City University of New York (CUNY)
CUNY Academic Works

Computer Science Technical Reports | CUNY Academic Works

2010

TR-2010009: Solving Linear Systems with Randomized Augmentation II

Victor Y. Pan

Guoliang Qian

How does access to this work benefit you? Let us know!

More information about this work at: https://academicworks.cuny.edu/gc_cs_tr/345
Discover additional works at: https://academicworks.cuny.edu

This work is made publicly available by the City University of New York (CUNY).
Contact: AcademicWorks@cuny.edu

Solving Linear Systems with Randomized Augmentation II *

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

With a high probablilty our randomized augmentation of a matrix eliminates its rank deficiency and ill conditioning. Our techniques avoid various drawbacks of the customary algorithms based on pivoting and orthogonalization, e.g., we readily preserve matrix structure and sparseness. Furthermore our randomized augmentation is expected to precondition quite a general class of ill conditioned input matrices. As a result, we dramatically accelerate the solution of both singular and ill conditioned linear systems of equations in terms of the estimated arithmetic time and the observed CPU time. The progress has been extended to various other fundamental matrix computations.

Key words: Linear systems of equations, Randomized augmentation, Conditioning of random matrices, Toeplitz matrices

1 Introduction

1.1 Background: computations of vectors and bases in the null space

Solution of a homogeneous linear system of equations $A\mathbf{y} = \mathbf{0}$ is a fundamental problem and is closely linked to some other central subjects of matrix computations (see our Sections 3.4 and 7, [35, Sections 7.2 and 11.1], and [40]).

The solution vectors $\mathbf{y}$ are said to be the *null vectors* of the matrix A. They form the *null space* $\mathcal{N}(A) = \{\mathbf{y} : A\mathbf{y} = \mathbf{0}\}$. If its basis is given by the columns of a matrix B, then we call B a *null matrix basis* for a matrix A. Hereafter we refer to such a basis as a nmb(A).

The customary algorithms compute null vectors and nmbs of a matrix by employing its LU or QR factorization with pivoting or its SVD. These computations are quite costly, particularly the computation of the SVD, but even "pivoting usually degrades the performance" [19, page 119].

*Supported by PSC CUNY Awards 61406–0039 and 62230–0040. Some results of this paper have been presented at the Fifth International Computer Science Symposium in Russia (CSR 2010) in Kazan' [36] and at the 16th Conference of the International Linear Algebra Society (ILAS) in Pisa, Italy, both in June 2010.

1.2 Randomized northern augmentation

Our alternative recipe is *randomized augmentation* of the input matrix A, linked to additive preprocessing in [32, Section 12] and [35, Section 4].

Let us otline some basic techniques. At first assume an $m \times n$ matrix A having a rank $\rho < n$ (and thus having the nullity $r = n - \rho$) and generate a random $r \times n$ matrix V. Then we readily prove that with a probability close to one the northern augmentation $A \to C = \begin{pmatrix} V \\ A \end{pmatrix}$ produces an $(m + r) \times n$ matrix C that has full column rank and therefore has an $n \times (m + r)$ left inverse $C^{(I)}$, such that $C^{(I)}C = I_n$. It is also easy to prove that its $n \times r$ western block $C^{(I)} \begin{pmatrix} I_r \\ O \end{pmatrix}$ is a nmb(A).

Thus with our augmentation we expect to *regularize the nmb problem* for a matrix A (possibly rank deficient) by reducing the nmb task to solving r linear systems with the above matrix C of full column rank. We involve neither pivoting nor orthogonalization, preserve matrix structure and sparseness, and significantly accelerate the customary algorithms.

Table 9.1 displays the average CPU time in our extensive tests for the computation of null vectors of an $n \times n$ singular Toeplitz matrix as well as the observed average acceleration factor $f(n)$ versus the QR-based solution. In particular $f(512) > 15$, $f(1024) > 90$, and $f(2048) > 350$.

We also prove that our randomized regularization is expected to cause no sacrifice in the output accuracy. Namely assume standard Gaussian random matrix V and an input matrix A scaled so that the norm $||A||$ is neither large nor small in the context (we say *mildly scaled*). Then we prove that the ratio $\frac{\operatorname{cond} C}{\operatorname{cond} A}$ is expected to be neither large nor small, and we support our formal estimates with numerical experiments (see Tables 9.3 and 9.4).

Furthermore suppose our mildly scaled matrix A is well conditioned; then its small norm random perturbation is expected to produce an ill conditioned matrix $\tilde{A}$ of full rank having *numerical nullity* r and having an *approximate nmb* $\tilde{C}^{(I)} \begin{pmatrix} I_r \\ O \end{pmatrix}$ for $\tilde{C} = \begin{pmatrix} V \\ \tilde{A} \end{pmatrix}$. Furthermore the condition numbers $\operatorname{cond} \tilde{C}$ and $\operatorname{cond} C$ are expected to be of the order of $\operatorname{cond} A$ provided V is standard Gaussian random matrix.

1.3 Our topics and some by-products and extensions

In this paper we supply some details to the above outline and extend it further. In particular we cover the computation of the nullity and numerical nullity, extend our northern augmentation to enable *preconditioning* for quite general matrix classes, extend our techniques to solving nonsingular ill conditioned linear systems of equations, estimate the complexity of our algorithms in the cases of general, structured, and sparse input matrices, and describe our numerical experiments.

We outline some of our main techniques and results in the two next subsections.

Some by-products of our study can be of independent interest, e.g., the links between solving homogeneous and nonhomogeneous linear systems of equations and between augmentation and aggregation as well as our estimates for the condition numbers of partly randomized matrix products, which enables us to extend the *Smoothed Analysis* of conditioning of random matrices in [43] to employing them for preconditioning by means of randomized augmentation.

We refer the reader to the papers [32]–[40] on various further extensions and applications of our approach to fundamental matrix and polynomial computations.

1.4 Randomized western and northwestern augmentation

While our scaled randomized northern augmentation is expected to have regularization power and to create no numerical problems for the task of the nmb computation, its combination with our scaled randomized western augmentation $A \to (B, A)$ is expected to have both regularization and preconditioning power for the same task.

Assume that a random $m \times q$ matrix B has been appended as the leftmost block to an $m \times n$ matrix A (for $q < m \le n$). Then it is easy to prove that with a probability close to one

$\operatorname{rank}(B, A) = \min\{m, q + \operatorname{rank} A\}$, so that the resulting matrix (B, A) is expected to have full rank for $q \geq m - \operatorname{rank} A$.

Furthermore similarly to the northern augmentation assume a mildly scaled matrix A and standard Gaussian random matrix B. Then we prove that the condition number $\operatorname{cond}(B, A)$ is expected to be of the order $\frac{\sigma_1(A)}{\sigma_{m-q}(A)}$ versus $\operatorname{cond} A = \frac{\sigma_1(A)}{\sigma_m(A)}$ where $\sigma_j(A)$ denotes the jth largest singular value of the matrix A.

Therefore our western augmentation is expected to regularize the matrix A if $q + \operatorname{rank} A \geq m$ and to precondition it if it has a positive numerical nullity at most q or equivalently if it is ill conditioned, whereas the ratio $\frac{\sigma_1(A)}{\sigma_{m-q}(A)}$ is not large.

Such a regularization and preconditioning power is preserved when we combine our western and northern augmentations into northwestern augmentation $A \to K = \begin{pmatrix} W & V \\ B & A \end{pmatrix}$. Assume an $n \times n$ matrix A of a rank ρ and random matrices B, V, and W such that K is an $(n+r) \times (n+r)$ matrix. Then it is expected that $\operatorname{rank} K = \min\{n + r, \rho + 2r\}$, that is the augmentation is expected to produce a nonsingular matrix K if (and only if) $r \geq n - \rho$.

Furthermore if A is a mildly scaled matrix and B, V, and W are standard Gaussian random matrices, then $\operatorname{cond} K$ is expected to have the order $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$. Here we can even choose, say $V = -B$ to involve fewer random parameters (see Section 6.4).

Thus our augmentation is expected to regularize (resp. precondition) the matrix A if and only if the matrix A has a positive nullity (resp. numerical nullity) at most r.

These observations can be a basis for binary search for the nullity, but we also propose a competitive search by means of aggregation (cf. Section 3.3).

To use the benefits of our augmentation we must reduce our original computational task for an input matrix A to computations with the matrix K. We specify such reductions for nmb computation with western and northwestern augmentations in Algorithms 5.1 and 6.1, but next we apply northwestern augmentation to facilitate the inversion of a nonsingular ill conditioned matrix A as well as the solution of a linear system $A\mathbf{y} = \mathbf{b}$ with such a matrix A.

1.5 Northwestern augmentation for matrix inversion and solving linear systems of equations

Surely we can reduce the solution of such a linear system to computing a nonzero null vector of the matrix $(\theta\mathbf{b}, A)$, obtained by appending a scaled vector $\mathbf{b}$ as the new leftmost (western) column to the matrix A. On the average input pair of a matrix A and a vector $\mathbf{b}$, both mildly scaled, this map works as preconditioning (see Theorem 7.1). Randomized northwestern augmentation and the following theorem enable even more powerful preconditioning of a linear system $A\mathbf{y} = \mathbf{b}$.

Theorem 1.1. *Let $K = \begin{pmatrix} W & V \\ B & A \end{pmatrix}$ where A, W and K are nonsingular matrices of sizes $n \times n$, $r \times r$, and $(n + r) \times (n + r)$, respectively, for $0 < r < n$. Write $S = A - BW^{-1}V$ and $R = I + VBW^{-1}$. (S is the Schur complement of the block W in the matrix K.) Then*

(a) S^{-1} is the $n \times n$ trailing principal (that is southeastern) block of the matrix K^{-1} and

(b) $A^{-1} = S^{-1} - S^{-1}BW^{-1}R^{-1}VS^{-1}$.

(Part (a) is well known and is readily verified. Part (b) follows from the the Sherman–Morrison–Woodbury formula [19, page 50].)

The theorem reduces the inversion of the matrix A to the inversion of the matrices W, K, and R. Therefore our northwestern augmentation is expected to work as preconditioning where A is a mildly scaled matrix having a positive numerical nullity at most r and B, V, and W are standard Gaussian random matrices.

Namely, according to our analysis the matrix K is expected to have the condition number of the order $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$ and the matrices W and R are expected to be well conditioned. (Under further scaling such that $||B|| \approx ||V|| \approx ||W|| \approx ||A|| \ll 1$, we can expect that $R \approx I$.) Thus our augmentation

is expected to reduce the inversion of an ill conditioned matrix A to the inversion of three well conditioned matrices and to some matrix-by-vector multiplications.

Of course, numerical problems cannot simply disappear from computations with an ill conditioned matrix A, but our techniques confine them to the stage of iterative refinement of the inverse S^{-1}, which must be computed highly accurately. The refinement is reduced to a sequence of matrix-by-vector multiplications performed with a low (e.g., double) precision [32], [38].

In Section 7.4 we adapt the above technique, based on Theorem 1.1, to the solution of a nonsingular linear system $A\mathbf{y} = \mathbf{b}$ of n equations. In Section 8 we estimate that this approach accelerates the customary solution algorithms by the factor n/r. We can replace this factor by $\frac{n}{d \log n}$ where the matrices A and K have displacement structure of Toeplitz or Hankel type and are represented with displacement generators of a length d. (Recall that $d \leq 2$ for Toeplitz and Hankel matrices [31].) The estimates also show significant acceleration of the known algorithms for computations with sparse unstructred matrices as well as multilevel Toeplitz and Hankel matrices (see Section 8.4).

1.6 Organization of the paper

In the next section we first recall some definitions and basic facts and then estimate the ranks and condition numbers of random matrices and randomized matrix products. In Sections 3, 5 and 6 we cover our randomized augmentation techniques, and in Section 4 the related techniques of randomized post-multiplication. In Section 7 we cover our algorithms for a nonhomogeneous linear system of equations and link them to computing nmbs and null vectors. In Section 8 we estimate the computational cost of our randomized algorithms. In Section 9 we present the results of our numerical tests, the contribution of the second author of this paper.

Acknowledgement. Marc Van Barel's pointer to his Toeplitz solver in [45] was most helpful for our tests.

2 Definitions and basic facts

2.1 General matrices, nmbs and annihilators

We use and extend the customary definitions in [19] and [42].

$\mathbb{C}$ (resp. $\mathbb{R}$) is the field of complex (resp. real) numbers.

Flop is an arithmetic operation with such numbers.

A^T and A^H denote the transpose and the Hermitian transpose of an $m \times n$ matrix A, respectively. ($A^H = A^T$ for a real matrix A.)

A matrix A is Hermitian if $A = A^H$. A matrix $A = B^H B$ is Hermitian positive definite if B is a nonsingular matrix.

$(B_1, \dots, B_k) = (B_j)_{j=1}^k$ is a $1 \times k$ block matrix with blocks $B_1, \dots, B_k$.

$\mathrm{diag}(B_1, \dots, B_k) = \mathrm{diag}(B_j)_{j=1}^k$ is a $k \times k$ block diagonal matrix with diagonal blocks $B_1, \dots, B_k$.

I_n or just I denote the identity matrix $(\mathbf{e}_i)_{i=1}^n = (\mathbf{e}_1, \dots, \mathbf{e}_n)$.

O and $O_{k,l}$ denote the $k \times l$ matrix filled with zeros.

$O_{k,1}$, $\mathbf{0_k}$, and $\mathbf{0}$ denote the vector of a dimension k filled with zeros.

A matrix U is unitary or orthonormal if $U^H U = I$.

$Q = Q(A)$ is the Q-factor in the unique thin QR factorization $A = QR$ of a matrix A where the upper triangular matrix R is a square matrix with positive diagonal entries [19, Theorem 5.2.2].

A matrix has full row (resp. column) rank if its rows (resp. columns) are linearly independent.

$\mathcal{R}(A)$ denotes the range of the matrix A, that is the linear space generated by its columns, $\mathcal{N}(A)$ its null space $\{\mathbf{v} : A\mathbf{v} = \mathbf{0}\}$, $\rho = \mathrm{rank}\, A = \dim \mathcal{R}(A)$ its rank, and $\mathrm{nul}\, A = \dim \mathcal{N}(A)$ its nullity. $\mathbf{v}$ is its null vector if $A\mathbf{v} = \mathbf{0}$.

A matrix H is a *complete annihilator* of a matrix A if $\mathcal{R}(H) = \mathcal{N}(A)$. Such an annihilator is a *null matrix basis* if it has full column rank. We use the abbreviations nmb, ca, $\mathrm{nmb}(A)$, and $\mathrm{ca}(A)$.

Given a ca(A), we can compute a nmb(A) by applying its LUP or QR factorization or the following simple fact.

Fact 2.1. *[35]. Suppose H is a* ca(A). *Then*
(a) H is a nmb(A) *if and only if* $\operatorname{nul} H = 0$ *and*
(b) HY is a nmb(A) *if X is a* ca(H) *and if (X, Y) is a nonsingular matrix.*

2.2 SVD, inverses, norms, condition number, and numerical nullity

$A = S_A \Sigma_A T_A^H$ is *SVD* or *full SVD* of an $m \times n$ matrix A of a rank ρ if $S_A S_A^H = S_A^H S_A = I_m$, $T_A T_A^H = T_A^H T_A = I_n$, $\Sigma_A = \operatorname{diag}(\widehat{\Sigma}_A, O_{m-\rho,n-\rho})$, and $\widehat{\Sigma}_A = \operatorname{diag}(\sigma_j)_{j=1}^{\rho}$.

Here $\sigma_j = \sigma_j(A) = \sigma_j(A^H)$ is the jth largest singular value of a matrix A, $j = 1, \dots, \rho$, equal to the distance from this matrix to a nearest matrix of rank $j-1$ for $j = 1, \dots, \rho$, as this follows from the Courant–Fischer minimax theorem [19, Theorem 8.1.2].

$\Sigma_A^+ = \operatorname{diag}((\widehat{\Sigma}_A)^{-1}, O_{n-\rho,m-\rho})$ and $A^+ = T_A \Sigma_A^+ S_A^H$ are the Moore–Penrose generalized (or pseudo) inverses of the matrices Σ_A and A, respectively.

The matrix $X = A^{(I)}$ is a left (resp. right) inverse of a matrix A if $XA = I$ (resp. $AX = I$). $A^{(I)} = A^+$ for a matrix A of full rank, $A^{(I)} = A^+ = A^{-1}$ for a nonsingular matrix A.

$\sigma_1(A) = ||A|| = ||A^H||$ is the 2-norm of a matrix $A = (a_{i,j})_{i,j=1}^{m,n}$.

$||A||_F = \sqrt{\sum_{i,j=1}^{m,n} |a_{i,j}|^2}$ is its Frobenius norm.

We have $||A||/\sqrt{mn} \le \max_{i,j=1}^{m,n} |a_{i,j}| \le ||A||$, $||A|| \le ||A||_F \le \sqrt{n}||A||$.

Hereafter the concepts "large", "small", "nearby", "approximate", "ill conditioned" and "well conditioned" as well as our notation $\approx$, $\ll$, and $\gg$ are quantified in the context of the computational task and computer environment.

We say that a matrix A is *mildly scaled* if its norm $||A||$ is neither large nor small.

We write $a \ll b$ and $b \gg a$ if the ratio b/a is large and write $a \approx b$ if it is close to one or if $b = 0$ and $|a|$ is small. For two matrices A and B we write $A \approx B$ if $||A - B|| \ll ||A||$.

$\operatorname{cond} A = \sigma_1(A)/\sigma_\rho(A) = ||A||\, ||A^+||$ is the condition number of a matrix A of a rank ρ. Such a matrix is *ill conditioned* if $\sigma_1(A) \gg \sigma_\rho(A)$ and is *well conditioned* otherwise. See [7], [19, Sections 2.3.2, 2.3.3, 3.5.4, 12.5], [21, Chapter 15], and [42, Section 5.3] on effective estimation of norms and condition numbers.

We write $S_{A,r} = S_A(O_{m-r,r}, I_r)^T$, $\mathbb{S}_{A,r} = \mathcal{R}(S_{A,r})$, $T_{A,r} = T_A(O_{n-r,r}, I_r)^T$, and $\mathbb{T}_{A,r} = \mathcal{R}(T_{A,r})$, so that $\mathbb{S}_{A,r}$ and $\mathbb{T}_{A,r}$ denote the left and right trailing singular spaces associated with the r smallest singular values of the matrix A, respectively.

An $m \times n$ matrix A where $m \ge n$ has numerical nullity r and numerical rank $n - r$ if it has exactly r singular values that are small relative to its norm. In an equivalent alternative definition, numerical nullity (resp. numerical rank) of a matrix A is the maximal nullity (resp. minimal rank) of a nearby martix $\tilde{A}$.

By zeroing the r smallest singular values of such a matrix A we turn it into a well conditioned matrix $\tilde{A}$ that lies nearby and has rank $n - r$. Conversely, in virtue of Corollary 2.2 in Section 2.4 a small norm random perturbation E of an $n \times n$ well conditioned matrix $\tilde{A}$ of a rank $n - r$ (for $0 < r < n$) is likely to produce a nonsingular ill conditioned matrix $A = \tilde{A} + E$ that has numerical nullity r.

The linear space $\mathbb{T}_{A,r}$ approximates the null space of the matrix $\tilde{A}$, and likewise the linear space $\mathbb{S}_{A,r}$ approximates the null space of the matrix $\tilde{A}^H$. Hereafter a matrix B is said to be an *approximate nmb* (resp. an *approximate ca*) of a matrix A if $B \approx \operatorname{nmb}(\tilde{A})$ (resp. if $B \approx \operatorname{ca}(\tilde{A})$).

We employ an approximate nmb in the following simple theorem to approximate the matrix A by a matrix of a smaller rank.

Theorem 2.1. *(See [35, Section 7.2].) Suppose an $n \times n$ matrix A has a positive numerical nullity r and suppose B is an $n \times r$ matrix such that the $r \times r$ matrix $B^H B$ is nonsingular and $||AB|| \ll ||A||\, ||B||$. Then the matrix $\widehat{A} = A(I - B(B^H B)^{-1} B^H)$ closely approximates the matrix A, has rank $n - r$, and turns into the matrix $A(I - BB^H)$ if B is a unitary matrix.*

Remark 2.1. *Unlike the nullity and the rank, numerical nullity and numerical rank are not well defined for a large class of ill conditioned matrices, in particular for all matrices A having nested clusters of small singular values but also for the matrix class exemplified by for a 1000×1000 matrix A with singular values $\sigma_j(A) = 10^{1000-j}$, $j = 1, 2, \dots, 1000$.*

Theorem 2.2. *[37]. 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 $\operatorname{cond}(AB) \leq (\operatorname{cond} A) \operatorname{cond} B$.*

2.3 Structured matrices

$J = J_n = (\mathbf{e}_1, \dots, \mathbf{e}_n) = (j_{i,k})_{i,k=0}^{n-1}$ is $n \times n$ the reflection matrix, $j_{i,k} = 1$ if $i + k = n - 1$, $j_{i,k} = 0$ unless $i + k = n - 1$. ($J^2 = I$.)

An $m \times n$ Toeplitz matrix $T = (t_{i-j})_{i=0,j=0}^{m-1,n-1}$ (resp. Hankel matrix $H = (h_{i+j})_{i=0,j=0}^{m-1,n-1}$) is defined by its $m + n - 1$ entries, e.g., by its first row and first (resp. last) column. TJ and JT are Hankel matrices for a Toeplitz matrix T, whereas HJ and JH are Toeplitz matrices for a Hankel matrix H.

$Z(\mathbf{v})$ denotes the lower triangular Toeplitz matrix defined by its first column vector $\mathbf{v} = Z(\mathbf{v})\mathbf{e}_1$. $Z(\mathbf{v}) = (Z(\mathbf{v}))^T$ denotes its transpose.

The Gohberg–Semencul formula expresses the inverse of a nonsingular $n \times n$ Toeplitz matrix T via its first column $T^{-1}\mathbf{e}_1$ and its last column $T^{-1}\mathbf{e}_n$ provided $\mathbf{e}_1^T T^{-1}\mathbf{e}_1 \neq 0$ (see [18], [44]). The latter provision has been relaxed in [20] and [17]. The following theorem (in [18] and [3, Theorem 7]) expresses the inverse via two columns (the first $\mathbf{v}$ and the last $\mathbf{w}$) of the inverse of an $(n+1) \times (n+1)$ Toeplitz matrix with the $n \times n$ leading principal (that is northwestern) block T.

Theorem 2.3. *Suppose $K = (t_{i,j})_{i,j=0}^{n}$ is a nonsingular $(n + 1) \times (n + 1)$ Toeplitz matrix, write $T = (t_{i,j})_{i,j=0}^{n-1}$, $\widehat{\mathbf{v}} = (v_i)_{i=0}^{n} = K^{-1}\mathbf{e}_1$, $\mathbf{v} = (v_i)_{i=0}^{n-1}$, $\mathbf{v}' = (v_i)_{i=1}^{n}$, $\widehat{\mathbf{w}} = (w_i)_{i=0}^{n} = K^{-1}\mathbf{e}_{n+1}$, $\mathbf{w} = (w_i)_{i=0}^{n-1}$, and $\mathbf{w}' = (w_i)_{i=1}^{n}$, and assume that $v_0 \neq 0$. Then the matrix $T = (t_{i,j})_{i,j=0}^{n-1}$ is nonsingular and $v_0 T^{-1} = Z(\mathbf{v})Z^T(J\mathbf{w}) - Z(\mathbf{w}')Z^T(J\mathbf{v}')$.*

Remark 2.2. *For any fixed positive integer q we can embed a nonsingular $n \times n$ Toeplitz matrix T into an $(n + q) \times (n + q)$ Toeplitz matrix K_q that has the $n \times n$ leading principal block T. Then we can recursively apply Theorem 2.3 to express the inverse T^{-1} via the two column vectors $K_q^{-1}\mathbf{e}_1$ and $K_q^{-1}\mathbf{e}_{n+q}$.*

A more general class of structured matrices having small displacement ranks d extends the classes of Toeplitz and Hankel matrices, for which $d \leq 2$. Such an $m \times n$ matrix can be represented by $(m + n)d$ parameters and can be multiplied and inverted fast provided its displacement rank d is small [4], [17], [22], [24], [31].

2.4 Random sampling and random matrices

$|\Delta|$ is the cardinality of a set Δ. *Random sampling* of elements from a set Δ is their selection from this set at random and independently of each other. A matrix is *random* if its entries are randomly sampled from a fixed set Δ. Random sampling is *uniform* if it is done under the uniform probability distribution on the set Δ.

Recall that the total degree of a polynomial in m variables is the sum of its degrees in each variable.

Lemma 2.1. *[10], [41], [49]. For a set Δ of cardinality $|\Delta|$ (in a fixed ring or field, e.g., in $\mathbb{C}$) let a polynomial in m variables have a total degree d, and let it not vanish identically on this set. Then the polynomial vanishes in at most $d|\Delta|^{m-1}$ points.*

Lemma 2.1 implies that a fixed nonvanishing polynomial vanishes with a probablity converging to zero if the values of its variables are sampled under any reasonable probability distribution on

the set Δ whose cardinality converges to the infinity. Under the uniform probability distribution the probability that the polynomial vanishes is estimated most readily.

Corollary 2.1. *Under the assumptions of Lemma 2.1 let the values of the variables of the polynomial be randomly and uniformly sampled from the set Δ. Then the polynomial vanishes with a probability of at most $\frac{d}{|\Delta|}$.*

Corollary 2.2. *Let A be an $m \times n$ matrix with random entries uniformly sampled from a finite set Δ of cardinality $|\Delta|$. Let $l = \min\{m, n\}$ and let M be any fixed $m \times n$ matrix. Then any $k \times k$ submatrix of the matrix $A + M$ for $k \le l$ is singular with a probability at most $k/|\Delta|$.*

Proof. The claimed bound holds for generic matrices. The singularity of a $k \times k$ matrix means that its determinant vanishes, but the determinant is a polynomial of degree k in the entries. □

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).

Definition 2.1. *$F_X(y) = \text{Probability}\{X \le y\}$ 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 $l = \min\{m, n\}$. $F_{\mu,\sigma}(y) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{y} \exp(-\frac{(x-\mu)^2}{2\sigma^2})dx$ is the CDF for a Gaussian random variable with a mean μ and a variance σ^2. $\Phi_{\mu,\sigma}(y) = F_{\mu,\sigma}(\mu + y) - F_{\mu,\sigma}(\mu - y)$ for $y \ge 0$.*

2.5 The extreme singular values of randomized matrix products

A standard Gaussian random matrix A (cf. Definition 2.1) is well conditioned with a high probability [8], [12]. Even perturbations by such a matrix is expected to turn any mildly scaled matrix M into a well conditioned matrix. We specify the respective estimates in Theorem 2.4, taken from [43] and cited in our Section 8. This theorem is also used in the proof of Theorem 2.7 from [38], which is extensively used in our present paper. Theorem 2.6 is not used and is included just to introduce Theorem 2.7 and to provide some insight into the subject.

Theorem 2.4. *(See [11, Theorem II.7].) Suppose $A \in \mathbb{R}^{n\times n}$ is a Gaussian random matrix with mean zero and a variance σ^2. Then $F_{||A||}(y) \ge 1 - \exp(-x^2/2)$ for $x = y/\sigma - 2\sqrt{n} \ge 0$.*

Theorem 2.5. *(See [43, 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, $W = \bar{U}(A + M)\bar{V}$, $l = \min\{m, n\}$, and $y \ge 0$. Then $F_W(y) \le 2.35\, y\sqrt{l}/\sigma$.*

Corollary 2.3. *(See [43, Theorem 3.1].) Under the assumptions of Theorem 2.5, let $||M|| \le \sqrt{l}$. Then $F_{\text{cond}(W)}(y) \ge 1 - (14.1 + 4.7\sqrt{(2\ln y)/n})n/(y\sigma)$ for all $y \ge 1$.*

This bound has been improved by the factor $\sqrt{\log n}$ in [48].

Theorem 2.6. *[37]. Suppose $G \in \mathbb{R}^{q\times m}$, $H \in \mathbb{R}^{n\times r}$, and a standard Gaussian random matrix $W \in \mathbb{R}^{m\times n}$ 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 $r_H = n$.*

The value $\text{cond}(AB)$ can be arbitrarily large even for $m \times r$ unitary matrices A and B^H if $m > r$, and so we cannot merely drop the above assumptions that $r_G = m$ and $r_H = n$, but the following theorem enables us to circumvent this problem.

Theorem 2.7. *[37]. Suppose $G \in \mathbb{R}^{r_G\times m}$, $H \in \mathbb{R}^{n\times r_H}$, $W \in \mathbb{R}^{m\times n}$, $\operatorname{rank} G = r_G < m$, $\operatorname{rank} H = r_H < n$, and the assumptions of Theorem 2.4 hold. Then (a) $F_{GW}(y) \le 2.35y\sqrt{l}/(\sigma_{r_G}(G)\sigma)$ and (b) $F_{WH}(y) \le 2.35y\sqrt{l}/(\sigma_{r_H}(H)\sigma)$.*

3 A nmb of a matrix via randomized northern augmentation

Our first algorithm begins with appending new scaled Gaussian random rows at the top of an $m \times n$ matrix A of a rank $\rho < n$ such that the matrix $C = \begin{pmatrix} V \\ A \end{pmatrix}$ is expected to have full column rank (cf. Theorem 3.1). If it does have it, then the algorithm computes the first $r = n - \rho$ columns of the left inverse $C^{(I)}$ and outputs them as a nmb(A). The matrix C is also expected to have condition number of the order of cond A, so that our randomized algorithm is expected to solve the rank nmb problem without creating new numerical problems.

We refer the reader to Sections 3.4 and 6.3 and Remark 5.3 on our treatment of ill conditioned inputs and to Section 4 on an alternative approach in the case where the input matrix A has full rank m.

One can similarly employ the southern augmentation $A \to \begin{pmatrix} A \\ V \end{pmatrix} = \begin{pmatrix} O & I_m \\ I_r & O \end{pmatrix} \begin{pmatrix} V \\ A \end{pmatrix}$.

3.1 Randomized northern augmentation: an algorithm

Algorithm 3.1. A nmb via randomized northern augmentation.

INPUT: *Three positive integers ρ, m, and n, an $m \times n$ matrix A of a rank ρ, and a random number generator.*

OUTPUT: *FAILURE or a matrix $B =$* nmb(A).

COMPUTATIONS:

1. *Write $r = n - \rho$. If $r = 0$, output the empty* nmb(A). *Otherwise generate a $r \times n$ random matrix V.*
2. *Output FAILURE and stop if the matrix C is column rank deficient.*
3. *Otherwise compute and output the block submatrix $B = C^{(I)} \begin{pmatrix} I_r \\ O \end{pmatrix}$, formed by the first r columns of a left inverse $C^{(I)}$ of the matrix $C = \begin{pmatrix} V \\ A \end{pmatrix}$ (cf. [19, Section 5.7]); then stop.*

Correctness proof. Let $Y =$ nmb$(A) \in \mathbb{C}^{n \times r}$ and write $B = C^{(I)} \begin{pmatrix} I_r \\ O \end{pmatrix}$. Then $CY = \begin{pmatrix} VY \\ O \end{pmatrix}$, $Y = C^{(I)} \begin{pmatrix} VY \\ O \end{pmatrix} = C^{(I)} \begin{pmatrix} I_r \\ O \end{pmatrix} VY$, and so

$$\mathcal{N}(A) = \mathcal{R}(Y) \subseteq \mathcal{R}(B). \tag{3.1}$$

It follows that $\mathcal{R}(B) = \mathcal{N}(A)$ because $\dim(\mathcal{R}(B)) = \operatorname{rank} B = r = \dim(\mathcal{N}(A))$.

3.2 Regularization power and conditioning of northern augmentation

Both our next theorem and Corollary 3.2 show that our randomized augmentation is likely to fix the input degeneracy, so that Algorithm 3.1 is unlikely to fail.

Theorem 3.1. *Suppose a random number generator in Algorithm 3.1 generates the matrices U and V by uniformly sampling their entries from a set $\Delta \in \mathbb{C}$ of a cardinality $|\Delta|$. Then the matrix C has full column rank n with a probability at least $1 - r/|\Delta|$.*

Proof. Let a $\rho \times n$ submatrix $A_{\rho,n}$ of the matrix A have full rank ρ and let $C_{n,n} = \begin{pmatrix} V \\ A_{\rho,n} \end{pmatrix}$. Clearly, $\det C_{n,n}$ is a polynomial of a degree at most r in the entries of the matrix V. It does not vanish identically in these entries (because the matrix $A_{\rho,n}$ has full rank), and so for random matrix V it vanishes with a probability at most $r/|\Delta|$ in virtue of Corollary 2.1. □

Under the assumptions of Theorem 3.1 let the set Δ have a large cardinality. Then Theorem 3.1 implies that the matrix C computed in Algorithm 3.1 is likely to have full rank, that is the randomized augmentation $A \to C$ is likely to yield regularization.

In the rest of this subsection we prove that the condition number $\operatorname{cond} C$ tends to be of the order $\frac{\sigma_1(A)}{\sigma_\rho(A)} = \operatorname{cond} A$ where V is Gaussian random matrix with the mean zero and a variance of the order $||A||^2$, e.g., where the matrix A is mildly scaled, whereas V is standard Gaussian random matrix.

Theorem 3.2. *Suppose that* $A \in \mathbb{C}^{m\times n}$, $V \in \mathbb{C}^{r\times n}$, $C = \begin{pmatrix} V \\ A \end{pmatrix}$, $\operatorname{rank} C = n$, $\operatorname{rank} A = n - r$, *and* $\operatorname{rank} V = r$. *Let* $A = S_A \Sigma_A T_A^H$ *be full SVD of the matrix* A *(where* $\Sigma_A = \operatorname{diag}(\widehat{\Sigma}_A, O)$ *and* $\widehat{\Sigma}_A$ *is a* $\rho \times \rho$ *diagonal matrix of the singular values). Write*

$$\operatorname{diag}(I_r, S_A^H) C T_A = \begin{pmatrix} M \\ O \end{pmatrix}, \ M = \begin{pmatrix} V_0 & V_1 \\ \widehat{\Sigma}_A & O \end{pmatrix}. \tag{3.2}$$

Then $\operatorname{cond} C \le (\frac{1}{\sigma_\rho(A)} + \frac{1}{\sigma_r(V_1)} + \frac{||V_0||}{\sigma_\rho(A)\sigma_r(V_1)})||C||$.

Proof. $||M^{-1}|| \le ||\widehat{\Sigma}_A^{-1}|| + ||V_1^{-1}|| + ||\widehat{\Sigma}_A^{-1}||\ ||V_1^{-1}||\ ||V_0||$ because $M^{-1} = \begin{pmatrix} O & \widehat{\Sigma}_A^{-1} \\ V_1^{-1} & -V_1^{-1} V_0 \widehat{\Sigma}_A^{-1} \end{pmatrix}$. Substitute $||\widehat{\Sigma}_A^{-1}|| = \frac{1}{\sigma_\rho(A)}$, $||V_1^{-1}|| = \frac{1}{\sigma_r(V_1)}$, $||M^{-1}|| = \frac{1}{\sigma_n(M)} = \frac{1}{\sigma_n(C)}$, and $\operatorname{cond} C = \frac{||C||}{\sigma_n(C)}$ and obtain the theorem. □

Corollary 3.1. *Under the assumptions of Theorem 3.2 suppose the matrices* A *and* V *have been scaled so that* $||A|| = ||V||$ *and write* $\kappa = \operatorname{cond} A$ *and* $\kappa_1 = \frac{||V||}{\sigma_r(V_1)}$. *Then* $\operatorname{cond} C \le \sqrt{2}(\kappa + \kappa_1 + \kappa\kappa_1)$.

Proof. We have $||C|| \le \sqrt{||A||^2 + ||V||^2} = \sqrt{2}||A|| = \sqrt{2}||V||$ because $||A|| = ||V||$. Moreover $||V_0|| \le ||VT_A|| = ||V||$ for $(V_0, V_1) = VT_A$. Substitute these bounds into Theorem 3.2. □

In the following theorem we assume that V is Gaussian random matrix and probabilistically estimate the value $\sigma_r(V_1)$ from below.

Theorem 3.3. *Under the assumptions of Theorem 3.2 suppose* $V \in \mathbb{R}^{r\times n}$ *is a Gaussian random matrix with mean zero and a variance* σ^2. *Then* $F_{V_1}(y) \le 2.35y\sqrt{r}/\sigma$.

Proof. Apply part (b) of Theorem 2.7 for $W = V$, $H = T_A \begin{pmatrix} O \\ I_r \end{pmatrix}$, $l = r_H = r$, $V_1 = WH$, and $\sigma_{r_H}(H) = 1$. □

Corollary 3.2. *Under the assumptions of Theorem 3.2 the matrix* C *is column rank deficient with the probability zero.*

Proof. Theorem 3.3 implies that the matrix V_1 is singular with the probability zero. Therefore the corollary follows from equation (3.2). □

3.3 The choice of the augmentation size via binary search or aggregation

In Algorithm 3.1 we assume that the rank ρ and the nullity $r = n - \rho$ are given to us, but otherwise we can compute r as the smallest integer for which the matrix C is expected to have full column rank n.

We can find this integer by testing the existence of a left inverse $C^{(I)}$ for various candidate values r. If the test fails, we would increment the integer parameter r. Otherwise we would either output it if the integer $r - 1$ has already failed the test or decrement this parameter.

To succeed with fewer tests we can apply binary search (where instead of minimizing the integer r that passes the test we can alternatively request that $AB = O$) or the following option.

Subalgorithm 3.1. *As soon as the current integer $r = q$ has passed the test, compute a matrix*

$$X = \mathrm{ca}(AB^{(q)}) \tag{3.3}$$

for $B^{(q)} = C^{(I)} \begin{pmatrix} I_q \\ O \end{pmatrix}$. *(If* $AB^{(q)} = O$, *then* $X = I$ *and* $B^{(q)}$ *is a* $\mathrm{nmb}(A)$.*) Compute* $\mathrm{ca}(A) = B^{(q)}X$ *and then a* $\mathrm{nmb}(A)$, *e.g., based on Fact 2.1.*

One can restrict this technique to computing just the nullity $\mathrm{nul}\, A = \mathrm{nul}\, B^{(q)}$ and then obtain a $\mathrm{nmb}(A)$ by applying Algorithm 3.1.

Theorem 3.4. *The matrix* $B^{(q)}X$ *computed in Subalgorithm 3.1 is a* $\mathrm{ca}(A)$.

Proof. First deduce from equation (3.3) that $AB^{(q)}X = O$, that is, $\mathcal{N}(A) \supseteq \mathcal{R}(B^{(q)}X)$. It remains to prove that $\mathcal{N}(A) \subseteq \mathcal{R}(B^{(q)}X)$, that is, $\mathbf{y} = B^{(q)}X\mathbf{w}$ for some vector $\mathbf{w}$ as soon as $A\mathbf{y} = \mathbf{0}$ for some vector $\mathbf{y}$. Clearly inclusion (3.1) holds for $B = B^{(q)}$ since the integer q has passed the test. This implies that $\mathbf{y} = B^{(q)}\mathbf{z}$ for some vector $\mathbf{z}$. Now it remains to prove that $\mathbf{z} = X\mathbf{w}$ for some vector $\mathbf{w}$. The latter equation follows because $A\mathbf{y} = AB^{(q)}\mathbf{z} = \mathbf{0}$ and because $X = \mathrm{ca}(AB^{(q)})$ by assumption. □

Searching for a $\mathrm{ca}(AB^{(q)})$ is simpler than for a $\mathrm{ca}(A)$ if $q \ll m$ because of the decrease of the input size from $m \times n$ to $n \times q$.

The above reduction of the input size exemplifies *aggregation methods*, which successively perform the following stages.

(a) Aggregate an input matrix M into a matrix M_1 of a smaller size. ($M = A$ and $M_1 = AB^{(q)}$ in the above example.)

(b) Compute the solution Z_1 for a given task, but for the aggregated input M_1. (Assuming the task of the computation of a ca, we have $Z_1 = X$.) At this stage, one can recursively reapply aggregation.

(c) Disaggregate the aggregated solution Z_1 to produce the solution Z for the original input M. (In the case of computing a nmb, we have $Z = B^{(q)}X$.)

Fact 2.1 defines aggregation of a matrix A into its complete annihilator $H = \mathrm{ca}(A)$ for the task of computing a nmb of a matrix A. Two examples of aggregation are also given in Sections 5 and 6 (cf. Remarks 5.2 and 6.1 and the Schur Aggregation in [32]). Among many other examples we recall the hierarchial aggregation processes in [27], which in the 1980s served as the springboard for Algebraic Multigrid, and trilinear aggregating in [23], [26], [29], [30], and [9], which is the basis of the fastest known algorithms for $n \times n$ matrix multiplication for $20 \le n \le 2^{20}$, is an important ingredient of the algorithm in [6] that supports the record exponent $\omega < 2.376$ for the complexity of matrix multiplication, and gave the first example where a nontrivial *tensor decomposition* enabled acceleration of some fundamental matrix computations.

3.4 Computing approximate nmbs

In numerical implementation of Algorithm 3.1, the input set consists of three integers m, n, and ρ, an $m \times n$ matrix A that has a numerical rank ρ, and a random number generator that generates a Gaussian random matrix V with the mean zero and a variance of the order of $||A||^2$ (e.g., a standard Gaussain random matrix V where the matrix A is mildly scaled). Then the algorithm computes a numerical $\mathrm{nmb}(A)$ by modifying Stage 2 of the algorithm as follows:

2. Output FAILURE and stop if the matrix C is rank deficient or ill conditioned.

Hereafter we refer to this algorithm as **Algorithm 3.1(num)**.

Suppose $\operatorname{rank}\tilde{A} = \rho$ and $\tilde{A} \approx A$, so that $C = \begin{pmatrix} V \\ A \end{pmatrix} \approx \tilde{C} = \begin{pmatrix} V \\ \tilde{A} \end{pmatrix}$, $B = C^{(I)} \begin{pmatrix} I_r \\ O \end{pmatrix} \approx \tilde{B} = \tilde{C}^{(I)} \begin{pmatrix} I_r \\ O \end{pmatrix}$, $\tilde{B} - B = (\tilde{C}^{(I)} - C^{(I)}) \begin{pmatrix} I_r \\ 0 \end{pmatrix}$, and therefore $||\tilde{B} - B|| \le ||\tilde{C}^{(I)} - C^{(I)}||$.

We can set $C^{(I)} = C^+$, $\tilde{C}^{(I)} = \tilde{C}^+$ and obtain that $||\tilde{B} - B|| \le ||\tilde{C}^+ - C^+|| \le 2||\tilde{C} - C||_F \max\{||C^+||^2, ||\tilde{C}^+||^2\}$ (see [19, Section 5.5.5]) and consequently $\operatorname{cond} C \approx \operatorname{cond} \tilde{C}$. Furthermore these two condition numbers are likely to have the order of $\operatorname{cond}\tilde{A} = \frac{\sigma_1(\tilde{A})}{\sigma_\rho(\tilde{A})} \approx \frac{\sigma_1(A)}{\sigma_\rho(A)}$ provided V is a Gaussian random matrix with the mean zero and a variance of the order of $||A||^2$ (cf. Section 3.2).

By extending our recipes from the previous subsection we can extend Algorithm 3.1(num) to the case where the input matrix A has an unknown positive numerical nullity (cf. Remark 2.1).

4 Nmbs of a matrix of full row rank via post-multiplication

In this section and in the next one we seek nmbs of $m \times n$ input matrices A having at least as many columns as rows. For $m > n$, we can apply the alternative techniques from our Sections 3 and 6 or from [35] or can shift the study to the case $m \le n$ based on the equations $\mathcal{N}(A) = \mathcal{N}(A^H A)$ or $\mathcal{N}(A) = \cap_{i=1}^h \mathcal{N}(B_i)$ where $A = \sum_{i=1}^h (O, B_i, O)^T$, B_i are $k_i \times n$ matrices for $i = 1, \dots, h$, and $\sum_{i=1}^h k_i \ge m$. [19, Theorem 12.4.1] enables us to simplify the computation of a nmb(A) from nmbs of the h matrices $(O, B_i, O)^T$ for $i = 1, \dots, h$.

Now suppose $A = (A_w, A_e)$ is an $m \times n$ matrix, $m \le n$, and the $m \times m$ western block A_w is nonsingular. Then we can immediately compute a nmb$(A) = \begin{pmatrix} -A_w^{-1} A_e \\ I_{n-m} \end{pmatrix}$. If, however, the block A_w is singular, but the matrix A has full rank m, we first choose an appropriate $n \times m$ matrix S that enables us to obtain a nonsingular $m \times m$ matrix AS; then we readily compute a desired nmb(A). The algorithm is restricted to input matrices of full rank, and within this input class saves some flops and memory space versus Algorithm 3.1.

Algorithm 4.1. A nmb via post-multiplication.

INPUT *and* OUTPUT *as in Algorithm 3.1, except that we have an input matrix $A \in \mathbb{C}^{m \times n}$ of full rank $m \le n$, rather than just any matrix A in $\mathbb{C}^{m \times n}$ and have an option of using no random number generator.*

COMPUTATIONS:

1. *Generate a nonsingular $n \times n$ matrix $W = (S, T)$ where $S \in \mathbb{C}^{n \times m}$.*
2. *Compute the $m \times m$ matrix AS.*
3. *Compute and output the matrix $W \begin{pmatrix} -(AS)^{-1} AT \\ I_{n-m} \end{pmatrix}$, a* nmb$(A)$. *This computation fails only if the matrix AS is singular. In this case output FAILURE and stop.*

Here are some relevant choices of the matrix S.

Theorem 4.1. *[19]. Let the matrix A in Algorithm 4.1 have full rank and write $S = A^H$. Then the matrix AS is nonsingular and* $\operatorname{cond}(AS) = (\operatorname{cond} A)^2$.

Theorem 4.2. *Assume that $m \le n$, an $m \times n$ matrix A has full rank, and S is an $n \times m$ Toeplitz (resp. general) matrix with $m + n - 1$ (resp. mn) random entries uniformly sampled from a finite set $\Delta \in \mathbb{C}$ of cardinality $|\Delta|$. Then the matrix AS is nonsingular (in which case the matrix S has full rank m) with a probability at least $1 - m/|\Delta|$.*

Proof. $\det(AS)$ is a polynomial of a degree at most m in the entries of the matrix S. The polynomial does not vanish identically in these entries because the matrix A has full rank. Now the theorem follows from Corollary 2.1. □

The theorem implies that Algorithm 4.1 is unlikely to fail. In numerical implementation we would stop and output FAILURE if the matrix AS is ill conditioned. Let us present some relevant estimates.

Theorem 4.3. *Assume that an $m \times n$ matrix A has full rank (so that $m \le n$) and that the $m \times m$ matrix AS is nonsingular. Then $\sigma_m(AS) \ge \sigma_m(A)\sigma_m(S)$ and $\mathrm{cond}(AS) \le (\mathrm{cond}\, A)\,\mathrm{cond}\, S$.*

Proof. The theorem follows from Theorem 2.2. □

We can scale the matrix S to have its norm within a desired range. Next we probabilistically estimate the value $\sigma_m(AS)$ from below.

Theorem 4.4. *Under the assumptions of Theorem 4.3 suppose $S \in \mathbb{R}^{n\times n}$ is a Gaussian random matrix with mean zero and a variance σ^2. Then $F_{AS}(y) \le cy\sqrt{m}/(\sigma_m(A)\sigma)$.*

Proof. The theorem follows from part (a) of Theorem 2.7 for G replaced by A, W by S, m by n, and r_G and l by m. □

Corollary 4.1. *Under the assumptions of Theorem 4.4 the matrix AS is singular with the probability zero.*

5 Rrandomized western augmentation

Our next idea is to append q scaled random columns on the left of a matrix $A \in \mathbb{C}^{m\times n}$ of a rank ρ such that $0 < \rho \le m \le n$, $m - \rho \le q < m$. As we prove later in this section, such a randomized western augmentation is expected to turn the matrix A into a matrix (B, A) of full rank. Furthermore, if V is a Gaussain random matrix with the mean zero and a variance of the order of $||A||^2$, then we can expect to have the condition number $\mathrm{cond}\, A = \frac{\sigma_1(A)}{\sigma_\rho(A)}$ at the level of $\mathrm{cond}(B, A) = \frac{\sigma_1(A)}{\sigma_{m-q}(A)}$. In other words our augmentation is likely to yield both regularization and (for $q > m - \rho$) preconditioning. The respective algorithm works as an aggregation process.

One can similarly employ the eastern augmentation $A \to (A, B) = (B, A)\begin{pmatrix} O & I_q \\ I_n & O \end{pmatrix}$.

5.1 Cas and nmbs via randomized western augmentation: an algorithm

Algorithm 5.1. A ca via randomized western augmentation.

INPUT: *Three positive integers m, n, and q and a matrix $A \in \mathbb{C}^{m\times n}$ such that $n \ge m$ and $q \ge r = \mathrm{nul}\, A^H$, a random number generator, and a randomized Subroutine* CA *(e.g., Algorithm 3.1 or 4.1) that either computes a ca of its input matrix or outputs FAILURE. (We assume that the subroutine definitely outputs FAILURE if its input matrix is rank deficient, but only with a low probability otherwise.)*

OUTPUT: *FAILURE or the nullity $r = \mathrm{nul}\, A^H$ and $\mathrm{ca}(A)$.*

COMPUTATIONS:

1. *(*Western augmentation.*) Generate a random $m \times q$ matrix B and apply the Subroutine* CA *to the matrix (B, A). If the subroutine fails, conclude that the matrix (B, A) is probably rank deficient, output FAILURE and stop.*
2. *(*Aggregation.*) Otherwise the subroutine computes a matrix $Z = \begin{pmatrix} Z_0 \\ Z_1 \end{pmatrix} = \mathrm{ca}(B, A)$ where $Z_0 \in \mathbb{C}^{q\times s}$, $Z_1 \in \mathbb{C}^{n\times s}$, and $q \le s \le q + r$. Compute and output the nullity $r = \mathrm{nul}\, Z_0 = \mathrm{nul}\, A^H$.*
3. *Then apply Algorithm 5.1 to the matrix Z_0 to compute an $s \times r$ matrix $X = \mathrm{ca}(Z_0)$.*
4. *(*Disaggregation.*) Compute and output the $n \times r$ matrix $Y = Z_1X = \mathrm{ca}(A)$.*

Correctness proof (cf. also Theorems 6.1 and 6.2 and Corollary 6.1). By the definition of the matrices Z and X, we have $BZ_0 + AZ_1 = O$ and $BZ_0X = O$. Therefore $AY = AZ_1X = O$. Conversely, if $A\widehat{\mathbf{y}} = \mathbf{0}$, then $(B, A)\begin{pmatrix}\mathbf{0}\\ \widehat{\mathbf{y}}\end{pmatrix} = \mathbf{0}$. It follows that $Z\widehat{\mathbf{x}} = \begin{pmatrix}\mathbf{0}\\ \widehat{\mathbf{y}}\end{pmatrix}$ for some vector $\widehat{\mathbf{x}}$ because $Z = \mathrm{nmb}((B, A))$. Consequently $Z_1\widehat{\mathbf{x}} = \widehat{\mathbf{y}}$.

Remark 5.1. *Having a* $\mathrm{ca}(A)$ *available, we can obtain a* $\mathrm{nmb}(A)$*, e.g., based on Fact 2.1.*

Remark 5.2. *Algorithm 5.1 is an aggregation process that first aggregates an input matrix A into the matrix Z_0 of a smaller size, then reapplies itself to this matrix to compute the matrix $X = \mathrm{ca}(Z_0)$, and finally disaggregates this output to produce the solution $Y = Z_1X = \mathrm{ca}(A)$.*

5.2 The regularization and preconditioning power of randomized western augmentation

Theorem 5.1. *Assume that $A \in \mathbb{C}^{m\times n}$, $m \le n$ and $r = m - \operatorname{rank} A$. Then (a) the matrices $C = (B, A)$ in Algorithm 5.1 are rank deficient for $q < r$, whereas (b) for $q \ge r$ the matrices B and (B, A) are rank deficient with a probability at most $\frac{r}{|\Delta|}$ provided that the entries of the matrix B are randomly and uniformly sampled from a finite set $\Delta \in \mathbb{C}$ of cardinality $|\Delta|$ and that the Subroutine* CA *fails with probability zero if its input matrix has full rank.*

Proof. $\operatorname{rank}(B, A) \le \operatorname{rank} B + \operatorname{rank} A \le q + \operatorname{rank} A = q + m - r$. This implies part (a). If $q \ge m - \operatorname{rank} A$ and the entries of the matrix B are indeterminates, then clearly the matrices B and (B, A) have full rank. Now part (b) of the theorem follows from Corollary 2.1. □

Theorem 5.1 implies that for $q \ge r$ the failure probability of Algorithm 5.1 is at most $\frac{r}{|\Delta|}$.

Our expected regularization is likely to decrease the condition number of the matrix A to the level of $\frac{\sigma_1(A)}{\sigma_{m-q}(A)}$ provided that $m - \rho < q < m$ and B is a Gaussian random matrix with the mean zero and a variance of the order $||A||^2$. This is implied by the two following theorems.

Theorem 5.2. *Suppose the matrix $C = (B, A)$ in Algorithm 5.1 has been scaled so that $||C|| \le 1$. Let $\operatorname{rank} C = m$, $\operatorname{rank} B = q$, and $\rho = \operatorname{rank} A = m - q > 0$. Let $A = S_A\Sigma_A T_A^H$ be a full SVD of the matrix A and write*

$$S_A^H C \operatorname{diag}(I_q, T_A) = (\bar{B}, \Sigma_A). \tag{5.1}$$

Note that the $n - m$ last columns of the matrix $(\bar{B}, \Sigma_A)$ in (5.1) vanish. Delete them and denote by

$$M = \begin{pmatrix}\bar{B}_0 & \widehat{\Sigma}_A\\ \bar{B}_1 & O\end{pmatrix} \tag{5.2}$$

the resulting nonsingular $m \times m$ matrix where $\widehat{\Sigma}_A$ is the $(m - q) \times (m - q)$ leading principal (northwestern) submatrix of the matrix Σ_A. Then $\operatorname{cond} C \le (\frac{1}{\sigma_{m-q}(A)} + \frac{1}{\sigma_q(\bar{B}_1)} + \frac{||\bar{B}_0||}{\sigma_{m-q}(A)\sigma_q(\bar{B}_1)})||C||$.

Proof. We have $\begin{pmatrix}\bar{B}_0\\ \bar{B}_1\end{pmatrix} = S_A^H B$, so that $||(\bar{B}_0^T, \bar{B}_1^T)|| = ||B||$. Invert equation (5.2) to obtain $M^{-1} = \begin{pmatrix}O & \bar{B}_1^{-1}\\ \widehat{\Sigma}_A^{-1} & -\widehat{\Sigma}_A^{-1}\bar{B}_0\bar{B}_1^{-1}\end{pmatrix}$ and deduce that $||M^{-1}|| \le ||\widehat{\Sigma}_A^{-1}|| + ||\bar{B}_1^{-1}|| + ||\widehat{\Sigma}_A^{-1}||\,||\bar{B}_1^{-1}||\,||\bar{B}_0||$. Substitute $||\widehat{\Sigma}_A^{-1}|| = \frac{1}{\sigma_{m-q}(\widehat{\Sigma}_A)} = \frac{1}{\sigma_{m-q}(A)}$, $||\bar{B}_1^{-1}|| = \frac{1}{\sigma_q(\bar{B}_1)}$, and $||M^{-1}|| = \frac{1}{\sigma_m(M)} = \frac{1}{\sigma_m(C)}$. Obtain that $\frac{1}{\sigma_m(C)} \le \frac{1}{\sigma_{m-q}(A)} + \frac{1}{\sigma_q(B_1)} + \frac{||\bar{B}_0||}{\sigma_{m-q}(A)\sigma_q(B_1)}$, which implies the theorem. □

Corollary 5.1. *Under the assumptions of Theorem 5.2 suppose $||B|| = ||A||$ and write $\kappa = \operatorname{cond} A$ and $\kappa_0 = \frac{||\bar{B}_0||}{\sigma_q(B_1)}$. Then* $\operatorname{cond} C \le \sqrt{2}(\kappa + \kappa_0 + \kappa\kappa_0)$.

Proof. Note that $||C|| \le \sqrt{||A||^2 + ||B||^2} = \sqrt{2}||A|| = \sqrt{2}||B||$ for $||A|| = ||B||$. Moreover $S_A^H B = \bar{B} = \begin{pmatrix}\bar{B}_0\\ \bar{B}_1\end{pmatrix}$, and so $||B|| = ||\bar{B}|| \ge ||\bar{B}_0||$. Substitute these relationships into Theorem 5.2. □

Next we assume that $B \in \mathbb{C}^{m \times q}$ is Gaussian random matrix and estimate the value $\sigma_q(\bar{B}_1)$.

Theorem 5.3. *Under the assumptions of Theorem 5.2 suppose $B \in \mathbb{R}^{n \times n}$ is a Gaussian random matrix with mean zero and a variance σ^2. Then $F_{\bar{B}_1}(y) \le cy\sqrt{q}/\sigma$.*

Proof. Apply part (a) of Theorem 2.7 for $G = (O, I_q)S_A^H$, $W = B$, $r_G = l = q \le m$, $GW = \bar{B}_1$, and $\sigma_{r_G}(G) = 1$. □

Corollary 5.2. *Under the assumptions of Theorem 5.3 the matrix $C = (B, A)$ is rank deficient with the probability zero.*

Proof. Theorem 5.3 implies that the matrix B_1 is singular with the probability zero. Therefore the corollary follows from equations (5.1) and (5.2). □

5.3 Further remarks

Remark 5.3. *Our correctness proof for Algorithm 5.1 applies to any integer $q \ge \operatorname{nul} A$. The following observations can guide us in choosing the integer parameter q. (Note some differences with the choice of the integer parameter r in northern augmentation.)*

1. *In virtue of Theorem 5.1* $\operatorname{rank}(B, A) < m$ *if* $q < m - \operatorname{rank} A$, *but we expect to have* $\operatorname{rank}(B, A) = m$ *if B is a scaled random $m \times q$ matrix and if $q \ge m - \operatorname{rank} A$.*

2. *According to the previous subsection, the condition number* $\operatorname{cond}(B, A)$ *is expected to be of the order $\frac{\sigma_1(A)}{\sigma_{m-q}(A)}$ in the case of an $m \times q$ scaled Gaussian random matrix B. Consequently by incrementing the integer parameter q we would expect to improve conditioning of the matrix (B, A) and thus to simplify the computation of its ca.*

3. *This advantage, however, should be weighed against the disadvantage of increasing the sizes of the $m \times (n + q)$ matrix (B, A) and the $q \times s$ matrix Z_0.*

4. *In numerical implementation of Algorithm 5.1 one should generate a Gaussian random matrix B with the mean zero and a variance of the order $||A||^2$ and work with numerical cas and numerical nullity instead of cas and nullity. For a fixed positive integer q one should seek a numerical ca of an $m \times n$ (possibly rank deficient or ill conditioned) input matrix A having an unknown positive numerical nullity r where $m \le n$ and $r \le q$. In this case one should apply (to the matrices (B, A) and Z_0) a randomized Subroutine* NUMERICAL CA *(e.g., numerical version of Algorithm 3.1 or 4.1) that either computes a numerical ca of an input matrix or outputs* FAILURE, *definitely if the matrix is rank deficient or ill conditioned, but only with a low probability otherwise.*

5. *Suppose an $m \times n$ matrix A is ill conditioned, whereas the $(m + q) \times n$ matrix (B, A) is well conditioned, that is* western augmentation acts as preconditioning. *(According to the previous subsection one should expect such a development under the choice of a scaled random matrix B unless the ratio $\sigma_1(A)/\sigma_{m-q}$ is large.) Then in numerical implementation of Algorithm 5.1 one must compute a* $\operatorname{ca}(B, A)$ *with high accuracy [32, Section 7], e.g., by applying iterative refinement and expecting that overall this would still simplify the original task wherever $q \ll \min\{m, n\}$ (see our Section 8 as well as [32, Sections 8 and 9] and [38]).*

We conclude with a recipe for saving some flops by combining western augmentation with a proper form of post-multiplication.

Remark 5.4. *Assume that Algorithm 5.1 produces the matrix (B, A) at its stage of western augmentation and that Algorithm 4.1 has been applied as the Subroutine* CA. *Then we can save some flops in Algorithm 5.1 by choosing $W = \operatorname{diag}(I_q, T)$ where either $T = A^H$ or T is a random $n \times m$ matrix scaled so that $||T|| \approx 1$. One can readily modify Theorems 4.1–4.4 to cover this case.*

6 Randomized northwestern augmentation

Given two positive integers m and n and a matrix $A \in \mathbb{C}^{m\times n}$, one can compute a $\mathrm{ca}(A)$ by applying Algorithm 5.1 with Algorithm 3.1 serving as the Subroutine *CA*. In this section we specify the resulting northwestern augmentation $A \to K = \begin{pmatrix} W & V \\ B & A \end{pmatrix}$, analyze it by combining the analysis in Sections 3 and 5, and supply some additional comments. In Section 7.4 we apply northwestern augmentation to preconditioning a nonsingular nonhomogeneous linear system of equations.

One can similarly employ the southeastern, southwestern or northeastern augmentations.

6.1 Cas and nmbs via randomized northwestern augmentation: an algorithm

Algorithm 6.1. A ca via randomized northwestern augmentation.

INPUT: *Three positive integers m, n, and ρ, a matrix $A \in \mathbb{C}^{m\times n}$ of rank ρ, and a random number generator.*

OUTPUT: *FAILURE or a* $\mathrm{ca}(A)$ *(cf. Remark 5.1).*

INITIALIZATION: *Fix two nonnegative integers $q \geq n - \rho$ and $r \geq m + q - n$ (see Remark 6.2).*

COMPUTATIONS:

1. *(*Norhwestern augmentation.*) Generate three random matrices V in $\mathcal{C}^{r\times n}$, B in $\mathcal{C}^{m\times q}$, and W in $\mathcal{C}^{r\times q}$. If the matrix $K = \begin{pmatrix} W & V \\ B & A \end{pmatrix} \in \mathcal{C}^{(m+r)\times(n+q)}$ is column rank deficient, output FAILURE and stop.*
2. *(*Aggregation.*) Otherwise compute matrices*

$$Y = (O, I_n)K^{(I)} \begin{pmatrix} O \\ B \end{pmatrix} \tag{6.1}$$

   and AY. IF $AY = O$, output $Y = \mathrm{ca}(A)$.
3. *Otherwise apply Algorithm 6.1 to the matrix AY to compute a $q \times s$ matrix $Z = \mathrm{ca}(AY)$.*
4. *(*Disaggregation.*) If the matrix Z has rank q, then compute and output the $n \times r$ matrix $YZ = \mathrm{ca}(A)$. Otherwise output FAILURE and stop.*

Note that $Y = I$ and Z is a $\mathrm{ca}(A)$ if $AY = O$.

Our remarks about Algorithms 3.1 and 5.1 can be readily extended to Algorithm 6.1. In particular Remark 5.1 can be applied unchanged, whereas Remark 5.2 changes as follows.

Remark 6.1. *Algorithm 6.1 is an aggregation process that first aggregates an input matrix A into the matrix AY of a smaller size, then reapplies itself to this matrix to output a matrix Z, and finally disaggregates this output to produce the solution $YZ = \mathrm{ca}(A)$ to the original input A.*

Here are some additional comments.

Remark 6.2. *We can first define the integer parameter q and generate a scaled random matrix $B \in \mathbb{C}^{m\times q}$ by following the recipes in Remark 5.3. Then we can compute the integer parameter r by following the recipes in Section 3.3 applied to the matrix (B, A) replacing A.*

The analysis of randomized northwestern augmentation in the next subsection combines our earlier analysis of northern and western augmentation and implies correctness of Algorithm 6.1.

6.2 Analysis of randomized northwestern augmentation

Theorem 6.1. (a) *Assume six positive integers m, n, q, r, s, and ρ such that $\rho \le \min\{m,n\}$ and $s = \min\{m+r, n+q, \rho+q+r\}$, and five matrices $A \in \mathcal{C}^{m\times n}$ of rank ρ, V in $\mathcal{C}^{r\times n}$, B in $\mathcal{C}^{m\times q}$, W in $\mathcal{C}^{r\times q}$, and $K = \begin{pmatrix} W & V \\ B & A \end{pmatrix} \in \mathcal{C}^{(m+r)\times(n+q)}$. Then we have* $\operatorname{rank} K \le s$.

(b) *Suppose under the assumptions of part (a) that either the entries of the matrices B, V, and W have been randomly and uniformly sampled from a set $\Delta \in \mathcal{C}$ of cardinality $|\Delta|$ or the entries of the matrices V and W have been randomly and uniformly sampled from such a set and $B = V$. Furthermore let $m \le \min\{n, \rho+q\}$. Then the matrix (B, A) has full rank m with a probability at least $1 - \frac{q}{|\Delta|}$. If in addition $r \ge n+q-m$, then* $\operatorname{rank} K = n+q = s$ *with a probability at least* $1 - \frac{q+r}{|\Delta|}$.

Proof. Part (a) of the theorem can be immediately verified. Part (b) is proved similarly to Theorems 3.1 and 5.1, based on Corollary 2.1. □

Theorem 6.2. *Under the assumptions of part (a) of Theorem 6.1, suppose that*

$$n+q \le m+r,\ q \le r,\ K^{(I)}K = I_{n+q} \text{ and } W^{(I)}W = I_q \tag{6.2}$$

for some matrices $K^{(I)}$ and $W^{(I)}$. Then

$$\mathcal{N}(A) \subseteq \mathcal{R}(Y) \text{ for } Y = (O, I_n)K^{(I)} \begin{pmatrix} O \\ B \end{pmatrix}. \tag{6.3}$$

Furthermore if $\operatorname{rank} B \le \operatorname{nul} A$, *then*

$$\mathcal{R}(Y) = \mathcal{N}(A). \tag{6.4}$$

Proof. Let $\mathbf{y} \in \mathcal{N}(A)$ and $\mathbf{x} \in \mathbb{C}^q$. Then $K \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} W\mathbf{x} + V\mathbf{y} \\ B\mathbf{x} \end{pmatrix}$. Substitute $\mathbf{x} = -W^{(I)}V\mathbf{y}$ and obtain that $K \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ B\mathbf{x} \end{pmatrix}$. Therefore $\mathbf{y} = (O, I_n)K^{(I)} \begin{pmatrix} \mathbf{0} \\ B\mathbf{x} \end{pmatrix}$. This proves claim (6.3), which implies claim (6.4) if $\operatorname{rank} B \le \operatorname{nul} A$ because $\operatorname{rank}(Y) \le \operatorname{rank} B$. □

Corollary 6.1. *Under the assumptions of part (a) of Theorem 6.1, suppose equations (6.2) hold and write $Y = (O, I_n)K^{(I)} \begin{pmatrix} O \\ B \end{pmatrix}$. Then*
(a) YZ is a $\operatorname{ca}(A)$ *if Z is a* $\operatorname{ca}(AY)$, *in particular if $AY = O$ and $Z = I$, and furthermore*
(b) Z is a $\operatorname{ca}(AY)$ *if YZ is a* $\operatorname{ca}(A)$ *and if the matrix Y has full column rank q.*

Proof. (a) Clearly $A(YZ) = (AY)Z = O$ if Z is a $\operatorname{ca}(AY)$. Conversely let $A\mathbf{u} = \mathbf{0}$. Then $\mathbf{u} = Y\mathbf{v}$ for some vector $\mathbf{v}$ in virtue of (6.3). Therefore $AY\mathbf{v} = A\mathbf{u} = \mathbf{0}$. It follows that $\mathbf{v} = Z\mathbf{z}$ for some vector $\mathbf{z}$ because Z is a $\operatorname{ca}(AY)$. Consequently $\mathbf{u} = Y\mathbf{v} = YZ\mathbf{z}$.

(b) Surely $(AY)Z = A(YZ) = O$ if YZ is a $\operatorname{ca}(A)$. Conversely let $AY\mathbf{u} = A(Y\mathbf{u}) = \mathbf{0}$. Then $Y\mathbf{u} = YZ\mathbf{v}$ for some vector $\mathbf{v}$ because YZ is a $\operatorname{ca}(A)$. Therefore $\mathbf{u} = Z\mathbf{v}$ since $\operatorname{rank} Y = q$. □

Assume the following specification of Algorithm 6.1 applied to an $m \times n$ matrix A: recursively augment the matrix A by appending new Gaussian random northern rows and western columns with the mean zero and a variance of the order $||A||^2$ (e.g., in the case of a mildly scaled input matrix A append standard Gaussian random rows and columns) until you arrive at an $h \times h$ nonsingular matrix K for $h = m+r = n+q$; then compute and output the matrix $(O, I_n)K^{(I)} \begin{pmatrix} O \\ B \end{pmatrix}$, expected to be a $\operatorname{ca}(A)$ if $\operatorname{rank} B \le \operatorname{nul} A$.

In virtue of our next theorem combined with Theorems 3.3 and 5.3, the above matrix K is expected to have condition number of the order of $\operatorname{cond} A$.

Theorem 6.3. *Assume dealing with the matrices A, B, V, W, and K in part (a) of Theorem 6.1. Suppose $A = S_A \Sigma_A T_A^H$ is a full SVD of the matrix A, $m+r = n+q$, $\Sigma_A = \mathrm{diag}(\widehat{\Sigma}_A, O)$, the matrices K and $\widehat{\Sigma}_A$ are nonsingular, and* $\mathrm{rank}\, A = \mathrm{rank}\, \Sigma_A = \mathrm{rank}\, \widehat{\Sigma}_A = \rho$. *Write*

$$M = \mathrm{diag}(I_r, S_A^H) K\, \mathrm{diag}(I_q, T_A) = \begin{pmatrix} W & V_0 & V_1 \\ \bar{B}_0 & \widehat{\Sigma}_A & O \\ \bar{B}_1 & O & O \end{pmatrix}, \tag{6.5}$$

$$f_1(\sigma) = \frac{1}{\sigma_\rho(A)} + \frac{1}{\sigma_q(\bar{B}_1)} + \frac{1}{\sigma_r(V_1)}, \tag{6.6}$$

$$f_2(\sigma) = \frac{||\bar{B}_0||}{\sigma_\rho(A)\sigma_q(\bar{B}_1)} + \frac{||V_0||}{\sigma_\rho(A)\sigma_r(V_1)} + \frac{||W||}{\sigma_q(\bar{B}_1)\sigma_r(V_1)}, \tag{6.7}$$

$$f_3(\sigma) = \frac{||\bar{B}_0||\ ||V_0||}{\sigma_\rho(A)\sigma_q(\bar{B}_1)\sigma_r(V_1)} \tag{6.8}$$

and assume that $q = \mathrm{rank}\, \bar{B}_1 = m - \rho$ and $r = \mathrm{rank}\, V_1 = n - \rho$, so that the matrices $\bar{B}_1$ and $\bar{V}_1$ are nonsingular, except that $\bar{B}_1$ (resp. V_1) is a dummy empty matrix if $\rho = m$ (resp. $\rho = n$). Then $\mathrm{cond}\, K \le (f_1(\sigma) + f_2(\sigma) + f_3(\sigma))||K||$.

Proof. By assumption the matrices K (and thus also M), $\bar{B}_1$, Σ_A, and V_1 are nonsingular, and clearly $||K^{-1}|| = ||M^{-1}||$. Equation (6.5) implies that

$$M^{-1} = \begin{pmatrix} O & O & \bar{B}_1^{-1} \\ O & \widehat{\Sigma}_A^{-1} & -\widehat{\Sigma}_A^{-1} \bar{B}_0 \bar{B}_1^{-1} \\ V_1^{-1} & -V_1^{-1} V_0 \widehat{\Sigma}_A^{-1} & V_1^{-1}(V_0 \widehat{\Sigma}_A^{-1} \bar{B}_0 - W)\bar{B}_1^{-1} \end{pmatrix}.$$

Deduce that $||K^{-1}|| = ||M^{-1}|| = f_1 + f_2 + f_3$ for $f_1 = ||\widehat{\Sigma}_A^{-1}|| + ||\bar{B}_1^{-1}|| + ||V_1^{-1}||$,

$$f_2 = ||\widehat{\Sigma}_A^{-1}||\ ||\bar{B}_1^{-1}||\ ||\bar{B}_0|| + ||\widehat{\Sigma}_A^{-1}||\ ||V_1^{-1}||\ ||V_0|| + ||\bar{B}_1^{-1}||\ ||V_1^{-1}||\ ||W||,$$

and $f_3 = ||\widehat{\Sigma}_A^{-1}||\ ||\bar{B}_1^{-1}||\ ||V_1^{-1}||\ ||\bar{B}_0||\ ||V_0||$.

Substitute $||\widehat{\Sigma}_A^{-1}|| = \frac{1}{\sigma_\rho(\widehat{\Sigma}_A)} = \frac{1}{\sigma_\rho(A)}$, $||\bar{B}_1^{-1}|| = \frac{1}{\sigma_q(\bar{B}_1)}$, and $||V_1^{-1}|| = \frac{1}{\sigma_r(V_1)}$ and obtain the claimed bound on the condition number $\mathrm{cond}\, K$. □

Corollary 6.2. *Under the assumptions of Theorem 6.3 let $||A|| = ||B|| = ||V|| = ||W||$. Write $\kappa = \mathrm{cond}\, A$, $\kappa_0 = \frac{||\bar{B}_0||}{\sigma_q(\bar{B}_1)}$, and $\kappa_1 = \frac{||V_0||}{\sigma_1(V_1)}$. Then* $\mathrm{cond}\, C \le 2(\kappa + \kappa_0 + \kappa_1 + \kappa\kappa_0 + \kappa\kappa_1 + \kappa_0\kappa_1 + \kappa\kappa_0\kappa_1)$.

Proof. Note that $||C||^2 \le ||A||^2 + ||B||^2 + ||V||^2 + ||W||^2$, and so $||C|| \le 2||A|| = 2||B|| = 2||V|| = 2||W||$. Moreover $S_A^H B = \begin{pmatrix} \bar{B}_0 \\ \bar{B}_1 \end{pmatrix}$, and so $||\bar{B}_0|| \le ||S_A^H B|| = ||B||$. Similarly deduce that $||V_0|| \le ||VT_A|| = ||V||$. Substitute these relationships into Theorem 6.3. □

Theorems 3.3 and 5.3 bound the values $\sigma_r(V_1)$ and $\sigma_q(\bar{B}_1)$ provided B, V, and W are Gaussian random matrices.

6.3 Strong regularization and strong preconditioning

All our results on the regularization and preconditioning power of northern, western, and northwestern augmentations and post-multiplication of the input matrix can be also applied to all its leading principal (that is northwestern) submatrices. In particular wherever the output matrix is expected to be nonsingular or well conditioned, so are its leading principal submatrices.

We refer the reader to [37] on some algorithmic applications of these properties.

6.4 Saving random parameters and a pitfall

Our analysis can be readily extended to the case where the matrix V is chosen to equal aB^H for a proper scalar a of our choice, e.g., $a = -1$. In this way we would involve fewer random parameters. The particular choice of $V = B^H$ can work as well, but is numerically inefficient where we request to obtain a Hermitian positive definite matrix $K = \begin{pmatrix} W & B^H \\ B & A \end{pmatrix}$. Indeed in this case the augmentation $A \Longrightarrow K$ cannot decrease the condition number due to the Interlacing Property of the eigenvalues of Hermitian matrices [19, Theorem 8.1.7].

7 The solution of a nonhomogeneous linear system of equations

In the previous sections we reduced a homogeneous linear system $A\mathbf{y} = \mathbf{0}$ to nonhomogeneous ones, of the form $CW = U$. Conversely, in the next two subsections we reduce the solution of a nonsingular linear system $A\mathbf{y} = \mathbf{b}$ to computing a null vector of either the matrix $(-\eta\mathbf{b}, A)$ for a nonzero scalar η or the matrix $(I_n - \frac{\mathbf{b}\mathbf{b}^H}{\mathbf{b}^H\mathbf{b}})A$. In Subsections 7.3 and 7.4 we employ our earlier techniques to precondition a linear system $A\mathbf{y} = \mathbf{b}$ provided a matrix A has a positive numerical nullity: we employ an approximate $\mathrm{nmb}(A)$ in Subsection 7.3 and randomized scaled northwestern augmentation in Subsection 7.4.

7.1 Solution with an auxiliary matrix defined by western augmentation

The null vector $\mathbf{z} = \begin{pmatrix} 1/\eta \\ \mathbf{y} \end{pmatrix}$ of the matrix $C = (-\eta\mathbf{b}, A)$ for a nonzero scalar η contains the vector $\mathbf{y}$ as a subvector. To compute the null vector we can apply the algorithms in the previous sections.

One can scale the matrix A to ensure that $||A|| = 1$ and can select the scalar η such that $||\eta\mathbf{b}|| = ||A||$. So we assume that $||\mathbf{b}|| = ||A|| = 1$ and then show that the map $A \to C = (-\eta\mathbf{b}, A)$ is expected to have preconditioning power on the average input A and $\mathbf{b}$.

Theorem 7.1. *Suppose $C = (-\mathbf{b}, A)$, $||A|| = ||\mathbf{b}|| = 1$, $A = S_A\Sigma_A T_A^H$ is a full SVD of an $n \times n$ matrix A, $S_A^H S_A = S_A S_A^H = T_A^H T_A = T_A T_A^H = I_n$, $\Sigma_A = \mathrm{diag}(\sigma_i)_{i=1}^n$, $\sigma_i = \sigma_i(A)$ for all i, $\mathbf{f} = (f_i)_{i=1}^n = -S_A^H\mathbf{b}$, $f_n \neq 0$, and $\gamma(\mathbf{f}) = \max_{i=1}^{n-1}|f_i|$. Then $\sigma_n(C) \geq \frac{|f_n|\sigma_{n-1} - (1+|f_n|)\sigma_n}{1+|f_n|}$.*

Proof. Write $\Sigma = \Sigma_A$ and $(\mathbf{f}, \Sigma) = S_A^H C\,\mathrm{diag}(1, T_A)$, so that $||\mathbf{f}|| = ||\mathbf{b}|| = 1$ and $\sigma_n(C) = \sigma_n(\mathbf{f}, \Sigma)$.

Let G be the $n \times n$ matrix obtained by deleting the last column of the matrix $(\mathbf{f}, \Sigma)$. The matrix G is nonsingular for $f_n \neq 0$, and we deduce from the Courant–Fischer minimax theorem that $\sigma_n(\mathbf{f}, \Sigma) \geq \sigma_n(G) - \sigma_n$. Therefore

$$\sigma_n(C) \geq \sigma_n(G) - \sigma_n. \tag{7.1}$$

It remains to estimate the values $\sigma_n(G) = \frac{1}{||G^{-1}||}$ from below or $||G^{-1}||$ from above. Write

$$g_i = \sigma_{i-1} \text{ and } \widehat{f}_i = f_{i-1} \text{ for } i = 2, \dots, n;\ g_1 = f_n\ , \widehat{f}_1 = 0, \text{ and } \widehat{\mathbf{f}} = (\widehat{f}_i)_{i=1}^n \tag{7.2}$$

and cyclically shift the rows of the matrix G down to arrive at the matrix $\widehat{G} = \mathrm{diag}(g_i)_{i=1}^n + \widehat{\mathbf{f}}\mathbf{e}_1^T$. Clearly $||\widehat{G}^{-1}|| = ||G^{-1}||$.

We have $\widehat{G} = \mathrm{diag}(g_i)_{i=1}^n(I_n + (\frac{\widehat{f}_i}{g_i})_{i=1}^n\mathbf{e}_1)$. Combine this equation and equations (7.2) and deduce that

$$\widehat{G}^{-1} = (I_n - (\frac{\widehat{f}_i}{g_i})_{i=1}^n\mathbf{e}_1^T)\,\mathrm{diag}(\frac{1}{g_i})_{i=1}^n = \mathrm{diag}(0, \mathrm{diag}(\frac{1}{\sigma_i})_{i=1}^{n-1}) - (\frac{\widehat{f}_i}{g_i f_n})_{i=1}^n\mathbf{e}_1^T + \frac{1}{f_n}\mathbf{e}_1\mathbf{e}_1^T.$$

Substitute $\widehat{f}_i = f_{i-1}$ for $i = 2, \dots, n$; $\widehat{f}_1 = 0$ (cf. (7.2)), and $\frac{1}{f_n} = \frac{f_n}{f_n g_1}$ and obtain that

$$||G^{-1}|| = ||\widehat{G}^{-1}|| \leq ||\,\mathrm{diag}(\frac{1}{\sigma_i})_{i=1}^{n-1}|| + ||(\frac{f_i}{g_i f_n})_{i=1}^n\mathbf{e}_n^T||.$$

Since $g_i = \sigma_i \geq \sigma_{n-1}$ for $i < n$ and $||(f_i)_{i=1}^{n-1}|| \leq ||\mathbf{f}|| = 1$, it follows that

$$||G^{-1}|| \leq \frac{1}{\sigma_{n-1}} + \frac{1}{|f_n|\sigma_{n-1}}||(f_i)_{i=1}^{n}|| \leq \frac{1}{\sigma_{n-1}} + \frac{1}{|f_n|\sigma_{n-1}} = \frac{1+|f_n|}{|f_n|\sigma_{n-1}}.$$

Consequently $\sigma_n(G) \geq \frac{|f_n|\sigma_{n-1}}{1+|f_n|}$ and $\sigma_n(C) \geq \frac{|f_n|\sigma_{n-1}}{1+|f_n|} - \sigma_n = \frac{|f_n|\sigma_{n-1}-(1+|f_n|)\sigma_n}{1+|f_n|}$. □

Corollary 7.1. *Under the assumptions of Theorem 7.1 we have* $\operatorname{cond} C \leq \frac{(1+|f_n|)\sqrt{2}}{|f_n|\sigma_{n-1}-(1+|f_n|)\sigma_n}$.

Proof. Recall that $\operatorname{cond} C = \frac{||C||}{\sigma_n(C)}$ and $||C|| \leq \sqrt{||A|| + ||\mathbf{b}||} = \sqrt{2}||A|| = \sqrt{2}$. Substitute these relationships into Theorem 7.1. □

The corollary shows that $\operatorname{cond} C$ has the order of at most σ_1/σ_{n-1} (rather than σ_1/σ_n) unless the value $|f_n|$ is small. (Note the similarity of this result to Theorem 5.2 for $q = 1$.) Therefore unless the value $|f_n|$ is small, we can expect to arrive at a well conditioned matrix C where the input matrix A is ill conditioned and has numerical nullity one. (Note that on the average $|f_n| = \frac{1}{\sqrt{n}}$ on the unit sphere $||\mathbf{f}|| = 1$.)

We would still need to perform some stages of our solution algorithm with a high precision, but we decrease the overall computational cost by confining the high precision computations to relatively few flops performed in iterative refinement of a null vector of the well conditioned matrix C. At that stage we would apply fast advanced algorithms for accurate sums and products (see our Section 8 and the papers [32], [38]).

7.2 Solution with an auxiliary matrix defined via projection

Next we solve a linear system $A\mathbf{y} = \mathbf{b}$ via computing a null vector of the auxiliary matrix $(I_n - \frac{\mathbf{b}\mathbf{b}^H}{\mathbf{b}^H\mathbf{b}})A$, which projects the input matrix A into the linear space orthogonal to the vector $\mathbf{b}$.

The projection little affects the condition number $\operatorname{cond} A$ on the average input A and $\mathbf{b}$ of a large size (although decreases it to one for 2×2 inputs).

Algorithm 7.1. Solution with an auxiliary matrix defined by projection

INPUT: *A nonsingular $n \times n$ matrix A and a vector $\mathbf{b}$ of dimension n.*

OUTPUT: *The solution $\mathbf{y}$ to the linear system $A\mathbf{y} = \mathbf{b}$.*

COMPUTATIONS:

1. *Compute the projection matrix $I_n - \frac{\mathbf{b}\mathbf{b}^H}{\mathbf{b}^H\mathbf{b}}$ and the auxiliary matrix $\widehat{A} = (I_n - \frac{\mathbf{b}\mathbf{b}^H}{\mathbf{b}^H\mathbf{b}})A$ (having rank $n-1$ in virtue of Corollary 7.2 below).*
2. *Compute its nontrivial null vector $\widehat{\mathbf{y}} = x\mathbf{y} \neq \mathbf{0}$ (e.g., based on the algorithms in the previous sections).*
3. *Compute the scalar x from the vector equation $A\widehat{\mathbf{y}} = x\mathbf{b}$.*
4. *Compute and output the solution vector $\mathbf{y} = \widehat{\mathbf{y}}/x$.*

The following simple results together imply correctness of the algorithm.

Lemma 7.1. $(I_n - \frac{\mathbf{b}\mathbf{b}^H}{\mathbf{b}^H\mathbf{b}})\mathbf{b} = \mathbf{0}$.

Corollary 7.2. $\operatorname{rank} \widehat{A} = \operatorname{rank}(I_n - \frac{\mathbf{b}\mathbf{b}^H}{\mathbf{b}^H\mathbf{b}}) = n - 1$.

Corollary 7.3. $\widehat{A}\mathbf{y} = \mathbf{0}$.

Proof. $\widehat{A}\mathbf{y} = ((I_n - \frac{\mathbf{b}\mathbf{b}^H}{\mathbf{b}^H\mathbf{b}})A)(A^{-1}\mathbf{b})$ because $\widehat{A} = (I_n - \frac{\mathbf{b}\mathbf{b}^H}{\mathbf{b}^H\mathbf{b}})A$ and $\mathbf{y} = A^{-1}\mathbf{b}$. Therefore $\widehat{A}\mathbf{y} = (I_n - \frac{\mathbf{b}\mathbf{b}^H}{\mathbf{b}^H\mathbf{b}})\mathbf{b}$, which vanishes in virtue of Lemma 7.1. □

7.3 Solution with auxiliary matrices defined by approximate nmbs

Assume that an $n \times n$ nonsingular normalized input matrix A with $||A|| = 1$ has a small positive numerical nullity r (so that the ratio $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$ is not large, whereas $\sigma_1(A) \gg \sigma_{n-r+1}(A)$). Further assume that approximate nmbs M_1 and N_1 in $\mathbb{C}^{n\times r}$ of the matrices A and A^H, respectively, are available (possibly computed based on a numerical version of one of our algorithms in the previous sections).

Then we can generate two standard Gaussian random matrices $S,\ T \in \mathbb{C}^{n\times(n-r)}$ and compute the matrices $M_0 = AS$, $N_0 = AT$, and $F = (M_0, M_1)^H A(N_0, N_1) = \begin{pmatrix} F_{00} & F_{01} \\ F_{10} & F_{11} \end{pmatrix}$ where $F_{ij} = M_i^H A N_j$ for each of i, j being zero or one.

The matrices AM_1 and $A^H N_1$, and consequently also the block submatrices $F_{01} \in \mathbb{C}^{n\times r}$, $F_{10} \in \mathbb{C}^{r\times n}$, and $F_{11} \in \mathbb{C}^{r\times r}$ nearly vanish, having the norms of at most the order of $\sigma_{n-r+1}(A) \ll \sigma_{n-r}(A)$. Therefore the $O(n^2 r)$ flops involved in the computation of the $(2n - r)r$ entries of these blocks (versus the order of n^3 flops used overall) must be performed in extended precision to counter the expected cancellation of the leading digits.

In contrast the value $\sigma_{n-r}(F_{00}) \in \mathbb{C}^{(n-r)\times(n-r)}$ is likely to have the order of $\sigma_{n-r}(A)$ in virtue of Theorem 2.7, and therefore the block F_{00} in the 2×2 block matrix F is dominant, nonsingular and well conditioned.

The map $A \Longrightarrow F$ and the block Gaussian elimination reduce the computation of the inverse A^{-1} and the solution of a linear system $A\mathbf{y} = \mathbf{b}$ to the similar operations with the matrices F_{00} and $G = F_{11} - F_{10}F_{00}^{-1}F_{01} \in \mathbb{C}^{n\times r}$ of smaller sizes, expected to be nonsingular and better conditioned. (The matrix G is called the Gauss transform and Schur complement [19].)

The tests of this technique in [38] have confirmed its strong preconditioning power.

7.4 Solution with preconditioning via northwestern augmentation

Suppose the $n \times n$ input matrix A has a small positive numerical nullity r. Then according to our study in Section 6.2, northwestern augmentation $K = \begin{pmatrix} W & V \\ B & A \end{pmatrix}$ for Gaussian random matrices $W \in \mathbb{C}^{r\times r}$, V, and B with the mean zero and a variance of the order $||A||^2$ produces a matrix K expected to be nonsingular and well conditioned, together with its block W. Theorem 1.1 implies that $\mathbf{y} = A^{-1}\mathbf{b} = (I_n - S^{-1}BW^{-1}R^{-1}VS^{-1})\mathbf{b}$ for $R = I + VBW^{-1}$ and $S^{-1} = (O_{n,r}, I_n)K^{-1}\begin{pmatrix} O_{r,n} \\ I_n \end{pmatrix}$ denoting the $n \times n$ trailing principal block of the matrix $K^{-1} = \begin{pmatrix} X & Y \\ Z & S^{-1} \end{pmatrix}$. This reduces the solution of the linear system $A\mathbf{y} = \mathbf{b}$ essentially to the inversion of the matrices W and R and the computation of the products $S^{-1}\mathbf{b}$ and $S^{-1}B$. The matrix equation $K^{-1}K = I_{n+r}$ implies that $ZW + S^{-1}B = O_{n,r}$ and consequently $ZW = -S^{-1}B$. Therefor we can compute the product $ZW = (O_{n,r}, I_n)K^{-1}\begin{pmatrix} W \\ O_{r,n} \end{pmatrix}$ instead of the product $-S^{-1}B$.

If the matrices R, W and K are well conditioned (as we can expect provided W, V, and B are Gaussian random matrices with the mean zero and a variance of the order $||A||^2$), then we can decrease the overall cost of computing the solution (see Section 8).

If the matrix A is given with its displacement generator of a small length d, we are motivated to choose scaled random matrices W, B, and V with consistent structure, representing them as well as the matrix K with displacement generators of length in $O(d)$. By employing this structure we can accelerate our computations with these matrices (cf. [31]). In the special case of a Toeplitz matrix A we can choose augmentation that produces a Toeplitz matrix K and then we can exploit this structure based either on Theorem 1.1 or alternatively on Theorem 2.3 and Remark 2.2. Our formal (resp. empirical) support for the regularization (resp. preconditioning) power of our randomized augmentation techniques can be extended to the respective randomized structured augmentation techniques.

8 Computational cost estimates

Clearly our algorithms accelerate the customary computation of nmbs and null vectors by avoiding pivoting and orthogonalization. Table 9.1 shows the respective decrease of the CPU time for the computation of null vectors of Toeplitz matrices versus the customary algorithms, whereas Tables 9.3 and 9.4 show that the output accuracy of the customary solution is matched by our algorithms and furthermore can be dramatically enhanced in a few loops of iterative refinement.

On the top of such advances (due to the regularization power of our randomized augmentation), the augmentation is also expected to have preconditioning power under proper scaling (cf. Table 9.2). This enables acceleration of the customary algorithms by roughly the factor $n/\min\{d, r\}$ where an $n \times n$ input matrix has a positive numerical nullity r and a displacement rank d.

In this section we specify the latter advance for the solution of a nonsingular ill conditioned linear system of equations, but the resulting estimates and their derivation can be readily extended to the computation of a nmb or a null vector of an ill conditioned matrix, which can be reduced to solving the associated nonsingular linear systems of equations (see Algorithms 3.1, 4.1, 5.1, and 6.1, and Part 5 of Remark 5.3).

If an $n \times n$ matrix A has a positive numerical nullity r, then the singular value $\sigma_{n-r+j}(A)$ is small relatively to the norm $||A||$, so that the matrix A lies near the algebraic variety of dimension $n - r$ defined by the r equations $\sigma_{n-r+j}(A) = 0$ for $j = 1, 2, \dots, r$. Next we specially consider the most typical case where $r = 1$ and the variety has the largest dimension $n - 1$.

8.1 The case of input matrices with numerical nullity one: outline

Suppose we apply augmentation in Section 7.1 to solve a nonsingular linear system $A\mathbf{y} = \mathbf{b}$ of n equations whose matrix has numerical nullity one, whereas the $n \times (n + 1)$ matrix $(\mathbf{b}, A)$ is well conditioned and $\mathrm{cond}(\mathbf{b}, A) \approx \frac{\sigma_1(A)}{\sigma_{n-1}(A)} \ll \mathrm{cond}\, A$ (cf. Sections 5.2, 5.3, and 7.1). Furthermore suppose that application of Algorithms 3.1 or 4.1 has reduced the computation of a null vector of the matrix $(\mathbf{b}, A)$ to the solution of a nonsingular and well conditioned linear system $F\mathbf{x} = \mathbf{c}$ of $n + 1$ or n equations, respectively. Likewise we can expect to arrive at such a nonsingular and well conditioned linear system $F\mathbf{x} = \mathbf{c}$ of $n + 1$ equations by applying randomized scaled northwestern augmentation in Section 7.4 for $r = 1$.

In all these cases we need a highly accurate solution of the linear system $F\mathbf{x} = \mathbf{c}$ because the input matrix A is ill conditioned. We, however, involve no flops in a high precision and instead apply more flops in a low precision; overall we still solve the original linear system $A\mathbf{y} = \mathbf{b}$ faster.

We first compute an approximate inverse $X \approx F^{-1}$ of the well conditioned matrix F and then iteratively refine the crude initial solution $X\mathbf{c}$ to the linear system $F\mathbf{x} = \mathbf{c}$ (cf. [32], [38]). (Instead of an approximate inverse one can employ, e.g., LU factorization of the matrix F.)

We can compute a matrix X by means of Gaussian elimination, whose computational cost dominates the cost of preprocessing. This stage involves as many flops as the direct solution of the original linear system $A\mathbf{y} = \mathbf{b}$, but the matrix F is expected to be well conditioned (unlike the matrix A), and if it is well conditioned, we can perform these flops in a lower precision.

At the stage of iterative refinement the computational cost is lower than in Gaussian elimination by roughly the factor L_A/M_F. Here and hereafter M_W and L_W denote the number of flops used in the customary algorithms for multiplication of a matrix W by a vector and solving a nonsingular linear system of equations with the coefficient matrix W, respectively. In the case of $n \times n$ general (resp. Toeplitz or Hankel) matrix W, Gaussian elimination (resp. its structured variant in [16]) supports the upper bound on L_W of the order n^3 (resp. n^2), whereas $M_W = 2n^2 - n$ (resp. $M_W = O(n \log n)$).

Thus in both of the above stages we significantly accelerate the customary solution.

8.2 The case of input matrices with numerical nullity one: elaboration

Next we specify the cost estimates outlined in the previous subsection.

Let $\mu(l)$ denote the time-complexity of a flop performed with a precision l. Expressed in terms of the number of bit-operations involved, $\mu(l)$ is a superlinear function, which ranges between the orders of $(l \log l) \log\log l$ and l^2, depending on the computer environement (cf. [2], [14], and our Remark 8.1).

Then the customary algorithms solve the original and auxiliary linear systems $A\mathbf{y} = \mathbf{b}$ and $F\mathbf{x} = \mathbf{c}$ within the time-complexity bounds $c_{\rm ls} = L_A\mu(p_s)$ and $c_{\rm me} = L_F\mu(p)$, respectively, where p_s and p denote the precision used in these computations. We choose the precison p substantially smaller than p_s where the matrix F is well conditioned, but the matrix A is not.

Every loop of iterative refinement amounts essentially to multiplying each of the matrices F and $X \approx F^{-1}$ by a vector and contributes the order of $p - \log_2 \operatorname{cond} F$ correct bits per an output entry provided that p exceeds $\log_2 \operatorname{cond} F$ [19], [21], [32], [42]. Iterative refinement can produce the output values as the sums implicitly represented with a high precision by their lower (e.g., single or double) precision summands, where every refinement step contrributes a summand to every sum until convergence (cf. [32], [34]). To ensure the output precision p_s we use the order of h refinement steps where $(p - \log_2 \operatorname{cond} F)h = p_s$. For a well conditioned matrix F we can realistically assume that $2\log_2 \operatorname{cond} F \le p$ (so that $ph \le 2p_s$). Then the overall cost of performing iterative refinement $c_{\rm ir}$ is in $O(M_F\mu(p)\frac{p_s}{p})$. This implies acceleration by the factor $\frac{\mu(p_s)}{\mu(p)}$, which is significant where p is significantly less than p_s.

Let us compare this bound with the time-complexity $c_{\rm ls} = L_A\mu(p_s)$ of solving the original linear system by means of the customary algorithms. The ratio $c_{\rm ir}/c_{\rm ls}$ is essentially the product of two factors $f_1 = L_A/M_F$ and $f_2 = \mu(p_s)p/(\mu(p)p_s)$, roughly representing the two decreases in the number of flops and in computational precision, respectively.

The first factor f_1 has the order n where both matrices A and F are general and has the order $n/\log n$ where both are Toeplitz or Hankel matrices or have the structure of Toeplitz or Hankel type. In both cases we achieve dramatic acceleration of the customary algorithms because $f_2 > 1$ for $p_s > p$. More precisely, the second factor f_2 is close to one where $\mu(l)$ is close to a linear function in l and has the order p_s/p where $\mu(l)$ is quadratic function.

To support the latter estimates in the case of matrices A with Toeplitz or Hankel structure, we must preserve the structure for the matrix F as well. This, however, comes almost automatically because appending c columns and r rows to a matrix can increase its displacement rank by at most $c + r$ [31], and in our case $c = r = 1$.

Remark 8.1. *We separately estimate the acceleration of the customary algorithms due to saving the flops involved and to decreasing the computational precision, but one can readily combine these estimates into a single acceleration factor under the Boolean or bit operation model of computing. (This model reflects the impact of both number of flops used and the precision of computing.) If the ratio p_s/p is sufficiently large, then under this model our algorithms support nearly optimal upper bounds on the computational cost, which are smaller than the cost of the customary algorithms by roughly the factor of n and are within a polylogarithmic factor from an information lower bound [38].*

8.3 The case where numerical nullity exceeds one

Our analysis can be quite readily extended to the case where the input matrix A has any positive numerical nullity q and where we apply randomized scaled northwestern augmentation in Section 7.4 for $r = q$.

In this case the original task is reduced essentially to solving $q+1$ linear systems with an $n \times n$ well conditioned auxiliary matrix F. The acceleration factor versus the customary algorithms becomes roughly $n/(q+1)$, and so we can still achieve their significant acceleration where $q + 1 \ll n$.

If however, the input matrix A as well as the the structured auxiliary matrix F are represented with their displacement generators of a small length d, then the matrix F^{-1} has the same displacement rank as F and is readily expressed via the solution of $2d$ linear systems of equations with the matrices F and F^T [31, Theorem 1.5.3 and Section 4.4]. Therefore in this case we have $c_{\rm ir} = O(dM_F\mu(p)\frac{p_s}{p})$, which implies acceleration by roughly the factor n/d of the customary solution

to a nonsingular linear system $A\mathbf{y} = \mathbf{b}$ of n equations, even where the structured matrix A has a large numerical nullity.

There is a delicate point here, however. To keep the displacement rank of the matrix F small we must restrict the number of random parameters used in the augmentation. E.g., given a Hankel matrix A, we can choose Hankel matrices $W \in \mathbb{C}^{q\times q}$, V and B that define a Hankel matrix $K = \begin{pmatrix} W & V \\ B & A \end{pmatrix}$, but due to its structure all its entries are already defined by the matrix A, except for $2q-1$ leading entries in the first row, which we can choose at random.

All our results on regularization and preconditioning by means of augmentation can be quite readily extended under this restriction, except for the extension of our basic Theorem 2.4, taken from [43]. Such an extension is an open problem. Consequently the proof of the preconditioning power of randomized scaled augmentation is also an open problem in the case of large structured inputs of this class.

Motivated by the test results in Table 9.1 and in [37] (the latter tests show that random Toeplitz matrices tend to be well conditioned), we conjecture that Theorem 2.4 still holds at least in the case of Toeplitz and Hankel matrices.

Finally we can avoid this and some other problems by devising algorithms based on the homotopy continuation techniques that involve no randomization, do not depend on numerical nullity, but still accelerate the customary algorithms by roughly the factor $n/\log^2 n$ [38] .

8.4 Sparse inputs, multilevel Toeplitz or Hankel inputs, and iterations using no approximate inverses

In lieu of iterative refinement one can employ other iterations such as the Conjugate Gradient and GMRES algorithms (cf. [1], [15], [19, Sections 10.2–10.4]). Like iterative refinement, they perform $O(M_A)$ flops per iteration loop, but unlike iterative refinement, they use no approximate inverse and thus save flops required for its computation. This is a significant advantage in the case of a sparse unstructured linear system as well as a multilevel Toeplitz or Hankel linear system, whose solution is generally much harder than multiplication of its coefficient matrix by a vector.

Typically decreasing the condition number of an input matrix is more critical (and thus preconditioning is more important) for the convergence of such algorithms versus iterative refinement. Theoretically the Conjugate Gradient algorithm requires at most n iteration loops to yield the solution of a nonsingular linear system of n equations, thus implying the bound $L_A = O(nM_A)$ for any $n \times n$ nonsingular matrix A, but these results are only proved for error-free computations and are not observed in the presence of rounding errors.

Various important algebraic geometric computations are routinely reduced to the solution of multivariate polynomial systems of equations and further to linear systems of equations with multilevel Toeplitz or Hankel coefficient matrices. For such an $n \times n$ multilevel Toeplitz or Hankel input matrices A, we have M_A in $O(n \log n)$ and frequently in $O(n)$, due to sparseness [13], [28]. The known numerical algorithms for the solution of linear systems with these multilevel matrices, however, run in cubic time.

Likewise sparse matrices A are also characterized by the bound $M_A \ll L_A$.

Generally we have neither effective displacement representation for a sparse or multilevel Toeplitz or Hankel matrix nor expressions for its inverse via the solutions of a small number of linear systems of equations with this matrix. The computation of its approximate inverse or factorization supporting iterative refinement becomes expensive, but our regularization and preconditioning techniques based on the other iterations such as the CG and GMRES algorithms keep their power. In particular if a sparse or multilevel Toeplitz or Hankel matrix A has a small positive numerical nullity r, then our randomized augmentations can reduce the computations with this matrix to the case of a matrix F expected to be well conditioned, and if it is indeed well conditioned, then our techniques support the acceleration of the known algorithms for a nonsingular sparse or multilevel Toeplitz and Hankel linear system of n equations by the factor $(n/r)\mu(p_s)p/(\mu(p)p_s)$.

9 Numerical tests

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. We conducted the tests 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 $\{x : \ 0 \le x < 1\}$. To shift to the range $\{y : \ b \le y \le a + b\}$ for fixed real a and b, we applied the linear transform $x \to y = ax + b$. CPU time was measured with the mclock function. We computed QR factorizations and SVDs by applying the LAPACK procedures DGEQRF and DGESVD, respectively.

9.1 Computations with Toeplitz matrices

a) Generation of singular Toeplitz matrices

To generate an $n \times n$ singular Toeplitz matrix, we first sampled $2n-2$ random entries $a_{i,j}$ for $j = 1$, $i = 1, \dots, n-1$ and for $i = 1$, $j = 2, \dots, n$ in the range $[-1, 1)$, then defined the $(n-1)^2$ entries $a_{i+1,j+1} = a_{i,j}$ for $i, j = 1, \dots, n-1$, and finally set $a_{n,1} = 0$. We arrived at an $n \times n$ Toeplitz matrix $A_0 = (a_{i,j})_{i,j=1}^{n}$, computed the entry $y_{n,1}$ of its inverse $A_0^{-1} = (y_{i,j})_{i,j=0}^{n-1}$, and changed the $(1, n)$th entry of the matrix A_0 into $a_{n,1} = -1/y_{n,1}$. (As we expected in virtue of Lemma 2.1, we always had $y_{n,1} \det A_0 \ne 0$ in our tests. Had we had $y_{n,1} = 0$, we could have regenerated the matrix A_0, whereas had it been singular, we would have stopped the computations and output it.)

The resulting matrix $A = (a_{i,j})_{i,j=1}^{n}$ had nullity one. Indeed it was a rank-one A-modification of a nonsingular matrix A_0, whereas $A\mathbf{y} = \mathbf{0}$ for $\mathbf{y} = A_0^{-1}\mathbf{e}_1$ because $A_0\mathbf{y} = \mathbf{e}_1$, $A = A_0 - \frac{1}{y_{n,1}}\mathbf{e}_1\mathbf{e}_n^T$, and $\mathbf{e}_n^T\mathbf{y} = y_{n,1}$.

b) Augmentation of singular Toeplitz matrices and the computation of their null vectors

We embedded our $n \times n$ singular Toeplitz matrix $A = (a_{i,j})_{i,j=1}^{n}$ into an $(n+1) \times (n+1)$ Toeplitz matrix $K = (a_{i,j})_{i,j=0}^{n} = \begin{pmatrix} w & \mathbf{v}^T \\ \mathbf{b} & A_0 \end{pmatrix}$ for $w = a_{0,0}$, $\mathbf{b} = (a_{i,0})_{i=1}^{n}$, and $\mathbf{v} = (a_{0,j})_{j=1}^{n}$. We defined the entries $a_{i,0}$ and $a_{0,j}$ for $i, j = 0, 1, \dots, n-1$ by applying the equations $a_{i,j} = a_{i+1,j+1}$ and sampled the two entries $a_{n,0}$ and $a_{0,n}$ at random in the range $[-1, 1)$. For such a matrix K we applied Theorem 6.2 for $r = 1$, to compute a null vector of the matrix A given by the vector $(0, I_n)K^{-1}\begin{pmatrix} 0 \\ \mathbf{b} \end{pmatrix}$. This amounted to solving a nonsingular Toeplitz linear systems of equations with the matrix K. For that task we applied the code in [45], based on the algorithms in [25], [46], [47]. For comparison 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 [19, Section 5.5], but we avoided them in our tests.

c) Output data in the tests with Toeplitz matrices

We use the abbreviations "n.-w.a.", "QR", and "SVD" as our pointers to the northwestern augmentation (based on Algorithm 6.1), QR factorization, and SVD, respectively. Table 9.1 covers our computation of null vectors for Toeplitz matrices. It shows the CPU time of this computation for each of the three methods as well as the ratios of these data for the QR-based and SVD-based solutions versus northwestern augmentation. The ratios are displayed in the last two columns of the

table. The CPU time is measured in terms of the CPU cycles. They can be converted 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 hyphen "-" where the tests required too long runtime and were not completed.

Table 9.1: CPU time for computing a null vector of a Toeplitz matrix

dimension	n.-w.a.	QR	SVD	QR/n.-w.a.	SVD/n.-w.a
256	3.8	18.4	317.8	4.8	83.6
512	8.0	148.0	5242.1	18.5	655.3
1024	16.1	1534.2	87371.2	97.0	5522.6
2048	33.6	11750.3	–	357.7	–
4096	79.5	–	–	–	–
8192	169.5	–	–	–	–

9.2 Computations with unstructured matrices

a) Generation of input matrices

We first fixed pairs of n and k for $n = 64, 128$ and $k = 7$. Then for every pair (n, k) we generated $m = 100$ instances of matrices A, B, V_0, and V_1 and vectors $\mathbf{b}$ as follows.

The matrices A have been generated as the error-free products $S\Sigma T^H$ where S and T were $n \times n$ random orthonormal matrices (generated with double precision) and $\Sigma = \mathrm{diag}(\sigma_j)_{j=1}^{n}$, $\sigma_{n-j} = 10^{j-17}$ for $j = 1, \ldots, k$, and $\sigma_{n-j} = 1/(n-j)$ for $j = k+1, \ldots, n-1$ (cf. [21, Section 28.3]).

B was random $n \times k$ matrix with $||B|| = ||A||$.

V was $k \times (n+k)$ matrix $V = (V_0, V_1)$ where and V_0 was the $k \times k$ identity matrix I_k and $V_1 = B^T$.

For every choice of these matrices we performed preconditioning tests and the solution tests as follows.

b) Preconditioning tests

We computed m ratios $\frac{\mathrm{cond}\, A}{\mathrm{cond}\, M}$ for $M = \begin{pmatrix} V_0 & V_1 \\ B & A \end{pmatrix}$.

Table 9.2 displays the average (mean), minimum, maximum, and standard deviation for the m ratios for $n = 64$ and $n = 128$.

Table 9.2: ratios $\frac{\mathrm{cond}\, A}{\mathrm{cond}\, M}$

matrix size	min	max	mean	std
64×64	3.29×10^9	1.65×10^{13}	2.49×10^{12}	2.60×10^{12}
128×128	8.27×10^8	2.56×10^{12}	5.51×10^{11}	6.44×10^{11}

c) The solution tests

We solved nonsingular linear systems $A\mathbf{y} = \mathbf{b}$ where A was the matrix generated above, $\mathbf{b}$ was a random vector, and we scaled them to have $||\mathbf{b}|| = ||A|| = 1$. We first computed the null vector $\mathbf{z}$ of the matrix $(-\mathbf{b}, A)$, then scaled it to obtain the vector $(1, \mathbf{y})^H$, and finally output the solution vector $\mathbf{y}$.

Tables 9.3 and 9.4 display the average (mean), minimum, maximum, and standard deviation for the relative residual norms $\frac{||A\mathbf{y}-\mathbf{b}||}{||\mathbf{y}||}$ in our tests for $n = 64$ and $n = 128$, respectively. For each input instance we computed the solution in two ways, that is by performing two iterations of the extended iterative refinement and with no such iteratiion.

Table 9.3: relative residual norms in the solution tests with 64×64 inputs

refinement	**min**	**max**	**mean**	**std**
2 iterations	7.89×10^{-48}	8.26×10^{-44}	1.40×10^{-45}	8.47×10^{-45}
no iteration	1.43×10^{-31}	7.30×10^{-28}	1.69×10^{-29}	9.12×10^{-29}

Table 9.4: relative residual norms in the solution tests with 128×128 inputs

refinement	**min**	**max**	**mean**	**std**
2 iterations	1.31×10^{-46}	1.37×10^{-43}	4.11×10^{-45}	1.67×10^{-44}
no iteration	8.57×10^{-31}	1.92×10^{-27}	5.12×10^{-29}	2.55×10^{-28}

References

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

[2] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Algorithms.* Addison-Wesley, Reading, MA, 1974.

[3] R. P. Brent, F. G. Gustavson, D. Y. Y. Yun, Fast Solution of Toeplitz Systems of Equations and Computation of Padé Approximations, *J. Algorithms*, **1**, 259–295, 1980.

[4] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations,* volume 1: Fundamental Algorithms, Birkhäuser, Boston, 1994.

[5] R.E. Cline, R.J. Plemmons, and G. Worm, Generalized inverses of certain Toeplitz matrices. *Linear Algebra and Its Applications,* **8**, 25–33, 1974.

[6] Coppersmith, S. Winograd, Matrix Multiplicaton via Arithmetic Progressions. *J. Symbolic Comput.*, **9(3)**, 251–280, 1990.

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

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

[9] C.-E. Drevet, M. N. Islam, E. Schost, Optimization Techniques for Small Matrix Multiplication, preprint, 2010.

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

[11] K. R. Davidson, S. J. Szarek, Local Operator Theory, 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.

[12] 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.

[13] I. Z. Emiris, V. Y. Pan, Symbolic and Numerical Methods for Exploiting Structure in Constructing Resultant Matrices, *J. of Symbolic Computation*, **33**, 393-413, 2002. Proc. Version in *ISSAC 97*.

[14] M. Fürer, Faster Integer Multiplication, *Proceedings of 39th Annual Symposium on Theory of Computing (STOC 2007)*, 57–66, ACM Press, New York, 2007.

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

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

[17] I. Gohberg, V. Olshevsky, Complexity of Multiplication with Vectors for Structured Matrices, *Linear Algebra and Its Applications*, **202**, 163–192, 1994.

[18] I. Gohberg, A. Semencul, On the Inversion of Finite Toeplitz Matrices and Their Continuous Analogs, *Matematicheskiie Issledovaniia*, **2**, 187–224, 1972.

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

[20] G. Heinig, Beitrage zur spektraltheorie von Operatorbuschen und zur algebraischen Theorie von Toeplitzmatrizen, Dissertation **B**, *TH Karl-Marx-Stadt*, 1979.

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

[22] G. Heinig, K. Rost, *Algebraic Methods for Toeplitz-like Matrices and Operators, Operator Theory*, **13**, Birkhäuser, 1984.

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

[24] T. Kailath, S. Y. Kung, M. Morf, Displacement Ranks of Matrices and Linear Equations, *Journal Math. Analysis and Appls*, **68(2)**, 395–407, 1979.

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

[26] 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.

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

[28] B. Mourrain, V. Y. Pan, Multivariate Polynomials, Duality and Structured Matrices, *J. of Complexity*, **16**, **1**, 110–180, 2000. (Proceedings Version in *STOC'98*.)

[29] 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.

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

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

[32] V. Y. Pan, D. Grady, B. Murphy, G. Qian, R. E. Rosholt, A. Ruslanov, Schur Aggregation for Linear Systems and Determinants, *Theoretical Computer Science*, *Special Issue on Symbolic–Numerical Algorithms* (D.A. Bini, V.Y. Pan, J. Verschelde editors), **409**, **2**, 255–268, 2008.

[33] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, *Linear Algebra and Its Applications*, **432**, 1070–1089, 2010. Proceedings version in *Proc. of the Third International Computer Science Symposium in Russia (CSR 2008)*, *Lecture Notes in Computer Science (LNCS)*, **5010**, 372–383, 2008.

[34] V. Y. Pan, B. Murphy, G. Qian, R. E. Rosholt, A New Error-free Floating-Point Summation Algorithm, *Computers and Mathematics with Applications*, **57**, 560–564, 2009.

[35] V. Y. Pan, G. Qian, Randomized Preprocessing of Homogeneous Linear Systems, *Linear Algebra and Its Applications*, **432**, 3272–3318, 2010.

[36] V. Y. Pan, G. Qian, A. Zheng, Advancing Matrix Computations with Randomized Preprocessing, in *Proc. of the Fifth International Computer Science Symposium in Russia (CSR 2010)*, Kazan, Russia, June 2010 , Farid Ablaev, Ernst W. Mayr (Eds.), *Lecture Notes in Computer Science (LNCS)*, pages 303-314, Springer, Berlin, 2010.

[37] V. Y. Pan, G. Qian, A. Zheng, Randomized Preconditioning versus Pivoting, Tech. Reports TRs 2009010 and 20100xx, *Ph.D. Program in Computer Science*, *Graduate Center, the City University of New York*, 2009 and 2010.

Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[38] V. Y. Pan, G. Qian, A. Zheng, Randomized Preconditioning of Linear Systems of Equations, Tech. Report TR 20100xx, *Ph.D. Program in Computer Science*, *Graduate Center, the City University of New York*, 2010.

Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[39] V. Y. Pan, G. Qian, A. Zheng, Z. Chen, Matrix Computations and Polynomial Root-finding with Preprocessing, *Linear Algebra and Its Applications*, in print.

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

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

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

[43] 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.

[44] W. F. Trench, A Note on a Toeplitz Inversion Formula, *Linear Algebra and Its Applications*, **29**, 55–61, 1990.

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

[46] 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.

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

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

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