled matrices S, U, and W, the value $\mathrm{cond}\, K$ is expected to be of the order of $\mathrm{cond}\, A = \sigma_1(K)/\sigma_{n-r}(A)$.

It is instructive to reverse the augmentation where we seek a $\mathrm{nmb}(K)$ for an $(n+r)\times(n+r)$ matrix $K = \begin{pmatrix} A & U \\ S & W \end{pmatrix}$ of rank n having nonsingular diagonal block A of the size $n \times n$. Then the matrix $B = \begin{pmatrix} A^{-1}U \\ -I_r \end{pmatrix}$ is a $\mathrm{nmb}(K)$. Indeed, $(A, U)B = 0$ and all rows in the lower block (S, W) of the matrix K are linear combinations of the rows of the upper block (A, U).

This gives us an approach to computing a $\mathrm{nmb}(K)$, which, however, fails (resp. is prone to numerical problems) where the matrix A is rank deficient (resp. ill conditioned). Unlike preprocessing by augmentation, in this case we have no random parameters involved with which we could have fixed such a deficiency (cf. [60] on some remedies).

Remark 4.1. *Unlike A-preprocessing, augmentation is flop-free and allows us to completely preserves the input matrix structure. Furthermore augmentation is always performed error-free, but with reasonable effort we can define error-free A-preprocessing as well. E.g., we can fill the random (or random sparse and structured) matrices U or V with shorter numbers. Furthermore 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 (cf. Example 3.1). The power of A-preconditioning is still preserved for such APPs according to the analysis and experiments in [54] and [55].*

5 Perturbation and error estimates

In this subsection, for the sake of completeness of our presentation in Section 6.6, we recall some basic estimates for the errors and perturbations in matrix computations. We assume computations in the field of real numbers (cf. [37] and [85, page 447] on the extension to computations in the complex field). We cover estimates for the general rectangular input, but in view of Section 3.2 the reader may omit a large part of this material and rely just on the simpler estimates for the square inputs.

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 employ the matrix norms $||\cdot||_l$ for $l = 1, 2, \infty, F$ and recall that $||A||_l = ||\,|A|\,||_l$ for $l = 1, \infty, F$ and that $||A|| \leq ||B||$ if $|A| \leq B$ (cf. [73, page 53]).

Theorem 5.1. *(Cf. [27, 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-by-matrix and matrix-by-vector 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 5.2. *(Cf. [27, 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 nonsingular 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 $\epsilon\, \mathrm{cond}_l\, A < 1$. Then $\frac{||(\tilde{\mathbf{x}}-\mathbf{x})||_l}{||\tilde{\mathbf{x}}||_l} \leq 2\epsilon \frac{\mathrm{cond}_l\, A}{1-\epsilon\, \mathrm{cond}_l\, A}$.*

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

Theorem 5.3. *(Cf. [27, Theorem 8.5], [73, 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 $\frac{||\tilde{\mathbf{x}}-\mathbf{x}||_l}{||\tilde{\mathbf{x}}||_l} \leq 1.12nu\, \mathrm{cond}_l\, T$ for $l = 1, \infty, F$.*

Theorem 5.4. *(Cf. [27, Theorems 9.3 and 9.4], [73, 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 $\tilde{L}\tilde{U} = A + \Delta(A)$, $|\Delta(A)| \le \gamma_n |\tilde{L}|\ |\tilde{U}|$.

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 $(A + \Delta(A))\tilde{\mathbf{x}} = \mathbf{v}$, $|\Delta(A)| \le 3\gamma_n |\tilde{L}|\ |\tilde{U}|$. Furthermore $\frac{||A(\tilde{\mathbf{x}}-\mathbf{x})||_l}{||\tilde{\mathbf{x}}||_l} \le 1.12(3 + 1.12nu)nu||L||_l||U||_l$ for $l = 1, \infty, F$.

For Gaussian elimination with rook and complete pivoting (which use from about $2n^2$ to $\frac{1}{3}n^3$ comparisons) the factor $g(A) = \frac{||L||_l||U||_l}{||A||_l}$ does not grow very rapidly with n. Even with partial pivoting (which uses $(n-1)n/2$ comparisons) we always have $||L||_l \le 1$, whereas according to empirical evidence the norm $||U||_l$ usually has order 10 [25, page 116].

Theorem 5.5. *(Cf. [27, 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 5.6. *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. [27, 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. [27, 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. [31, Chapter 16]).

Theorem 5.7. *(Cf. [27, 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 5.8. *(Cf. [27, 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

$$||\tilde{\mathbf{x}} - \mathbf{x}|| = ||\tilde{\mathbf{x}} - A^+\mathbf{v}|| \le ||A^+||\ ||A\tilde{\mathbf{x}} - \mathbf{v}||. \tag{5.1}$$

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. [2], [27, 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

$$c_{m,n}(\operatorname{cond} A)^2 u||\mathbf{x}|| \tag{5.2}$$

for a scalar $c_{m,n}$ depending on m and n [27, 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. [1]), respectively.

We conclude with the following perturbation estimates by Wedin 1973 (cf. [27, Theorem 20.1]).

Theorem 5.9. *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* $\frac{||\mathbf{x}-\mathbf{y}||}{||\mathbf{x}||} \le \frac{\epsilon \operatorname{cond} A}{1-\epsilon \operatorname{cond} A}(2 + (\operatorname{cond} A + 1)\frac{||\mathbf{v}-A\mathbf{x}||}{||A||\ ||\mathbf{x}||})$.

6 Null bases via A-preprocessing

In this section we compute nmbs based on randomized A-preprocessing, Theorems 3.1 and 4.2 and Corollary 3.1. We specificy our algorithms by employing the left inverses $C^{(I)}$ and $K^{(I)}$ given by the Moore–Penrose generalized inverses C^+ and K^+, respectively, but any other choice of the left inverses can be used instead.

6.1 The auxiliary least squares computations, symmetrization and matrix structure

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. We can reduce our problem to the case where $m = n$, $C^+ = C^{-1}$ (see Section 3.3), and then we would just compute the matrix $C^{-1}U$. Alternatively we can choose among various effective numerical methods for the least squares task [3], [25, Section 5.3], [27, Chapter 20], [31], [73, Section 4.2]. Even though $\operatorname{cond}(C^HC) = \operatorname{cond}(CC^H) = (\operatorname{cond} C)^2$, the symmetrizations $C \Longrightarrow C^HC$ and $C \Longrightarrow CC^H$ (leading to normal and corrected seminormal equations) are still 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" [25, 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" [73, page 298].)

The structure of a matrix C may deteriorate a little, but is not lost in the symmetrization, which at most doubles the displacement rank $\operatorname{dr}_{A,B}(M)$ and makes similar impact on the bandwidth and the rank of a matrix (cf. the definitions in Section 2.2).

Theorem 6.1. *(Cf. [48, Section 4.7].) We have* $\operatorname{dr}_{A,A}(C^HC) \le 2\operatorname{dr}_{A,B}(C)$ *and* $\operatorname{dr}_{B,B}(CC^H) \le 2\operatorname{dr}_{A,B}(C)$.

Theorem 6.2. *(Cf. [20, Theorem 4.1].) Let a matrix* C *have a lower bandwidth (resp. lower rank)* $l = l_C$ *and an upper bandwidth (resp. upper rank)* $u = u_C$. *Then* $l_G \le l + u$ *and* $u_G \le l + u$ *for* $G = C^HC$ *and* $G = CC^H$.

Proof. Clearly, $l_A = u_{A^H} = u_{A^T}$ for any matrix A, in particular $l_{C^H} = u_C$ and $l_C = u_{C^H}$. Now the theorem follows because $l_{AB} \le l_A + l_B$ and $u_{AB} \le u_A + u_B$ for any matrix product AB. (Transposition of the matrices A, B, and AB shows equivalence of the two latter inequalities.) For banded matrices the inequalities follow from the inclusion $\mathbb{D}_s\mathbb{D}_t \subseteq \mathbb{D}_{s+t}$. Here $\mathbb{D}_q$ is the class of matrices whose nonzero entries can appear only on their qth subdiagonal for $q \ge 0$ and only on their $(-q)$th superdiagonal for $q \le 0$. Lemma 6.1 completes the proof of the theorem. □

Lemma 6.1. *$l_{AB} \le l_A + l_B$ for a pair of rank structured matrices A and B of compatible sizes.*

Proof. We assume dealing with $n \times n$ matrices A and B. (Rectangular matrices can be embedded into square matrices banded with zeros.) Represent a subdiagonal block $V_h = (v_{ij})_{i=n-h,j=0}^{n-1,n-h-1}$ of the matrix $V = AB$ as $A_h B_h + \widehat{A}_h \widehat{B}_h$ where

$$A_h = (a_{ij})_{i=n-h,j=0}^{n-1,n-h-1},\ B_h = (b_{jk})_{j=0,k=0}^{n-h-1,n-h-1},$$

$$\widehat{A}_h = (a_{ij})_{i=n-h,j=n-h}^{n-1,n-1},\ \widehat{B}_h = (b_{jk})_{j=n-h,k=0}^{n-1,n-h-1}.$$

Observe that A_h and $\widehat{B}_h$ are subdiagonal blocks of the matrices A and B, respectively. Therefore $\operatorname{rank} A_h \le l_A$ and $\operatorname{rank} \widehat{B}_h \le l_B$. Consequently $\operatorname{rank}(A_h B_h) \le \operatorname{rank} A_h \le l_A$, $\operatorname{rank}(\widehat{A}_h \widehat{B}_h) \le \operatorname{rank} \widehat{B}_h \le l_B$, and $\operatorname{rank} V_h = \operatorname{rank}(A_h B_h + \widehat{A}_h \widehat{B}_h) \le \operatorname{rank}(A_h B_h) + \operatorname{rank}(\widehat{A}_h \widehat{B}_h) \le l_A + l_B$. □

6.2 Error-free computation of nmbs

In this subsection we cover symbolic error-free computation of nmbs in a fixed field $\mathbb{F}$, e.g., the field of rational, real or complex numbers. We begin with a simpler case where we are given the nullity of an input matrix.

Algorithm 6.1. Computing a 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 $m \ge n > r \ge 0$.*

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

INITIALIZATION: *Set $k \Longleftarrow 1$. 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 $(2/|\Delta|)^\nu \le \delta$.*

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 $C = A + UV^H$. If this matrix is rank deficient, then either output FAILURE and stop if $k \ge \nu$ or otherwise set $k \Longleftarrow k + 1$ and go to Stage 1.*
3. *Compute the matrix C^+U. Compute and output a matrix basis for its range and stop.*

Remark 6.1. *The computation of the matrix V at Stage 1 can be omitted, but if $m \le n$ and the matrices C, U and V have full ranks, we can compute the matrix $V^H C^+$ and test whether it is a left nmb(A), that is whether $V^H C^+ A = 0$. If $m = n$, we can recall Corollary 3.2 and modify Algorithm 6.1 and its latter extension to compute both left and right nmbs by using the matrix $G = I_r - V^H C^+ U$ instead of the matrices AC^+U and $V^H C^+ A$. This remark can be applied to our subsequent algorithms as well.*

Unless it fails, the algorithm produces a 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 3.3b for $r = 1$, that is the overall probability of failure is at most $(2/|\Delta|)^\nu \le \delta$. This proves correctness of the algorithm.

In each invocation of Stage 1, it generates $(m + n)r$ random parameters in the case of general APP UV^H, but *as in all other our algorithms, we need much fewer parameters for a sparse or structured APP* (cf. Example 3.1).

If the nullity $\operatorname{nul} A$ is unknown, we can recall Theorems 3.1 and 3.3 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,

i') 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,

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

ii') the rank of an APP UV^H such that the matrix $C = A + UV^H$ has full rank and the matrix $AC^+U\mathbf{x}$ vanishes with a probability near one for a random vector $\mathbf{x}$.

Next, assuming some initial range $[r_-, r_+]$ for the nullity $r = \operatorname{nul} A$ such that $0 \le r_- \le r \le r_+ \le n-1$, we generate 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 search for the nullity r in this range. We specify two algorithms that perform linear (sequential) search based on properties i) and ii) above, where we successively choose $i = r_-, r_- + 1, \dots, r$ or $i = r_+, r_+ - 1, \dots, r$, respectively (see Remark 6.2 and Algorithm 6.4 on the acceleration of the search). One can modify the algorithms by adding randomization and relying on properties i') and ii'), respectively. The algorithms employ a black box Subroutine FULL·RANK that tests whether a given matrix has full rank (cf. Remark 6.5).

Algorithm 6.2. A nmb via the nullity search from below.

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

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

INITIALIZATION: *Set $q \Longleftarrow r_-$. Fix a sufficiently large set Δ of real or complex numbers such that $2\frac{r_+ + 1}{|\Delta|} \le \delta$. Sample from this set the entries of 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 $C \Longleftarrow C + \mathbf{u}\mathbf{v}^H$, $U \Longleftarrow (U, \mathbf{u})$, $V \Longleftarrow (V, \mathbf{v})$ (cf. Remark 6.1), and $q \Longleftarrow q + 1$, and go to Stage 1.*
3. *If the matrix C has full rank, compute the matrices C^+U and AC^+U. If $AC^+U \ne 0$, output FAILURE. Otherwise compute and output an $n \times r$ matrix basis B for* $\operatorname{range}(C^+U)$, *where $r \le q$. (B is the $n \times 0$ empty matrix if $r = 0$.) Output the integer r and stop.*

In the case of general APP UV^H the algorithm generates $(m+n)r_-$ random parameters at the Initialization Stage and then $m + n$ new random parameters in each invocation of Stage 2.

$\operatorname{rank} C = \operatorname{rank} A + r_-$ at the initialization stage with a probability of at least $1 - 2\frac{r_-}{|\Delta|}$, due to Theorem 3.3b for $r = r_-$, whereas in every recomputation of the matrix C (at Stage 2) its rank increases with a probability of at least $1 - \frac{2}{|\Delta|}$, due to Theorem 3.3b for $r = 1$. This means a probability of at least $(1 - \frac{2}{|\Delta|})^{r_+ - r_- + 1}(1 - 2\frac{r_-}{|\Delta|}) > 1 - 2\frac{r_+ + 1}{|\Delta|} \ge 1 - \delta$ 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 the 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).

Algorithm 6.2 tests property i) of the nullity $\operatorname{nul} A$ where $C = A + UV^H$ and the APPs UV^H have rank recursively increasing from r_-. Our next algorithm tests property ii) for the matrices $C = A + UV^H$ where the APPs UV^H have ranks recursively decreasing from r_+.

Algorithm 6.3. A nmb via the nullity search from above.

INPUT *and* OUTPUT *as in Algorithm 6.2.*

INITIALIZATION: *Set $q \Longleftarrow r_+$. 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 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(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. Set $C \Longleftarrow C - \mathbf{u}\mathbf{v}^H$, $q \Longleftarrow q - 1$, and go to Stage 1.*

In the case of a general APP UV^H the algorithm generates $(m+n)r_+$ random parameters at the Initialization Stage and then generates $m + n$ new random parameters in each invocation of Stage 2.

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 3.3b 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 3.3b 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 claim). The algorithm outputs FAILURE only if it encounters a rank deficient matrix C, but according to the above estimates this occurs with a low probability where the cardinality $|\Delta|$ is large, and similarly for our next algorithm.

Remark 6.2. *Both Algorithms 6.2 and 6.3 compute the nullity by means of the linear (sequential) search in the range $[r_-, r_+]$ based on properties i) and ii) 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 matrix C^+ (by applying SMW formula (2.4)). Our next algorithm, based on Corollary 3.1, demonstrates yet another acceleration technique: it applies aggregation to compute the nullity and a nmb(A).*

Algorithm 6.4. A nmb via aggregation *(see Section 6.3).*

INPUT, OUTPUT, INITIALIZATION *and Stages 1 and 2 of* COMPUTATIONS *are as in Algorithm 6.3, except that at the* INITIALIZATION *we require that $8\frac{r_+}{|\Delta|} \leq \delta$.*

COMPUTATIONS:

3. Otherwise apply the algorithm to the $m \times q$ matrix AC^+U. (The algorithm can fail with a probability of at most $2\frac{q}{|\Delta|} \leq 2\frac{r_+}{|\Delta|}$.) Unless it fails, it computes an integer $s \leq q$, a $q \times s$ matrix $X = \mathrm{nmb}(AC^+U)$, and the matrix $H = C^+UX$, which is a $\mathrm{ca}(A)$. In view of Lemma 3.2, we can expect that $s = \mathrm{nul}\, A$ and H is a $\mathrm{nmb}(A)$. To yield a verified nmb, apply the algorithm supporting Fact 2.1 to the matrix $H = C^+UX$ and compute an $s \times r$ matrix Y such that HY is a $\mathrm{nmb}(A)$. Then output the matrix HY and the integer r and stop.

Correctness of the algorithm follows from Fact 2.1, Corollary 3.1 and Lemma 3.2. Indeed in virtue of Corollary 3.1, $H = C^+UX$ is a $\mathrm{ca}(A)$. Lemma 3.2 implies that C^+U is a matrix of full rank with a probability of at least $1 - \frac{q}{|\Delta|}$. If it has full rank, then so does the matrix C^+UX, because X is a nmb. In this case $H = C^+UX$ is a $\mathrm{nmb}(A)$. Otherwise HY is a $\mathrm{nmb}(A)$ in virtue of Fact 2.1.

Remark 6.3. *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 $G = I_r - V^HC^{-1}U$ by using $2(2r-1)nr + r$ flops.*

Remark 6.4. *To compute a single null vector rather than a nmb, we can fix a vector* $\mathbf{c}$ *and compute the vector* $C^+U\mathbf{c}$ *instead of the matrix* C^+U *in Algorithms 6.1–6.4, and similarly we can simplify Stage 3 of Algorithm 6.4. We can reapply the same algorithm to the matrix* $\begin{pmatrix} A \\ \mathbf{y}^H \end{pmatrix}$ *to obtain another null vector. Indeed for a fixed null vector* $\mathbf{y} \in N(A)$*, its orthogonal complement in the null space* $N(A)$ *is given by the null space of the matrix* $\begin{pmatrix} A \\ \mathbf{y}^H \end{pmatrix}$*. We can continue this process recursively until the computed null vectors form a* $\mathrm{nmb}(A)$*.*

Remark 6.5. *The Subroutine FULL·RANK is used in Algorithms 6.3 and 6.4 only as a stopping criterion at Stages 1 or 2. Instead of applying it, we can implicitly test whether the matrix* C *is rank deficient by trying to compute the matrix* C^+U*. If the computation fails, then the matrix* C *is definitely rank deficient. Otherwise it has full rank with a probability near one.*

6.3 Null Aggregation

Unless $AC^+U = 0$ at Stage 2, Algorithm 6.4 is a new instance in the general class of *aggregation/disaggregation methods* that 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.

In applications to matrix computations, one can seek a matrix W that reduces a linear system $A\mathbf{y} = \mathbf{b}$ to a system $B\mathbf{y}_1 = \mathbf{f}$ of a smaller size such that $\mathbf{y} = W\mathbf{y}_1$. In this case $\mathbb{I} = A$, $\mathbb{I}_1 = W$, $Y_1 = \mathbf{y}_1$, and $Y = \mathbf{y}$. Recursively this has led to the hierarchial aggregation processes in [36], which in the 1980s served as the springboard for *Algebraic Multigrid*.

For another example, SMW formula (2.4) defines the *Schur aggregation* in [52], where $\mathbb{I} = A$, $\mathbb{I}_1 = G$, $Y_1 = G^{-1}$, and $Y = A^{-1}$, with the aggregate G being the Schur complement in the input matrix A.

Algorithm 6.4 defines *Null Aggregation*, where $\mathbb{I} = A$, $\mathbb{I}_1 = AC^+U$, $Y_1 = X$, and $Y = C^+UX$. If $m = n$, one can alternatively choose $\mathbb{I}_1 = G = I_r - V^HC^+U$ (cf. Remark 6.1). Fact 2.1b) defines aggregation of a matrix A into its complete annihilator H, in which case $\mathbb{I} = H$, $\mathbb{I}_1 = X^H$.

Trilinear aggregating in [42] and [43] has supported the design of the fastest known theoretical and practical algorithms for matrix multiplication in [11], [28], and [32]. The method works by first reducing matrix multiplication to *tensor decomposition* and then compressing the associated tensors by means of special aggregation techniques.

6.4 Numerical computation of nmbs: initial comments

Suppose our algorithms have been performed numerically, with rounding errors. Let $B + E$ denote the output matrix where $B = \mathrm{nmb}(A)$ and E denotes the error matrix. Then generically $A(B + E) = AE \neq 0$ and furthermore the computed matrix $\tilde{C} = \mathrm{fl}(A + UV^H)$ has full rank even where $\mathrm{rank}\, U = \mathrm{rank}\, V < \mathrm{nul}\, A$.

In the next subsection we modify Algorithms 6.1–6.4 to accomodate these changes. Instead of testing whether the matrix C is rank deficient and whether $AC^+U = 0$, we apply two Subroutines ILL·CONDITIONED and NORM. For two fixed tolerance values τ and t, they test whether

$$\tau\, \mathrm{cond}\, C > 1 \tag{6.1}$$

(which means that a rank deficient matrix approximates the matrix C within the norm bound $\tau||C||$) and whether the residual norm $||AB||$ is small enough, namely whether

$$||AB|| \leq t||A||\ ||B||. \tag{6.2}$$

In virtue of Theorem 3.3 randomized A-preprocessing $A \Longrightarrow C = A + UV^H$ fixes degeneracy with a probability near one if $\mathrm{rank}(UV^H) \geq \mathrm{nul}\, A$. In virtue of Corollary 3.5 and Theorems 3.11 and 3.12, this preprocessing, applied to the well conditioned rank deficient matrix A, is expected to

produce a well conditioned A-modifiication C (of full rank) under proper scaling of the APP, and then the matrices $C + E$ must be also well conditioned if the ratio $||E||/||C||$ is small.

We further comment on the two subroutines and the tolerance bounds in Section 6.6.

6.5 Numerical computation of nmbs: algorithms

Let us specify numerical versions of Algorithms 6.1–6.4 for general matrices A and C (cf. Section 9 on the case of structured inputs A). Remarks 6.1–6.5 can still be readily extended.

Algorithm 6.5. Computing a numerical nmb given the nullity.

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 or an $n \times r$ matrix B such that bound (6.2) holds.*

INITIALIZATION: *Set $k \Longleftarrow 1$. Fix a small positive integer ν (say, $\nu = 1$ or $\nu = 2$) and a sufficiently large set Δ of real or complex numbers.*

COMPUTATIONS:

1. *Generate two 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 $U \Longleftarrow \frac{||A||}{||U||}U$, $V \Longleftarrow \frac{V}{||V||}$, and $C \Longleftarrow A + UV^H$.*
2. *Compute the matrices $B = C^+U$ and AB.*
3. *Apply the Subroutine NORM to the matrix AB. If bound (6.2) holds, output the matrix B and stop. Otherwise either output FAILURE and stop if $k \geq \nu$ or set $k \Longleftarrow k + 1$ and go to Stage 1 if $k < \nu$.*

Unless it fails, the algorithm verifies correctness of its output at Stage 3. The failure can occur for two reasons: a) because of an unlikely 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. The same comments apply to our next algorithms as well.

In Algorithm 6.5 we assume that we are given the nullity $r = \operatorname{nul} A$. In our next numerical counterparts of Algorithms 6.2–6.4 we compute the nullity. In Algorithm 6.6 we expect to invoke at least $r - r_-$ ill conditioned matrices C of full rank that satisfy bound (6.1), whereas Algorithm 6.7 involves such matrices with a probability near zero. Then again (cf. Remark 6.2), we can update the matrix C and its inverse in Algorithms 6.6 and 6.7 at a lower arithmetic cost based on SMW formula (2.4).

Algorithm 6.6. 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 $n \times r$ matrix B such that bound (6.2) holds.*

INITIALIZATION: *Set $q \Longleftarrow r_-$. Fix a sufficiently large set Δ of real or complex numbers. Sample from this set the entries of random matrices U of size $m \times q$ and V of size $n \times q$, scale these matrices to have $||U| \approx ||A||$ and $||V|| \approx 1$, and compute the matrix $C = A+UV^H$. (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* cond $C > \frac{1}{\tau}$, *then either output FAILURE and stop if* $q \geq r_+$ *or, otherwise, sample from the set* Δ *the entries of two random column vectors* $\mathbf{u} = (u_i)_{i=1}^m$ *and* $\mathbf{v} = (v_i)_{i=1}^n$, *scale them and append to the matrices* U *and* V *as follows,* $\mathbf{u} \Longleftarrow \frac{\mathbf{u}}{\gamma||\mathbf{u}||}$ *for* $\gamma = \sqrt{q}$, $\mathbf{v} \Longleftarrow \frac{\mathbf{v}}{||\mathbf{v}||}$, $U \Longleftarrow (U, \mathbf{u})$ *and* $V \Longleftarrow (V, \mathbf{v})$. *Set* $C \Longrightarrow C + \mathbf{u}\mathbf{v}^H$ *and* $q \Longleftarrow q + 1$, *and go to Stage 1.*

3. *If* cond $C \leq \frac{1}{\tau}$, *then compute the* $n \times q$ *matrix* $B = C^+U$ *and set* $r = q$.

4. *Compute the matrix* AB *and apply the Subroutine NORM to this matrix. If bound (6.2) holds, output the integer* r *and the matrix* B *and stop. Otherwise output FAILURE and stop.*

Algorithm 6.7. A numerical nmb via the nullity search from above.

INPUT *and* OUTPUT *are as in Algorithm 6.6.*

INITIALIZATION: *Set* $q \Longleftarrow r_+$, *fix a sufficiently large set* Δ *of numbers, sample from this set the entries of random matrices* U *of size* $m \times q$ *and* V *of size* $n \times q$, *scale these matrices to have* $||U| \approx ||A||$ *and* $||V|| \approx 1$, *and compute the matrix* $C = A + UV^H$.

COMPUTATIONS:

1. *Apply the Subroutine ILL·CONDITIONED to the matrix* C.

2. *If* cond $C > \frac{1}{\tau}$, *then output FAILURE and stop. Otherwise compute the* $n \times q$ *matrix* $B = C^+U$.

3. *Compute the matrix* AB *and apply the Subroutine NORM to this matrix. If bound (6.2) holds, output the integer* $r = q$ *and the matrix* B *and stop.*

4. *Otherwise output FAILURE and stop if* $q = r_-$. *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* $C \Longleftarrow C - \mathbf{u}\mathbf{v}^H$, *set* $q \Longleftarrow q - 1$, *and go to Stage 1.*

Algorithm 6.8. A numerical nmb via aggregation. *(Cf. Section 6.3)*

INPUT, OUTPUT, INITIALIZATION, *and Stages 1, 2 and 3 of* COMPUTATIONS *are as in Algorithm 6.7, except that the input includes a small positive integer* ν *and that an additional parameter COUNTER is initialized at zero.*

COMPUTATIONS:

4. Otherwise stop and output FAILURE if COUNTER exceeds ν. *Otherwise set COUNTER* $\Longleftarrow 1+COUNTER$ *and apply the same algorithm to the* $m \times q$ *matrix* AB *but with another tolerance bound. Namely, unless the algorithm outputs FAILURE, require that it output a* $q \times r$ *matrix* X *such that* $r_- \leq r \leq q$ *and* $||ABX|| \leq t||A||\,||BX||$. *Then set* $B \Longleftarrow BX$, *output the integer* r *and the* $n \times r$ *matrix* B *and stop.*

Remark 6.6. *According to our study in Sections 3.5 and 3.6, the random matrices* U, V, *and* B *in Algorithms 6.5–6.8 are expected to be well conditioned, but to yield additional numerical stabilization one can orthogonalize them, which takes low cost if an upper bound* r_+ *on the numerical nullity* nnul A *is small.*

6.6 Numerical computation of nmbs: the error and tolerance bounds

Bound (6.1) holds if and only if our A-modification does not decrease the value cond C to the desired level. Bound (6.2) shows us that the matrix B approximates a nmb(A) within a fixed tolerance to the residual norm. By employing these two bounds we can extend rules i), i'), ii), and ii') in Section 6.2 to numerical computations.

To test bound (6.1) we can apply the effective condition estimators in [12], [25, Section 3.5.4], [27, Chapter 15], and [73, Section 5.3] or extend our comments in Remark 6.5 respectively.

Let us link bounds (6.1) and (6.2) to the respective error estimates based on the results in Section 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 $||A|| = ||UV^H|| = 1$ and therefore $||C|| \le 2$. Further assume that $\Delta(C) = 0$, thus ignoring the smaller errors in computing the matrix C (cf. Remark 4.1). 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 5 for the errors in computing matrix products and the solutions and least squares solutions to linear systems of equations, we obtain that $||\Delta(C^+)|| \le c_{m,n}u\kappa_-$ and $||\Delta(AC^+U|| \le c_{m,n}u||A||\kappa_-$.

We can decrease the value u and therefore the output residual norm bounds if we increase the precision of computing. Alternatively we can stay with the double precision computations but apply fast advanced algorithms in [16], [27], [30], [38], [57], and [71] (which rapidly compute sums and products error-free or with high accuracy) and the extended iterative refinement in [52, Section 9] for the solution of the auxiliary well conditioned linear systems of equations.

Remark 6.7. *The error estimates are more favorable in the case of square (nonsingular) matrices C, and this can motivate using the respective techniques in Section 3.3.*

6.7 Computation of nmbs via augmentation

We can readily replace A-preprocessing in Algorithms 6.1–6.8 with preprocessing by augmentation, but we only specify such a modification for Algorithm 6.1.

Algorithm 6.9. Computing a nmb given the nullity via augmentation *(cf. (4.2)).*

INPUT, OUTPUT, *and* INITIALIZATION *as in Algorithm 6.1.*

COMPUTATIONS:

1. *Randomly sample from the set Δ the entries of three matrices, U of the size $m \times r$, S of the size $n \times r$, and W of the size $r \times r$.*
2. *Compute the matrix $K = \begin{pmatrix} A & U \\ S & W \end{pmatrix}$. If this matrix is rank deficient, then either output FAILURE and stop if $k \ge \nu$ or otherwise set $k \Longleftarrow k + 1$ and go to Stage 1.*
3. *Compute the matrix $(I_n, 0)K^+ \begin{pmatrix} U \\ 0 \end{pmatrix}$. Compute and output a matrix basis for its range and stop.*

7 Approximating nmbs and the tails and heads of the SVD. Approximation by lower rank matrices and by structured matrices and A-preconditioning

7.1 Approximation of nmbs and the tails and heads of the SVD

Suppose $\tilde{A} = A + E$ is a perturbation of an $n \times n$ matrix A of a rank ρ and the ratio $||E||/\sigma_\rho(A)$ is small. Let $A = S\Sigma T^H$ and $\tilde{A} = \tilde{S}\tilde{\Sigma}\tilde{T}^H$ be the SVDs. Write $T = (\mathbf{t}_j)_{j=1}^n$, $T_r = (\mathbf{t}_j)_{j=n-r+1}^n$, $T^{(n-r)} = (\mathbf{t}_j)_{j=1}^{n-r}$, $\tilde{T} = (\tilde{\mathbf{t}}_j)_{j=1}^n$, $\tilde{T}_r = (\tilde{\mathbf{t}}_j)_{j=n-r+1}^n$, $\tilde{T}^{(n-r)} = (\tilde{\mathbf{t}}_j)_{j=1}^{n-r}$. Now if $r = \mathrm{nnul}\,\tilde{A}$, then the linear space $\tilde{\mathbb{T}}_{n-r,r} = \mathrm{range}\,\tilde{T}_r$ closely approximates the null space $N(A)$ [25, Theorem 8.6.5], [74, Section 3.3.1], and we can extend some of our earlier study to A-preprocessing $\tilde{A} \Longrightarrow \tilde{C} = \tilde{A} + UV^H$.

E.g., for $\tilde{r} \le r$, $\operatorname{nnul} \tilde{C}$ is expected to equal $r - \tilde{r}$ assuming random and properly scaled matrices $U \in \mathbb{C}^{m \times \tilde{r}}$ and $V \in \mathbb{C}^{n \times \tilde{r}}$. We can obtain similar extensions of Corollary 3.5 and Theorem 3.12. Furthermore, based on Theorem 4.3 or directly (cf. [59]) we can extend these results to preprocessing by augmentation $\tilde{A} \Longrightarrow \tilde{K} = \begin{pmatrix} \tilde{A} & U \\ S & W \end{pmatrix}$ for random and properly scaled matrices U, S, and W.

Further assume that A-modification $C = A + UV^H$ has full rank n and is well conditioned, $\operatorname{range}(C^+U) = N(A)$, and the perturbation $\tilde{C}^+ - C^+$ has a small norm (see Section 7.4 on the respective estimates in terms of the norm $||E||$). Then clearly $\tilde{R} = \operatorname{range}(\tilde{C}^+U) \approx N(A) \approx \tilde{\mathbb{T}}_{n-r,r} = \operatorname{range} \tilde{T}_r$ (the r-tail of the matrix $\tilde{A}$), whereas the orthogonal complement of the linear space $\tilde{R}$ approximates the linear space $\tilde{\mathbb{T}}_{0,q} = \operatorname{range} \tilde{T}^{(q)}$ (the q-head of the matrix $\tilde{A}$) for $q = n - r$.

We can directly approximate the linear space $\tilde{\mathbb{T}}_{0,q}$ by $\operatorname{range}((\tilde{C}_-)^{+H}U)$ where $(\tilde{C}_-)^{+H} = (\tilde{A}^{+H} + UV^H)^{+H}$, $U \in \mathbb{C}^{m \times q}$, $V \in \mathbb{C}^{n \times q}$, $m \ge n$, and $q = \operatorname{nnul}(A^{+H})$ (cf. (2.6)). This follows because the q-head $\tilde{\mathbb{T}}_{0,q}$ of an $m \times n$ matrix $\tilde{A}$ of full rank coincides with the q-tail of the matrix $\tilde{A}^{+H}$ for $m \ge n \ge q$. One may prefer dealing with $n \times q$ rather than $n \times r$ matrices U and V where $q \ll r$ or may prefer applying the dual (rather than standard) SMW formulae (see Section 11.4).

Remark 7.1. *If $r = \operatorname{nnul} \tilde{A} > 0$, we can rapidly approximate the null space $N(A)$ and the r-tail $\tilde{\mathbb{T}}_{n-r,r}$ by choosing zero as the initial approximate eigenvalue and applying the Rayleigh quotient iteration, inverse iteration [25], [74], or their modifications in [61], [68], based on A-preprocessing.*

7.2 Approximation by lower rank matrices and by structured matrices

Suppose we are given an $m \times n$ matrix $\tilde{A}$ for $m \ge n$, its numerical nullity r, a scaled random APP UV^H of rank r defining a well conditioned A-modification $\tilde{C} = \tilde{A} + UV^H$ of full rank, and a unitary matrix Q (e.g., $Q = Q(\tilde{B})$ for $\tilde{B} = \tilde{C}^{(I)}U$ in (3.4)) that closely approximates a $\operatorname{nmb}(A)$ where A denotes an unknown nearby matrix of rank $\rho = n - r$ for $r > 0$. (Alternatively we can approximate a $\operatorname{nmb}(A)$ by applying the augmentation echniques in Theorem 4.1.) Then the linear space $\operatorname{range}(I - QQ^H) = N(Q^H)$ of dimension ρ closely approximates range $\tilde{A}$, and so we can approximate the matrix $\tilde{A}$ with its orthogonal projection $\tilde{A}(I_n - QQ^H)$ (of rank ρ) onto this orthogonal complement.

Alternatively, one can at first compute the matrix $(\tilde{C}_-)^+ = (\tilde{A}^+ + \widehat{U}\widehat{V}^H)^+$ for a scaled random APP $\widehat{U}\widehat{V}^H$ of rank ρ (cf. equation (2.6) for UV^H replaced by $\widehat{U}\widehat{V}^H$), then compute the unitary matrix $\widehat{Q} = Q((\tilde{C}_-)^+\widehat{U})$ of rank ρ (to approximate the ρ-tail of the matrix $\tilde{A}^+$ or equivalently the ρ-head of the matrix $\tilde{A}$), and finally approximate the matrix $\tilde{A}$ with the matrix $\tilde{A}\widehat{Q}\widehat{Q}^H$ of rank ρ, which is the orthogonal projection of the matrix $\tilde{A}$ onto the range of the matrix $\widehat{Q}$. This is most attractive where the integer ρ is small, e.g., where we seek a low rank approximation $T - ZTZ^T$ to the displacement $\tilde{T} - Z\tilde{T}Z^T$ of a matrix $\tilde{T}$ that lies near a Toeplitz-like matrix T of a small displacement rank ρ [47], [64]. Having such an approximation available we could immediately approximate the matrix $\tilde{T}$ by the structured matrix T.

7.3 Approximate nmbs and A-preconditioning

Let us comment on approximating a $\operatorname{nmb}(A)$ by means of Algorithms 6.7 and 6.8. Suppose an $m \times n$ input matrix A has full rank, is ill conditioned, and has numerical nullity $r = \operatorname{nnul} A$, $r < n \le m$. Suppose two matrices U of size $m \times r$ and V of size $n \times r$ are random and properly scaled and have full rank r. In view of Sections 3.4 and 3.6 we can expect that the matrix $C = A + UV^H$ has full rank and is well conditioned. Then the ratio $t = ||AB||/(||A||\ ||B||)$ is small for $B = Q(C^+U)$ (cf. (3.2)), that is the matrix B closely approximates a $\operatorname{nmb}(A)$.

We detect if $r > \operatorname{nnul} A$ by observing that the ratio $||AB||/(||A||\ ||B||)$ is too large. If so, we can set $r \Longleftarrow r + 1$ and recursively reapply the same algorithm to the matrix A until we yield a matrix B such that $AB = 0$ (cf. Algorithm 6.7).

Alternatively we can reapply the algorithm to the matrix AB to compute the matrix $X = \operatorname{nmb}(AB)$, and then we can obtain and output the matrix $BX = \operatorname{nmb}(A)$ (cf. Algorithm 6.8). In this way we confine our numerical problems to the computations of and with the matrix AB of a

smaller size. Such computations require high accuracy [52, Section 7], but (as we recalled in Section 6.6) we can stay with double precision computations by employing the effective algorithms in [16], [27], [30], [38], [57], and [71] for sums and products and the extended iterative refinement in [52].

7.4 Perturbation and residual norm estimates in approximation of nmbs

Next we estimates the norm $||(C+E)^+ - C^+||$ in terms of the norms $||E||$ and $||E||_F$. Here and hereafter $||\cdot||_F$ denotes the Frobenius norm, $||M|| \le ||M||_F \le \sqrt{\rho}||M||$ for a matrix M of a rank ρ.

Lemma 7.1. *Let C and $C+E$ be two matrices of full rank. Then*

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

Proof. See [25, Section 5.5.5] for $\delta A = E$. □

Lemma 7.2. *Suppose C is a nonsingular matrix, E is a matrix of the same size, and $||C^{-1}E|| = \theta < 1$. Then $||I-(C+E)^{-1}C|| \le \frac{\theta}{1-\theta}$, so that $\delta_+||E|| = ||(C+E)^{-1} - C^{-1}|| \le \frac{\theta}{1-\theta}||C^{-1}||$.*

Proof. See [73, Theorem 1.4.18] for $P = -C^{-1}E$. □

In the remainder of this subsection we directly link the perturbation norms $||E||$ and $||E||_F$ with the relative residual norm $||AC^+U\mathbf{x}||/(||A||\ ||C^+U\mathbf{x}||)$. We assume that A is a full rank approximation of a rank deficient matrix $A-E$ and simplify our notations by dropping the character "tilde" and writing A and C instead of $\tilde{A}$ and $\tilde{C}$.

Theorem 7.1. *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 $\mathbf{y}$, premultiply it by C^+, substitute $C^+C = I_n$, and obtain that $\mathbf{y} = C^+A\mathbf{y} + C^+U\mathbf{z}$ for $\mathbf{z} = V^H\mathbf{y}$. □

The theorem implies that a vector $\mathbf{y}$ lies near the space range(C^+U) provided the norm $||A\mathbf{y}||$ is small and the norm $||C^+||$ is not very large. Conversely, our next theorem bounds the norm $||A\mathbf{y}||$ for the vectors $\mathbf{y} \in$ range(C^+U).

Theorem 7.2. *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 $||E|| \le ||E||_F = \delta$, $||A|| = 1$, 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 7.1 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 7.2 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 $m = n$. □

8 Generating and improving A-preconditioners

8.1 Improving A-preconditioners via orthogonalization

Let us come back to the notations of Section 7.1, where A is an $n \times n$ rank deficient matrix with a positive nullity $r < n$, whereas $\tilde{A} = A + E$ is an ill conditioned matrix of full rank. Then we can obtain a crude A-preconditioner and the integer $\operatorname{nnul} \tilde{A}$, e.g., by extending the algorithms in Section 6 to approximation of nmbs. In this subsection we refine such an A-preconditioner.

Suppose that U and V be a pair of unitary matrices and UV^H is an APP of the rank r such that the A-modification $C = A + UV^H$ has full rank. We may have $\operatorname{cond} C > \operatorname{cond} A$ and even $\operatorname{cond} C \gg \operatorname{cond} A$, but the following transform serves as a remedy,

$$(U \Longleftarrow Q(C^+U), \quad V \Longleftarrow Q(C^{+H}V)). \tag{8.1}$$

With the new APP UV^H the A-modification $C = A + UV^H$ still has full rank and, in virtue of our next theorem, shares the condition number with the matrix A.

Theorem 8.1. *Assume an $n \times n$ matrix A of a rank $\rho < n$ such that $\sigma_1(A) \geq 1 \geq \sigma_\rho(A)$ and let U and V be a pair of $n \times r$ unitary matrices such that $r = n - \rho = \operatorname{nul} A$ and the matrix $C = A + UV^H$ is nonsingular. Let $U_1 = Q(C^+U)$ and $V_1 = Q(C^{+H}V)$ denote the respective updates of the matrices U and V according to policy (8.1). Then the matrix $A + U_1V_1^H$ is nonsingular and* $\operatorname{cond}(A + U_1V_1^H) = \operatorname{cond} A$.

Proof. Due to Theorem 3.1, the updated matrices U_1 and V_1 remain the right and left nmbs for the matrix A, respectively. Let $A = \sum_{j=1}^{\rho} \sigma_j \mathbf{s}_j \mathbf{t}_j^H$ be an SVD of the matrix 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 $A + U_1V_1^H = \sum_{j=1}^{r} \mathbf{u}_j \mathbf{v}_j^H + \sum_{j=1}^{\rho} \sigma_j \mathbf{s}_j \mathbf{t}_j^H$. Theorem 8.1 follows because $r = n - \rho$ and $\sigma_1 \geq 1 \geq \sigma_\rho$. □

Suppose the singular matrix A in this theorem is well conditioned. Then so is the nonsingular matrix $A + U_1V_1^H$ as well as all nearby matrices. Therefore, the APP $U_1V_1^H$ preconditions all ill conditioned matrices $\tilde{A} = A + E$ lying near the matrix A provided $\operatorname{nnul} \tilde{A} = \operatorname{rank}(U_1V_1^H)$. According to the test results in [54, Table 7.2], transformation (8.1) substantially increases the preconditioning power of an APP $U_1V_1^H$ for such an average matrix $\tilde{A}$ and for $U = U_1$, $V = V_1$.

8.2 Generating and improving A-preconditioners via inflation and compression

Suppose we have an upper bound r^+ on the unknown number $r = \operatorname{nnul} A$ of small (positive and zero) singular values $\sigma_{n-r+1}(A), \dots, \sigma_n(A)$ of an $m \times n$ input matrix A for $m \geq n$. To approximate an r-tail of the matrix A, we can generate a scaled random APP UV^H of rank r^+, compute the A-modification $C = A + UV^H$, approximate the matrix C^+U, and test whether the matrix AC^+U has a small norm (within a fixed tolerance bound). If not so, we can choose a candidate integer $r < r^+$ and approximate the r-tail $\mathbb{T}_{n-r,r}$ of the matrix A by extending transform (8.1) to the compression of the APP as follows.

Flowchart 8.1. Inflation/Compression of an APP *(cf. [87]).*

1. (Generation of an inflated APP.) *Generate an APP UV^H of rank r^+.*
2. (Approximation of a nmb.) *Compute two unitary or well conditioned matrix bases $T(U)$ and $T(V)$ for the r-tails of the matrices AC^+U and $A^HC^{+H}V$, respectively. (If $m = n$ and the matrices U and V are unitary, then in virtue of Corollary 3.2 we can compute just the left and right r-tails of the matrix $G = I_{r^+} - V^HC^+U$.)*
3. (Compression.) *Compute and output the new generators $U \Longleftarrow Q(C^+UT(U))$ and $V \Longleftarrow Q(C^{+H}VT(V))$ and the new APP UV^H.*

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

X. Wang in [87] has applied an algorithm similar to Flowchart 8.1 to 10×10 Hilbert input matrices $A = (\frac{1}{i+j-1})_{i,j=1}^{10}$ and has consistently arrived at $\operatorname{cond} C \approx \frac{\sigma_1(A)}{\sigma_{10-r^+}(A)}$ in his extensive tests for various choices of positive $r^+ \le 10$ and $r < r^+$.

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 8.1 complements this choice to yield A-preconditioners of rank $\operatorname{nnul} A$.

9 A-preprocessing and matrix sparseness and structure

Suppose an input matrix A as well as an APP UV^H can be multiplied by a vector fast. (This property holds for APPs of small ranks as well as sparse and structured APPs of any rank, e.g., the APPs in Example 3.1.) Then we can multiply the A-modification C by a vector fast, and this makes iterative algorithms attractive for computing the matrices C^+U and V^HC^+. In particular the iterative refinement and *the Conjugate Gradient algorithms become attractive* if such a structured preprocessing turns an ill conditioned sparse or structured matrix A into its well conditioned A-modification C.

Direct algorithms can be also effective as long as we preserve matrix structure in A-preprocessing and the subsequent computation of a nmb, that is in the computation of the APP UV^H and either the matrices $C = A + UV^H$, V^HC^+, C^+U, $G = I - V^HC^+U$, and G^{-1} or the matrices $H = I + V^HAU$, H^{-1}, $(C_-)^+ = A - AUH^{-1}V^HA$, and $A^+ = C_- - UV^H$ where we assume that these matrices have full rank (cf. Section 2.5). This involves only a small number of matrix additions, multiplications, and inversions. They do not destroy matrix structure (although usually spoil it a little), and we can perform these operations fast. We can employ APPs in Example 3.1 in Section 3.6, [54, Examples 4.1–4.6], [55, Examples 1–6] to match the structure of the input matrix A. It can be effective if one first modifies the structure of the matrix A by applying displacement transformations in [44] and [48], e.g. transforms the input matrix A with the structure of Vandermonde or Cauchy type into a Toeplitz-like matrix (cf. Remark 2.1), and then applies A-preprocessing.

In the augmentation $A \Longrightarrow K = \begin{pmatrix} A & U \\ S & W \end{pmatrix}$ in Section 4 where $W \in \mathbb{C}^{r \times r}$, we can even more readily preserve the structure of the matrix A by choosing appropriate matrices W, S and U. Namely we can obtain Toeplitz, Hankel, Vandermonde, or Cauchy matrix K if so is the matrix A, and this still allows $2r$ random parameters (or r in the Vandermonde case).

Now assume a Toeplitz-like matrix A and its expression via its displacement $A - ZAZ^T = \sum_{k=1}^{d} \mathbf{g}_k \mathbf{h}_k^T$ (cf. (2.2)), augment every column vector $\mathbf{g}_k$ as well as $\mathbf{h}_k$ for $k = 1, \dots, d$ by appending r random coordinates at its bottom (for a total of $2dr$ random parameters), and then define the Toeplitz-like matrix K via expression (2.3) for $e = f = 0$. This defines the matrix $K = \begin{pmatrix} A & U \\ S & W \end{pmatrix}$ with $W \in \mathbb{C}^{r \times r}$ and the matrices S, U, and W randomized with $2dr$ random parameters. Similar augmentations of matrices A with the structures of Hankel, Vandermonde, and Cauchy types also preserve the structure type and displacement rank allowing $s(r, d)$ random parameters where $s(r, d)$ equals $2dr$, $(2d + 1)r$, and $2(d + 1)r$ for matrices with the structures of Hankel, Vandermonde, and Cauchy types, respectively (cf. [48, Section 4.4] on representations of such matrices via their displacements).

Clearly we can also preserve sparseness as well as the sparseness structure in the augmentation, in particular we can preserve a lower bandwidth l and an upper bandwidth u involving $s(r, d) = (l + u + 1)r$ new random entries.

Finally 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^HC^+ of size $r \times n$ and C^+U of size $n \times r$ approximate some matrix bases for the left and right r-dimensional singular spaces associated with the r smallest singular values of the matrix A. This holds even where these singular spaces have no structured matrix bases.

10 Numerical experiments

In a series of numerical experiments performed in the Graduate Center of the City University of New York, we tested our algorithms for computing nmbs and null vectors of general and Toeplitz matrices. The tests were conducted on a Dell server with a dual core 1.86 GHz Xeon processor and 2G memory running Windows Server 2003 R2. The test Fortran code was compiled with the GNU gfortran compiler within the Cygwin environment. Random numbers were generated with the random_number intrinsic Fortran function assuming the uniform probability distribution over the range $[-1, 1) = \{x : -1 \le x < 1\}$. To shift to the range $\{y : b \le y \le a+b\}$ for fixed real a and b, we applied the linear transform $x \Longrightarrow y = ax + b$. CPU time was measured with the mclock function. We computed QR factorizations and SVDs by applying the LAPACK procedures DGEQRF and DGESVD, respectively.

10.1 Generation of singular structured matrices

We present our test results for random singular circulant and symmetric Toeplitz matrices. Similar tests with random general Toeplitz matrices have produced similar results [60].

a) Generation of singular circulant input matrices

For $n = 2^h$ being the powers of two, we generated real singular $n \times n$ circulant matrices $A = (a_{i,j})_{i,j=0}^{n-1}$ by fixing their first columns $\mathbf{a}$ as follows. For every odd integer $i = 2j - 1$, we randomly sampled the value $a_{i,0}$ in the range $[-1, 1)$ and then set $a_{i,0} = a_{i-1,0}$ for all even i. A factorization in [10] implies that the resulting circulant matrices are singular.

b) Generation of symmetric Toeplitz matrices with nullity one.

To generate an $n \times n$ real symmetric singular Toeplitz matrix, we first sampled $n - 1$ random entries $a_{0,j} = a_{j,0}$ for $j = 0, 1, \dots, n-2$ in the range $[-1, 1)$, then defined the $(n-1)^2$ entries $a_{i+1,j+1} = a_{i,j}$ for $i, j = 0, 1, \dots, n-2$, and set $a_{n-1,0} = a_{0,n-1} = 0$, to obtain an $n \times n$ real symmetric Toeplitz matrix $A_0 = (a_{i,j})_{i,j=0}^{n-1}$. Then we computed the entries $x_{0,0}$ and $x_{0,n-1}$ of its inverse $A_0^{-1} = (x_{i,j})_{i,j=0}^{n-1}$ and changed the pair of the $(n-1, 0)$th and the $(0, n-1)$st entries into $a_{n-1,0} = a_{0,n-1} = -1/(x_{0,0} + x_{0,n-1})$. (As we expected in virtue of Corollary 3.3, we always had $(x_{0,0} + x_{0,n-1}) \det A_0 \neq 0$ in our tests. Had $x_{0,0} + x_{0,n-1} = 0$, we could have regenerated the matrix A_0, whereas had it been singular, we would have written $A = A_0$ and output it.)

The resulting matrix $A = (a_{i,j})_{i,j=0}^{n-1}$ had nullity one. Indeed, being a rank-one modification of a nonsingular matrix A_0, it had nullity at most one, whereas $A\mathbf{x} = \mathbf{0}$ for $\mathbf{x} = A_0^{-1}(\mathbf{e}_0 + \mathbf{e}_{n-1})$ because

$$A = A_0 - \frac{1}{x_{0,0} + x_{0,n-1}}(\mathbf{e}_0\mathbf{e}_{n-1}^T + \mathbf{e}_{n-1}\mathbf{e}_0^T),$$

$\mathbf{e}_{n-1}^T\mathbf{x} = x_{n-1,0} + x_{n-1,n-1}$, $\mathbf{e}_0^T\mathbf{x} = x_{0,0} + x_{0,n-1}$, $x_{n-1,0} = x_{0,n-1}$ (since the matrix X was symmetric), and $x_{n-1,n-1} = x_{0,0}$ (since the matrix X was persymmetric, that is since the matrix XJ was symmetric).

10.2 Augmentation of singular Toeplitz matrices and the computation of their null vectors

We computed null vectors of the matrices A from Section 10.1 based on Algorithm 6.9 for $r = 1$. The computation preserved the Toeplitz structure and the symmetry but not the circulant structure of circulant inputs. (We could have immediately computed the null vectors of a circulant matrix based on its factorization in [10], but instead we used circulant inputs just as additional representatives of the class of Toeplitz inputs.) Namely, we first generated singular circulant and symmetric Toeplitz matrices A according to the previous subsection and randomly sampled scalars $s_0 = u_0$ in the range $[-1, 1)$. Then we defined the $(n+1) \times (n+1)$ Toeplitz matrices $K = (k_{i,j})_{i,j=0}^{n} = \begin{pmatrix} A & \mathbf{u} \\ \mathbf{s}^T & w \end{pmatrix}$ where

the entries w and the vectors $\mathbf{s} = (s_i)_{i=0}^{n-1}$ and $\mathbf{u} = (u_i)_{i=0}^{n-1}$ were completely defined by the matrices A and scalars $s_0 = u_0$ due to the Toeplitz conditions $k_{i+1,j+1} = k_{i,j}$ for all $i, j = 0, \dots, n-1$. To every such a matrix K we applied our Algorithm 6.9 for $r = 1$, $U = \mathbf{u}^T$, $S = \mathbf{s}$, and $W = w$ to compute a null vector of the matrix A given by the vector $(I_n, 0)K^{-1}\begin{pmatrix}\mathbf{u}\\0\end{pmatrix}$. The computation amounted to the solution of a nonsingular Toeplitz linear system of equations. For this task we applied the code in [78], based on the algorithms in [29], [79], [80]. We also obtained the null vectors of the same matrices A based on computing their QR factorizations and SVDs. We have a little decreased the CPU time by using QR (rather than QRP) factorization. The latter one, that is QR factorization with pivoting (performed by LAPACK procedures DGEQPF and DGEQP3) is recommended for dealing with ill conditioned inputs [25, Section 5.5], but we avoided them in our tests.

Remark 10.1. *We could have employed two distinct parameters s_0 and w_0 (instead of the single one $s_0 = w_0$) at the price of giving up the symmetry but not the Toeplitz structure.*

10.3 Output data in the tests with Toeplitz matrices

Tables 10.1 and 10.2 cover our computation of null vectors for circulant and symmetric Toeplitz input matrices, respectively. The tables show the CPU time of this computation for each of the three methods based on Algorithm 6.9, QR factorization and SVD as well as the ratios of these CPU time data. The abbreviations "Alg. 6.9", "QR", and "SVD" point out to the respective algorithms. The ratios are displayed in the last two columns of the table. The CPU time is measured in terms of the CPU cycles. One can convert them into seconds by dividing them by a constant CLOCKS_PER_SEC, which is 1000 on our platform.

In all our tests the computed approximate null vectors $\mathbf{y}$ had relative residual norms $\frac{||A\mathbf{y}||}{||A||\ ||\mathbf{y}||}$ of the order of 10^{-17}.

All data are average over 100 tests for each input size 2^k from 256 to 8192. The table entries are marked by a "-" where the tests required too long runtime and were not completed.

10.4 Generation of unstructured input matrices and APPs

For $n = 64$ and $n = 128$, we computed the $n \times n$ unstructured input matrices A numerically, with double precision, as the products $S\Sigma T^T$ (cf. [27, Section 28.3]). Here we generated random real orthonormal matrices S and T, being the Q-factors in the QR factorization of matrices with random integer entries from the range $[-10^4, 10^4)$ and with positive diagonal entries of the R-factors. We defined diagonal matrices $\Sigma = \mathrm{diag}(\sigma_i)_{i=1}^n$ with the diagonal entries $\sigma_1, \dots, \sigma_1$ from one of the four following classes.

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

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

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

Class 4. $\sigma_i = \frac{1}{i}$ for $i = 1, \dots, n-k-l$, $\sigma_i = \frac{10^{-9}}{i-n+k+l}$ for $i = n-k-l+1, \dots, n-k$, $\sigma_i = \frac{10^{-14}}{i-n+k}$ 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 $n = 128$.

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

10.5 Computation and approximation of nmbs with A-preprocessing

For each pair $\{n, r\}$, $n = 64$ and $n = 128$, we tested 1000 instances of the input matrices A, U and V defined in the previous subsection.

In these tests we computed approximate nmbs by applying Algorithm 6.5 for Classes 1 and 2 and Algorithm 6.8 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 10.3 and 10.4 show the results of our tests performed with double precision and without using the extended iterative refinement from [52].

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 10.5 and 10.6 display the results of these tests. As we expected, in the case of matrices A of Classes 2 and 4, the residual norms decrease only to the level of 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 rapidly decreasing towards zero. (We stopped the iterative refinement process with the ratios at the levels well below 10^{-40}.)

Table 10.1: CPU time (in cycles) for computing null vectors of circulant matrices

size	**Alg. 6.9**	**QR**	**SVD**	**QR/Alg. 6.9**	**SVD/Alg. 6.9**
256	3.0	18.8	261.5	6.3	87.2
512	7.3	147.9	4220.9	20.3	578.2
1024	16.1	1538.3	70452.5	97.1	4445.8
2048	35.5	11748.3	–	342.1	–
4096	78.7	–	–	–	–
8192	170.4	–	–	–	–

Table 10.2: CPU time (in cycles) for computing null vectors of symmetric Toeplitz matrices

size	**Alg. 6.9**	**QR**	**SVD**	**QR/Alg. 6.9**	**SVD/Alg. 6.9**
256	4.7	18.0	291.5	3.8	62.0
512	6.9	148.9	4728.4	21.6	685.3
1024	15.7	1536.9	78653.3	98.6	5046.2
2048	35.3	11747.8	–	343.2	–
4096	79.4	–	–	–	–
8192	170.4	–	–	–	–

11 Conclusion

We conclude with a brief sumary of our present advances and some examples of their extensions and applications, partly covered in the papers [59] – [61] and [68].

Table 10.3: residual norms for 64×64 unstructured 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 10.4: residual norms for 128×128 unstructured 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}

11.1 Brief sumary

Standard solution algorithms for homogeneous linear systems of equations involves pivoting, orthogonalization or SVD, which are expensive particularly in the case of structured inputs. Our noncostly alternative randomization techniques are expected to remove degeneracy of rank deficient matrices and furthermore to yield a substantial decrease of the condition number for quite a general class of ill conditioned inputs. We proved these results for scaled random general preprocessors, but in our extensive tests, we observed the same power even where we used structured preprocessors defined by a small number of bounded integer parameters. This has led us to dramatic acceleration of the standard algorithms in the case of Toeplitz inputs, both in terms of the flop count and the CPU time involved. We extended our algorithms to some other fundamental matrix computations, yielding new insights and significant acceleration. We included detailed analysis; part of it (e.g., our estimates for the impact of randomized preprocessing on condition numbers, variations of the Sherman–Morrison–Woodbury classical formula, and links of preprocessing to aggregation, Newton's iteration, and iterative refinement) can be of independent interest. Our further applications include acceleration of the solution of nonhomogeneous linear systems of equations, approximation of a matrix by a nearby matrix having a lower rank or a lower displacement rank, eigen-solving by means of the inverse iteration, and root-finding via matrix methods. Some of these applications are briefly covered in our next subsections.

11.2 Eigen-solving

Matrix eigen-solving, that is approximation of eigenvalues of a matrix and the associated eigenspaces is one of the two most fundamental problems in matrix computations [25], [74]. Inverse iteration, also called Rayleigh quotient (hereafter RQ) iteration is among most popular solution algorithms. Given a matrix A and an approximation $\lambda^{(0)}$ to its simple eigenvalue, one can fix a crude initial approximation to its normalized associated eigenvector $\mathbf{y}^{(0)}$, $||\mathbf{y}^{(0)}|| = 1$ and recursively compute

Table 10.5: residual norms for 64×64 unstructured 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 10.6: residual norms for 128×128 unstructured 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}

vectors $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i+1)}$ and scalars $\lambda^{(i+1)}$ as follows,

$$\mathbf{x}^{(i)} = (A^{(i)})^{-1}\mathbf{y}^{(i)},\ \mathbf{y}^{(i+1)} = \mathbf{x}^{(i)}/||\mathbf{x}^{(i)}||, \tag{11.1}$$

$$\lambda^{(i+1)} = \lambda^{(i)} + (\mathbf{y}^{(i+1)})^H A^{(i)}\mathbf{y}^{(i+1)} = \lambda^{(i)} + (\mathbf{y}^{(i+1)})^H \mathbf{y}^{(i+1)}/||\mathbf{x}^{(i)}||^2 \tag{11.2}$$

for $A^{(i)} = A - \lambda^{(i)}I$, $i = 0, 1, \ldots$. One can stop the iteration where

$$||A^{(k)}\mathbf{y}^{(k)}|| \le t \tag{11.3}$$

and t is a fixed tolerance or $t = t'|\lambda^{(k)}|$ for a fixed tolerance t'. The iteration has local quadratic convergence and allows some flop saving simplifications. In particular one can check stopping criterion periodically in $h > 1$ iteration loops, rather than in every loop. Furthermore one can simplify updating the eigenvalues as follows,

$$\lambda^{(i+1)} = \lambda^{(i)} + y_j^{(i)}/x_j^{(i)}. \tag{11.4}$$

Here we assume that $y_j^{(i)}$ and $x_j^{(i)}$ denote the j-th coordinates of vectors $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$, respectively, and $x_j^{(i)} \neq 0$, and hereafter we call the ratios $y_j^{(i)}/x_j^{(i)}$ *simple quotients* or *SQs* versus RQs $(\mathbf{y}^{(i+1)})^H\mathbf{y}^{(i+1)}/||\mathbf{x}^{(i)}||^2$ in (11.2). Practically one can choose the integer j maximizing the values $|x_j^{(i)}|$ over a fixed or random subset of, say three integers in the set $\{1, 2, \ldots, n\}$.

Even in the case of the RQ iteration, the iteration cost is generally dominated at the stage of solving linear system (11.1), which becomes more and more ill conditioned as the approximations $\lambda^{(i)}$ converge to an eigenvalue. The resulting growth of rounding errors does not destroys convergence,

but ill conditioning complicates or even precludes application of some highly effective iterations such as Conjugate Gradient algorithms and iterative refinement.

A-preprocessing, however, is a simple way around due to Corollary 2.1 and our results in Section 3.6. Indeed in virtue of Corollary 2.1 we can shift from linear system (11.1) to the following one,

$$\mathbf{z}^{(i)} = (C^{(i)})^{-1}\mathbf{u}^{(i)},\ \mathbf{y}^{(i+1)} = \mathbf{z}^{(i)}/||\mathbf{z}^{(i)}|| \tag{11.5}$$

for $C^{(i)} = A^{(i)} + (\mathbf{u}^{(i)})^H\mathbf{v}^{(i)}$, whereas according to the results in Section 3.6 the matrix $C^{(i)}$ tends to be well conditioned in the case of a single isolated eigenvalue λ and random scaled APP $\mathbf{u}^{(i)}(\mathbf{v}^{(i)})^H$. In [61] local quadratic convergence of this modified iteration is proved, and the test results show that the number of iteration steps until convergence is of the same order with and without A-preprocessing. (In [68] such a proof and similar test results are presented also, but only for the variant of the iteration where $\mathbf{u}^{(i)} = \mathbf{y}^{(i)}$ for all i.) Under equation (11.5) we can simplify stopping criterion (11.3) as follows,

$$|1 - (\mathbf{v}^{(k)})^H\mathbf{z}^{(k)}|\ ||\mathbf{u}^{(k)}|| \le t||\mathbf{z}^{(k)}||$$

for $\mathbf{z}^{(i)}$ in (11.5). Indeed $\mathbf{r}^{(k)} = A^{(k)}\mathbf{z}_k = A^{(k)}(C^{(k)})^{-1}\mathbf{u}^{(k)}$, $A^{(k)} = C^{(k)} - \mathbf{u}^{(k)}(\mathbf{v}^{(k)})^H$, $C^{(k)}\mathbf{z}^{(k)} = \mathbf{u}^{(k)}$, and so $A^{(k)}(C^{(k)})^{-1} = I - \mathbf{u}^{(k)}(\mathbf{v}^{(k)})^H(C^{(k)})^{-1}$, $\mathbf{r}^{(k)} = \mathbf{u}^{(k)} - \mathbf{u}^{(k)}(\mathbf{v}^{(k)})^H(C^{(k)})^{-1}\mathbf{u}^{(k)} = \mathbf{u}^{(k)}(1 - (\mathbf{v}^{(k)})^H\mathbf{z}^{(k)})$, and $||A^{(k)}\mathbf{y}_k|| = \frac{||\mathbf{r}^{(k)}||}{||\mathbf{z}^{(k)}||} = |1 - (\mathbf{v}^{(k)})^H\mathbf{z}^{(k)}|\ \frac{||\mathbf{u}^{(k)}||}{||\mathbf{z}^{(k)}||}$.

Instead of aiming A-preprocessing at the decrease of the condition number, one can direct it towards simplifying linear system (11.1). E.g., suppose A is an upper Hessenberg matrix H. Then with an APP of rank one we can turn it into a 2×2 block triangular matrix with two Hessenberg diagonal blocks of half size. In the next subsection A-preprocessing enables similar but more significant simplification of the inverse iteration in its application to root-finding.

Like the inverse iteration itself, its latter modifications can be extended to approximating eigenspaces where the eigenvalues are multiple or clustered.

We conclude this subsection by showing two other options of employing APCs in eigen-solving for an upper Hessenberg matrix H (cf. [56], [58]). In this case we shift the task to the matrix $(H - \mu I)^{-1}$ for a random scalar μ sampled in a fixed range $[-\beta, \beta]$ where $2\beta \approx ||H||$, say. For such a choice the matrix $H - \mu I$ is expected to be nonsingular, and then $(H - \mu I)^{-1} = R - \mathbf{u}\mathbf{v}^H$ where R is an upper triangular matrix. Actual computation of the matrix $\mathbf{u}\mathbf{v}^H$ is a delicate task [20], [81]–[84], but as soon as this is achieved, we can apply the inverse iteration with the APP $\mathbf{u}\mathbf{v}^H$. Its every iteration step essentially amounts to solving the triangular linear system $(R - \lambda^{(i)}I)\mathbf{z}^{(i)} = \mathbf{u}$. Having computed an eigendecomposition $(H - \mu I)^{-1} = G_\mu\Lambda_\mu G_\mu^{-1}$ where Λ_μ is an $n \times n$ diagonal matrix and G_μ is a nonsingular matrix, we immediately obtain the eigendecompositions $H - \mu I = G_\mu\Lambda_\mu^{-1}G_\mu^{-1}$ and $H = G_\mu(\Lambda_\mu^{-1} + \mu I)G_\mu^{-1}$.

In an alternative way of eigen-solving for the matrix $R - \mathbf{u}\mathbf{v}^H$ (cf. [7]), one successively computes the eigendecompositions $R = G^{-1}\Lambda G$, $\Lambda - G\mathbf{u}\mathbf{v}^HG^{-1} = F^{-1}\tilde{\Lambda}F$, $R - \mathbf{u}\mathbf{v}^H = (FG)^{-1}\tilde{\Lambda}FG$, and finally $H = (FG)^{-1}(\tilde{\Lambda} + \mu I)^{-1}FG$.

11.3 Root-finding

Root-finding for a univariate polynomial $p(x) = \sum_{i=0}^n p_i x^i = p_n\prod_{j=1}^n(x - \lambda_j)$, $p_n \neq 0$, is a classical and highly important problem, equivalent to approximating the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of the associated *companion* or *generalized companion* matrices. This equivalence is the basis for some of the most effective recent polynomial root-finders. In particular such a root-finder in [6] turned out to be competitive with the current best polynomial root-finders in the package MPSOLVE for approximating all roots of a polynomial (cf. [5]), has additional power for approximating only a single root or only the roots in a fixed region, and is highly effective also for solving the secular equation associated with a polynomial $p(x)$ [7], [21], [35]. Even a relatively minor acceleration of this algorithm can give it upper hand and make it the root-finder of choice. Next we employ A-preprocessing towards this decisive step.

First recall that the algorithms in [6] rely on application of RQ and SQ iterations (11.1)–(11.4) to the Frobenius companion matrix $F_{\mathbf{p}} = Z - \mathbf{p}\mathbf{e}_{n-1}^T$ or the generalized companion matrix $C_{\mathbf{p}} = \mathrm{diag}(s_i)_{i=1}^n - \mathbf{u}\mathbf{v}^H$ (cf. Remark 11.1 below). Here $\mathbf{p} = (p_i/p_n)_{i=0}^{n-1}$, $Z = F_{\mathbf{0}} = (z_{i,j})_{i,j=0}^{n-1}$ is the

downshift matrix (cf. Section 2.2), and one can choose any n-tuple of distinct scalars $s_1, \dots, s_n$ (possibly crude approximations to the roots) and any pair of vectors $\mathbf{u} = (u_i)_{i=1}^n$ and $\mathbf{v} = (v_i)_{i=1}^n$ such that $u_i v_i = -p(s_i)/q'(s_i)$ for $q(x) = \prod_{i=1}^n (x - s_i)$, $i = 1, \dots, n$.

At every iteration step the computational cost is dominated by the cost of the solution of a linear system of equations with a shifted matrix $M - \mu I$ for a scalar μ and $M = F_{\mathbf{p}}$ or $M = C_{\mathbf{p}}$. This takes $6n - 5$ flops (based on Gaussian elimination) or $9n$ flops [6], respectively.

Now A-preprocessing with the APPs $\mathbf{p}\mathbf{e}_{n-1}^T$ enables us to decrease the cost to $2n-1$ flops where $M = F_{\mathbf{p}}$. Note that the matrix $F_{\mathbf{p}} + \mathbf{p}\mathbf{e}_{n-1}^T - \mu I = Z - \mu I$ is well conditioned for $\mu \geq 1$, which typically holds for the close approximations μ to the eigenvalues λ such that $|\lambda| > 1$. If, however, $p_0 \neq 0$ (which we can assume w.l.o.g.) and if we seek an eigenvalue λ such that $|\lambda| < 1$, then we can shift to the reverse polynomial $x^n p(1/x) = \sum_{i=0}^n p_{n-i} x^i = p_0 \prod_{j=1}^n (x - 1/\lambda_j)$, thus mapping this eigenvalue into its reciporocal $1/\lambda$.

Likewise with the APP $\mathbf{u}\mathbf{v}^T$ we can decrease the cost to $2n$ flops where $M = C_{\mathbf{p}}$.

According to the cited analysis in [61], this acceleration of every iteration step preserves local quadratic convergence of the iteration and, according to the results of experiments in [61], rather little affects global convergence.

Remark 11.1. *There are well known techniques that reduce eigen-solving for general matrix to the case of companion matrices [25, page 348], but the following simple reduction to the DPR1 case seems to be less explored. Recall that an $n \times n$ DPR1 matrix $C_{\mathbf{p}}$ associated with a fixed polynomial $p(x)$ of degree n can be defined by the values $p(s_i)$ and $q'(s_i)$ at n distinct points $s_1, \dots, s_n$. Given an $n \times n$ matrix M, we can fix a scalar $a \approx 0.1/||M - \mathrm{trace}(M)||$, readily compute the values $c(\omega_n^i) = \det(a(M - \mathrm{trace}(M)) - \omega_n^i I)$ of the characteristic polynomial of the matrix aM for a primitive n-th root of unity ω_n and $i = 1, \dots, n$, and obtain a DPR1 matrix $C_{\mathbf{p}}$ that shares the eigenvalues with the matrix M. (Note that the matrices $a(M - \mathrm{trace}(M)) - \omega_n^i I$ are diagonally dominant and can be readily factorized.) Then the algorithm in [6] as well as a number of other effective eigen-solvers for the DPR1 matrices can rapidly produce and then refine crude approximations to the eigenvalues of the matrix M.*

11.4 Matrix inversion with A-preprocessing, SMW and dual SMW formulae, and residual correction

Given an $n \times n$ matrix M and an initial approximation X_0 to its inverse or generalized inverse, one can rapidly refine this approximation with *Newton's iteration*, $X_{i+1} = X_i(2I - MX_i)$, $i = 0, 1, \dots$, which can be traced back to Hotelling 1933 and Schultz 1933 (cf. [49], [65], and the bibliography therein on this subject). The residuals $R_i = MX_i - I$ are squared in each step, $R_{i+1} = R_i^2$, which implies global quadratic convergence, right from the start if $||R_0|| < 1$. The known initialization policies support the bounds $||R_0|| < 1 - \frac{2n}{(1+n)(\mathrm{cond}\, M)^2}$ (cf. [65]), which makes the convergence highly sensitive to the value $\mathrm{cond}\, M$. This is even more critical for *Newton's structured iteration* with *recompression*, $X_{i+1} = c(Y_i)$, $Y_i = X_i(2I - MX_i)$, $i = 0, 1, \dots$, where $c(Y)$ is the compression function that recovers the structure of the approximate inverses X_i partly lost in the Newton's transition to Y_i. E.g., a Newton's step can triple the displacement rank d_i of the matrix X_i, but we can periodically set to zero all singular values of the displacement of the matrix Y_i except for the d_i largest ones, to recover the structure and thus to perform the Newton's steps fast. (These techniques were proposed in [45]–[47]. See [8], [49], [51], [64], [66], and the bibliography therein on their variations and some subsequent work.) This and other recompression techniques, nontrivially extended to tensor decomposition in [41] (cf. also [39] and [40]), little affect convergence where $||R_i||$ is small, but can easily destroy it otherwise. Therefore, as soon as we precondition the input matrix, we can rapidly approximate the inverse with high accuracy by performing Newton's steps (at a low cost in the case of structured inputs). The iteration is particularly valuable for the approximation of the generalized (Moor–Penrose) inverses of structured matrices.

We can observe the same features in other *residual correction processes* for computing inverses, generalized inverses and solutions of linear systems of equations, except that the celebrated iterative refinement is fast for structured input even with no recompression. For well conditioned inputs it

refines the approximate solution to a linear system advancing to any accuracy with linear rate and can be implemented with the IEEE standard double precision (cf. [52]).

Randomized scaled A-preprocessing and augmentation are natural tools towards the desired preconditioning, whereas the SMW and dual SMW formulae extend the solution from the preconditioned matrix to the original input. Even for ill conditioned inputs, we still invert matrices and solve linear systems with high accuracy involving no extended precision. We just perform (with double precision) more stages of iterative refinement or other residual correction iterations, highly effective in the case of structured inputs.

In the rest of this section we discuss the combined application of A-preprocessing with SMW formulae (2.4) or (2.5) and dual SMW formulae (2.6) and (2.8) to computing the matrices A^+ and A^{-1}. The SMW and dual SMW formulae reduce this task to the respective operations with the matrices $C = A + UV^H$ or $(C_-)^+ = (A^+ + UV^H)^+$.

Now suppose a nonsingular ill conditioned input matrix $A \in \mathbb{C}^{m\times n}$ has numerical nullity $r = \operatorname{nnul} A$, $r < n \le m$. Then according to Section 3.6, the transition $A \Longrightarrow C = A+UV^H$ is expected to yield a well conditioned matrix C in the case of scaled random generators $U \in \mathbb{C}^{m\times r}$ and $V \in \mathbb{C}^{n\times r}$. If $\operatorname{cond} A$ is large, whereas $\operatorname{cond} C$ is not, then the matrix $G = I_r - V^H C^+ U$ in the formulae (2.4) and (2.5) has a small norm [52, Section 7], and consequently many leading bits of the diagonal entries of the matrix G are cancelled in the process of the computation. (We have similar problems if we compute the matrix G as the solution of the matrix equation $V^H C^{(I)} A = GV^H$ in Theorem 3.2. Indeed we have $||V^H C^{(I)} A|| \le ||G||\ ||V||$.) These observations seem to imply that extended precision is required to obtain uncorrupted matrices G, $V^H C^{(I)} A$, and $AC^{(I)}U$, but we can stay with double precision if we apply the fast advanced algorithms in [16], [27], [30], [38], [57], [71] for sums and products as well as the extended iterative refinement in [52, Section 9], which computes highly accurate solutions of well conditioned linear systems of equations (cf. Section 7.3).

Instead of combining the SMW formulae with A-preprocessing, we can combine it with the augmentation $A \Longrightarrow K = \begin{pmatrix} A & U \\ S & W \end{pmatrix}$ for random and properly scaled matrices U, S, and W. Suppose that $K \in \mathbb{C}^{(m+r)\times(n+r)}$ and $W \in \mathbb{C}^{r\times r}$ are full rank matrices (according to Section 3.4 this holds with a probability near one for random matrices U, S, and W if $r \ge \operatorname{nul} A$). Recall that the generalised inverse F^+ of the Schur complement $F = A - UW^{-1}S$ is the leading principal block of the generalized inverse K^+, that is $F^+ = (I_m, 0)K^+ \begin{pmatrix} I_n \\ 0 \end{pmatrix}$. Now deduce from SMW formula (2.4) that $A^+ = F^+ - F^+ UG^{-1}V^H F^+$ for $V^H = W^{-1}S$ and $G = I_r - V^H F^+ U$. Our comments on the benefits and shortcomings of A-preprocessing in the previous paragraph can be reapplied.

Finally, instead of the SMW formulae (2.4) and (2.5), we can employ dual SMW formulae (2.6) and (2.8) if we obtain a crude estimate for the norm $||A^+||$ (cf., e.g., [12]) and compute the integer $q = \operatorname{nnul} A^+ = n - \operatorname{nnul} A$. Here is the respective procedure using A-preprocessing (cf. Remark 11.2 below).

Dual SMW Inversion Procedure

(a) Generate a pair of random matrices $U \in \mathbb{C}^{m\times r}$ and $V \in \mathbb{C}^{n\times r}$ and scale them to have the ratio $||UV^H||/||A^+|| = ||UV^H||\sigma_n(A)$ neither large nor small. (Then according to our study in Section 3.6, the matrix $C_- = A^+ + UV^H$ is expected to be well conditioned.)

(b) Compute the $q \times q$ matrix $H = I_q + V^H AU$ with high accuracy. (This stage involves no inverses, and we can apply the advanced algorithms in [16], [27], [30], [38], [57], [71].)

(c) Invert the matrix H.

(d) Compute the matrix $(C_-)^+ = A - AV^H H^{-1} UA$ with high accuracy. (If $q = 1$, then $U = \mathbf{u}$ and $V = \mathbf{v}$ are vectors, $H = h$ and $V^H H^{-1} U = \frac{1}{h}\mathbf{v}^H\mathbf{u}$ are scalars, and $(C_-)^+ = A(I - \frac{1}{h}\mathbf{v}^H\mathbf{u}A)$.)

(e) Compute its generalized inverse C_-.

(f) Compute and output the matrix $A^+ = C_- - UV^H$.

Remark 11.2. *For a nonsingular ill conditioned $n \times n$ matrix A, the matrix C_- is ill conditioned if $q < \operatorname{nnul} A^+ = n - \operatorname{nnul} A$, whereas the matrix H is ill conditioned if $q \geq \operatorname{nnul} A^+$ [52, Section 7]. If the matrix A has an unnkown numerical nullity* $\operatorname{nnul} A$*, we can extend the above procedure by incorporating binary or linear search or aggregation as in Algorithms 6.2–6.4.*

11.5 Solution of a linear system of equations as a null vector. Extension to a Toeplitz solver

A nonsingular linear system $A\mathbf{y} = \mathbf{b}$ of n equations with n unknowns is essentially equivalent to the homogeneous linear system $A\mathbf{y} - \theta z\mathbf{b} = \mathbf{0}$ with the $(n+1)$st additional unknown z and any nonzero scalar θ (one can choose this scalar satisfying $||\theta\ \mathbf{b}|| \approx ||A||$). It remains to compute a null vector $\mathbf{z} = (z_j)_{j=0}^n$ of the matrices $(A, -\theta\mathbf{b})$ and $\begin{pmatrix} 0 & 0 \\ A & -\theta\mathbf{b} \end{pmatrix}$ or $(-\theta\mathbf{b}, A)$ and $\begin{pmatrix} 0 & 0 \\ -\theta\mathbf{b} & A \end{pmatrix}$ (we can apply the algorithms in this paper or [59]) and then rescale it to obtain the solution vector $\mathbf{y}$ as a subvector.

For an ill conditioned matrix A with $\operatorname{nnul} A = 1$, the augmented matrix quite typically becomes well conditioned (see Section 3.6). If it does, we would need a highly accurate null vector to recover the vector $\mathbf{y}$, and we would apply the extended iterative refinement with double precision.

A-modifications of rank-one little change matrix structure, but let us fully preserve it for a nonsingular Toeplitz matrix A. The Gohberg–Semencul celebrated formula expresses the inverse A^{-1} through its two column vectors, the first $\mathbf{x} = A^{-1}\mathbf{e}_0$ and the last $\mathbf{z} = A^{-1}\mathbf{e}_{n-1}$, satisfying the linear systems $A\mathbf{x} = \mathbf{e}_0$ and $A\mathbf{z} = \mathbf{e}_{n-1}$. Each of the two systems is immediately reduced to computing a null vector of the $(n-1) \times n$ Toeplitz matrix T obtained by deleting the first or the last row of the matrix A.

We append a new row at the top (resp. bottom) of the matrix T, preserving its Toeplitz structure and still including one free entry t into the new Toeplitz matrix K. Then Theorem 4.1 implies that $\mathbf{s} = K^{-1}\mathbf{e}_0$ (resp. $\mathbf{s} = K^{-1}\mathbf{e}_{n-1}$) is a null vector of the matrix T. Suppose $\operatorname{nnul} A = 1$, the matrix T is well conditioned, and we choose a properly scaled random value t. Then according to our extensive tests, we would expect to arrive at a well conditioned matrix K and, if so, would readily approximate the solutions $\mathbf{s}$ to the linear systems $K\mathbf{s} = \mathbf{e}_h$ for $h = 0$ and $h = n-1$. We would need these solutions with high accuracy and would apply the extended iterative refinement.

This technique can be similarly combined with the Heinig's modification of the Gohberg–Semencul formula (cf., e.g., [48, Exercise 2.24b]) and can be extended to Toeplitz-like matrices A.

Appendix

A Estimating the condition numbers of Gaussian random sparse and structured matrices

Let us deduce some crude upper estimates for the condition numbers of Gaussian random sparse, Toeplitz, Hankel, Toeplitz-like and Hankel-like matrices under a conjecture extending the study in [13], [19], [75], and [76]. We begin with some definitions.

Definition A.1. *A nonnegative random function $X(n)$ is a* ppg function*, that is has* probabilistic polynomial growth*, if $F_{X(n)}(cn^d) \to 1$ as $n \to \infty$ for two fixed positive constants c and d.*

Definition A.2. *A sparse matrix is standard Gaussian random if all its nonzero entries are independent standard Gaussian random variables. Such a matrix is called nonsingular if the substitution of indeterminates for its random entries makes it nonsingular.*

Definition A.3. *A Toeplitz, Hankel, Toeplitz-like or Hankel-like matrix defined by k parameters (cf. Section 2.2) is standard Gaussian random if all its defining parameters are independent standard Gaussian random variables.*

Hereafter we write $\mathrm{diag}(M) = \mathrm{diag}(m_{i,i})_i$ for a matrix $M = (m_{i,j})_{i,j}$.

Theorem 2.3 implies the following result.

Theorem A.1. *Suppose $M = AB$, A and B are random $n \times n$ matrices, and $||A||$, $1/\sigma_n(A)$, $||B||$, and $1/\sigma_n(B)$ are ppg functions. Then $||M||$ and $1/\sigma_n(M)$ are ppg functions.*

Theorem A.2. *Suppose $B = (b_{i,j})_{i,j=0}^{n-1}$ is a standard Gaussian random diagonal matrix that is diagonal matrix whose diagonal entries are independent standard Gaussian random variables. Then $||B||$ and $1/\sigma_n(B)$ are ppg functions.*

Proof. Note that $||B|| = \max_i |b_{i,i}|$, $\sigma_n(B) = \min_i |b_{i,i}|$, $F_{||B||}(y) = 1 - \Phi_{\max_i |b_{i,i}|}(y)$, and $F_B(y) = 1 - \Phi_{\min_i |b_{i,i}|}(y)$. Apply Lemma 2.1 and deduce that $F_{||B||}(y) \geq (1 - \frac{1}{a}\sqrt{\frac{2}{\pi}}\exp(-\frac{a^2}{2}))^n$ for $y \geq a \geq 1$, whereas $1 - F_B(y) \geq (1 - y\sqrt{\frac{2}{\pi}})^n$ for $y \geq 0$. □

Conjecture A.1. *Suppose $||M||$ and $1/\sigma_n(M)$ are ppg functions, whereas A is a matrix filled with zeros and standard Gaussiam random variables independent of the matrix M and pairwise either coinciding or independent of each other. Then* $\mathrm{cond}(A + M)$ *is a ppg function.*

Theorem A.3. *Suppose Conjecture A.1 holds true. Let $B = (b_{i,j})_{i,j=0}^{n-1}$ be standard Gaussian random a) noningular sparse, b) Toeplitz, c) Hankel, d) Toeplitz-like or e) Hankel-like matrix. Then* $\mathrm{cond}\, B$ *is a ppg function.*

Proof. a) Clearly there is a permutation matrix P such that $M = \mathrm{diag}(PB)$ is a random diagonal matrix. Then $||M||$ and $1/\sigma_n(M)$ are ppg functions in virtue of Theorem A.2. Write $A = PB - M$ and deduce from Conjecture A.1 that $\mathrm{cond}(PB)$ is a ppg function. This proves part a) because $\sigma_j(B) = \sigma_j(M)$ for all j.

b) Write $M = \mathrm{diag}(B) = b_{0,0}I$, $A = B - M$. Clearly, $||M|| = |b_{0,0}| \leq ||B||$, and so $||M||$ is a ppg function. With probability one we have $b_{0,0} \neq 0$, and then $1/\sigma_n(M) = 1/|b_{0,0}|$ is a ppg function as well. Apply Conjecture A.1 to obtain part b).

Part c) follows from part b) because BJ is standard Gaussian random Toeplitz matrix for the unitary reversion matrix J.

Let us deduce part d) from part c) by induction based on equation (2.3) for $e = 1$ and $f = -1$.

Write $B_k = \sum_{h=1}^{k} Z_1(\mathbf{g}_h)Z_{-1}(\mathbf{h}_h)^T$. It is sufficient to prove by induction that $||B_k||$ and $1/\sigma_n(B_k)$ are ppg functions for $k = 1, \ldots, d$.

We first prove this for $k = 1$. Indeed $||Z_1(\mathbf{g}_1)||$, $1/\sigma_n(Z_1(\mathbf{g}_1))$, $||Z_{-1}(\mathbf{h}_1)||$, and $1/\sigma_n(Z_{-1}(\mathbf{h}_1))$ are ppg functions in virtue of Theorem 3.9 and Remark 3.4. Therefore $||B_1||$ and $1/\sigma_n(B_1)$ are ppg functions in virtue of Theorem A.1.

By inductive assumption let $||B_k||$ and $1/\sigma_n(B_k)$ be ppg functions for $k < l \leq d$ and let us extend this property to $k = l$. Write $M = B_{l-1}Z_{-1}(\mathbf{h}_l)^{-T}$ and observe that $||M||$ and $1/\sigma_n(M)$ are ppg functions in virtue of Theorem A.1 (because $||Z_{-1}(\mathbf{h}_l)^{-T}||$ and $1/\sigma_n(Z_{-1}(\mathbf{h}_l)^{-T}) = ||Z_{-1}(\mathbf{h}_l)||$ are ppg function in virtue of Theorem 3.9 and Remark 3.4).

Write $A = Z_1(\mathbf{g}_l)$, apply Conjecture A.1, and deduce that $||Z_1(\mathbf{g}_l) + M||$ and $1/\sigma_n(Z_1(\mathbf{g}_l) + M)$ are ppg functions. Now recall that $B_l = (Z_1(\mathbf{g}_l) + M)Z_{-1}(\mathbf{h}_l)^T$ and apply Theorem A.1 to deduce that $||B_l||$ and $1/\sigma_n(B_l)$ are also ppg functions. This completes the inductive proof of part d).

Part e) follows from part d) because BJ is standard Gaussian random Toeplitz-like matrix for the unitary reversion matrix J. □

References

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

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

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

[4] R. R. Bitmead, B. D. O. Anderson, Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations, *Linear Algebra and Its Applications*, **34**, 103–116, 1980.

[5] D. A. Bini, G. Fiorentino, Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder, *Numerical Algorithms*, **23**, 127–173, 2000.

[6] D. A. Bini, L. Gemignani, V. Y. Pan, Inverse Power and Durand/Kerner Iteration for Univariate Polynomial Root-finding, *Computers and Mathematics (with Applications)*, **47**, **2/3**, 447–459, 2004. Also Technical Report TR 2002 020, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2002. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[7] D. A. Bini, L. Gemignani, V. Y. Pan, Fast and Stable QR Eigenvalue Algorithms for Generalized Companion Matrices and Secular Equation, *Numerische Math.*, **3**, 373–408, 2005. (Also Technical Report 1470, *Department of Math., University of Pisa*, Pisa, Italy, July 2003.)

[8] D. A. Bini, B. Meini, Approximate Displacement Rank and Applications, in *AMS Conference "Structured Matrices in Operator Theory, Control, Signal and Image Processing"*, Boulder, 1999 (edited by V. Olshevsky), *American Math. Society*, 215–232, Providence, RI, 2001.

[9] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, J. Zhu, A Superfast Algorithm for Toeplitz Systems of linear Equations, *SIAM. J. on Matrix Analysis and Applications* **29**, **4**, 1247–1266, 2007.

[10] R.E. Cline, R.J. Plemmons, and G. Worm, Generalized Inverses of Certain Toeplitz Matrices, *Linear Algebra and Its Applications,* **8**, 25–33, 1974.

[11] D. Coppersmith, S. Winograd, Matrix Multiplicaton via Arithmetic Progressions. *J. of Symbolic Computation*, **9 3**, 251–280, 1990.

[12] J. D. Dixon, Estimating Extremal Eigenvalues and Condition Numbers of Matrices, *SIAM J. on Numerical Analysis*, **20, 4**, 812–814, 1983.

[13] J. Demmel, The Probability That a Numerical Analysis Problem Is Difficult, *Math. of Computation*, **50**, 449–480, 1988.

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

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

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

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

[18] K. R. Davidson, S. J. Szarek, Local Operator Teory, Random Matrices, and Banach Spaces, in *Handbook on the Geometry of Banach Spaces* (W. B. Johnson and J. Lindenstrauss editors), pages 317–368, North Holland, Amsterdam, 2001.

[19] A. Edelman, Eigenvalues and Condition Numbers of Random Matrices, PhD Thesis (106 pages), Math Dept., MIT, 1989 and *SIAM J. on Matrix Analysis and Applications*, **9**, **4**, 543–560, 1988.

[20] Y. Eidelman, I. Gohberg, On a New Class of Structured Matrices, *Integral Equations and Operator Theory*, **34**, 293–324, Birkhäuser, Basel, 1999.

[21] G. H. Golub, Some Modified Matrix Eigenvalue Problems, *SIAM Review*, **15**, 318–334, 1973.

[22] M. Gu, Stable and Efficient Algorithms for Structured Systems of Linear Equations, *SIAM J. on Matrix Analysis and Applications*, **19**, 279-306, 1998.

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

[24] I. Gohberg, T. Kailath, V. Olshevsky, Fast Gaussian Elimination with Partial Pivoting for Matrices with Displacement Structure, *Mathematics of Computation*, **64**, 1557–1576, 1995.

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

[26] G. Heinig, Inversion of Generalized Cauchy Matrices and Other Classes of Structured Matrices, in *Linear Algebra for Signal Processing, IMA Vol. Math. Appl.*, **69**, pp. 63-81, Springer, New York, 1995.

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

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

[29] P. Kravanja, M. Van Barel, Algorithms for Solving Rational Interpolation Problems Related to Fast and Superfast Solvers for Toeplitz Systems, *SPIE*, 359–370, 1999.

[30] 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. Available at http //crd.lbl.gov/x̃iaoye/XBLAS/.

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

[32] 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.

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

[34] M. Morf, Doubling Algorithms for Toeplitz and Related Equations, *Proceedings of IEEE International Conference on ASSP*, 954–959, IEEE Press, Piscataway, New Jersey, 1980.

[35] A. Melman, A Unifying Convergence Analysis of Second-Order Methods for Secular Equations, *Math. Comp.*, **66**, 333–344, 1997.

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

[37] 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.

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

[39] V. Olshevsky, I. V. Oseledets, E. E. Tyrtyshnikov, Tensor Properties of Multilevel Toeplitz and Related Matrices, *Linear Algebra and Its Applications*, **412**, 1–21, 2006.

[40] V. Olshevsky, I. V. Oseledets, E. E. Tyrtyshnikov, Superfast Inversion of Two-Level Toeplitz Matrices Using Newton Iteration and Tensor-Displacement Structure, *Operator Theory: Advances and Applications*, **179**, 229–240, 2008.

[41] I. V. Oceledetz, E. E. Tyrtyshnikov, Approximate Inversion of Matrices in the Process of Solving a Hypersingular Integral Equation, *Computational Math. and Math. Physics*, **45**, **2**, 302–313, 2005 (Translated from *JVM i MF*, **45**, **2**, 315–326, 2005).

[42] V. Y. Pan, On Schemes for the Evaluation of Products and Inverses of Matrices (in Russian), *Uspekhi Matematicheskikh Nauk*, **27**, **5 (167)**, 249-250, 1972.

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

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

[45] V. Y. Pan, Parallel Solution of Toeplitz-like Linear Systems, *J. of Complexity*, **8**, 1–21, 1992.

[46] V. Y. Pan, Concurrent Iterative Algorithm for Toepliz-like Linear Systems, *IEEE Transactions on Parallel and Distributed Systems*, **4, 5**, 592–600, 1993.

[47] V. Y. Pan, Decreasing the Displacement Rank of a Matrix, *SIAM Journal on Matrix Analysis and Applications*, **14**, **1**, 118–121, 1993.

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

[49] V. Y. Pan, Newton's Iteration for Matrix Inversion, Advances and Extensions, in MATRIX METHODS: THEORY, ALGORITHMS AND APPLICATIONS (Dedicated to the Memory of Gene Golub, edited by Vadim Olshevsky and Eugene Tyrtyshnikov), World Scientific Publishing, ISBN-(978-981-283-601-4)-(981-283-601-2), pages 364–381, 2008.

[50] F. Poloni, A Note on the O(n)-Storage Implementation of the GKO Algorithm, arXiv:0903.4569, Subjects: Numerical Analysis (math.NA), 2009.

[51] V. Y. Pan, S. Branham, R. Rosholt, A. Zheng, Newton's Iteration for Structured Matrices and Linear Systems of Equations, *SIAM volume on Fast Reliable Algorithms for Matrices with Structure* (edited by T. Kailath and A. H. Sayed), 189–210, SIAM Publications, Philadelphia, 1999.

[52] 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), **409**, **2**, 255–268, 2008.

[53] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Wang, X. Yan, Root-finding with Eigen-solving, Chapter 14, pages 219–245 in *Symbolic-Numeric Computation*, (Dongming Wang and Li-Hong Zhi, editors), Birkhäuser, Basel/Boston, 2007.

[54] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, accepted by *Linear Algebra and Its Applications*.

[55] 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.

[56] V. Y. Pan, M. Kunin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, W. Cao, Linking the TPR1, DPR1 and Arrow-head Matrix Structures, *Computers and Mathematics with Applications*, **52, 10–11**, 1603–1608, 2006.

[57] V. Y. Pan, B. Murphy, G. Qian, R. E. Rosholt, Error-free Computations via Floating-Point Operations, *Computers and Mathematics (with Applications)*, **57**, 560–564, 2009.

[58] V. Y. Pan, B. Murphy, R. E. Rosholt, Y. Tang, X. Wang, A. Zheng, Eigen-solving via Reduction to DPR1 matrices, *Computers and Mathematics with Applications*, **56**, 166–171, 2008.

[59] V. Y. Pan, G. Qian, On Solving Linear System with Randomized Augmentation, Tech. Report TR 2009009, *Ph.D. Program in Computer Science*, *Graduate Center, the City University of New York*, 2009.
Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[60] V. Y. Pan, G. Qian, A. Zheng, On Randomized Preprocessing versus Pivoting, Tech. Report TR 2009010, *Ph.D. Program in Computer Science*, *Graduate Center, the City University of New York*, 2009.
Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352.

[61] V. Y. Pan, G. Qian, A. Zheng, Null Space Computations and Eigen-solving with Randomized Preprocessing and Extensions, preprint, 2009.

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

[63] V. Y. Pan, Y. Rami, Newtons Iteration for the Inversion of Structured Matrices, in *Structured Matrices: Recent Developments in Theory and Computation*, (edited by D. Bini, E. Tyrtyshnikov and P. Yalamov), 79–90, Nova Science Publishers, USA, 2001.

[64] V. Y. Pan, Y. Rami, X. Wang, Structured Matrices and Newton's Iteration: Unified Approach, *Linear Algebra and Its Applications*, **343–344**, 233–265, 2002.

[65] V. Y. Pan, R. Schreiber, An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications, *SIAM Journal on Scientific and Statistical Computing*, **12, 5**, 1109–1131, 1991.

[66] V. Y. Pan, M. Van Barel, X. Wang, G. Codevico, Iterative Inversion of Structured Matrices, *Theoretical Computer Science*, **315**, **2–3** (Special Issue on Algebraic and Numerical Computing), 581–592, 2004.

[67] V. Y. Pan, X. Wang, Degeneration of Integer Matrices Modulo an Integer, *Linear Algebra and Its Applications*, **429**, 2113–2130, 2008.

[68] V. Y. Pan, X. Yan, Additive Preconditioning, Eigenspaces, and the Inverse Iteration, *Linear Algebra and Its Applications*, **430**, 186–203, 2009.

[69] V. Y. Pan, A. Zheng, On Root-finding with Eigen-solving and Preprocessing, preprint.

[70] G. Rodriguez. Fast solution of Toeplitz- and Cauchy-like least squares problems, *SIAM J. Matrix Analysis and Applications*, **28, 3**, 724–748, 2006.

[71] 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.

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

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

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

[75] A. Sankar, D. Spielman, S.-H. Teng, Smoothed Analysis of the Condition Numbers and Growth Factors of Matrices, *SIAM Journal on Matrix Analysis*, **28**, **2**, 446–476, 2006.

[76] 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.

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

[78] M. Van Barel, A Supefast Toeplitz Solver, 1999.
Available at http://www.cs.kuleuven.be/ marc/software/index.html

[79] M. Van Barel, G. Heinig, P. Kravanja, A Stabilized Superfast Solver for Nonsymmetric Toeplitz Systems, *SIAM Journal on Matrix Analysis and Applications*, **23**, **2**, 494–510, 2001.

[80] M. Van Barel, P. Kravanja, A Stabilized Superfast Solver for Indefinite Hankel Systems, *Linear Algebra and its Applications*, **284**, **1–3**, 335–355, 1998.

[81] R. Vandebril, *Semiseparable Matrices and the Symmetric Eigenvalue Problem*, PhD Thesis, Computer Science Dept., Katholieke Universiteit Leuven, Leuven, Belgium, 2004.

[82] R. Vandebril, M. Van Barel, G. Golub, N. Mastronardi, A Bibliography on Semiseparable Matrices, *Calcolo*, **42, 3–4,** 249–270, 2005.

[83] R. Vandebril, M. Van Barel, N. Mastronardi, *Matrix Computations and Semiseparable Matrices: Linear Systems* (Volume 1), The Johns Hopkins University Press, Baltimore, Maryland, 2007.

[84] R. Vandebril, M. Van Barel, N. Mastronardi, *Matrix Computations and Semiseparable Matrices: Eigenvalue and Singular Value Methods* (Volume 2), The Johns Hopkins University Press, Baltimore, Maryland, 2008.

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

[86] M. Wschebor, Smoothed Analysis of $\kappa(a)$, *J. of Complexity*, **20**, 97–107, 2004.

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

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