2007

TR-2007003: Additive Preconditioning for Matrix Computations

Victor Y. Pan

Dmitriy Ivolgin

Brian Murphy

Rhys Eric Rosholt

Yuqing Tang

See next page for additional authors

Additive Preconditioning for Matrix Computations *

Victor Y. Pan[1,3], Dmitriy Ivolgin[2],
Brian Murphy[1], Rhys Eric Rosholt[1],
Yuqing Tang[2], and Xiaodong Yan[2]

[1] Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
firstname.lastname@lehman.cuny.edu

[2] Ph.D. Program in Computer Science
The City University of New York
New York, NY 10036 USA
firstnameinitiallastname@gc.cuny.edu

[3] http://comet.lehman.cuny.edu/vpan/

Abstract

Multiplicative preconditioning is a popular SVD-based techniques for the solution of linear systems of equations. Our novel SVD-free additive preconditioners are more readily available and better preserve matrix structure. We study their generation and their affect on conditioning of the input matrix. In other papers we combine additive preconditioning with aggregation and other relevant techniques to facilitate the solution of linear systems of equations and other fundamental matrix computations. Our analysis and experiments show the power of our algorithms, guide us in selecting most effective policies of preconditioning and aggregation, and provide some new insights into these and related subjects.

Key words: Matrix computations, Additive preconditioning

*Supported by PSC CUNY Awards 66437-0035 and 67297-0036

1 Introduction

1.1 Additive preconditioning: why and how?

Originally, preconditioning of a linear systems of equations $A\mathbf{y} = \mathbf{b}$ meant the transition to an equivalent but better conditioned linear systems $MA\mathbf{y} = M\mathbf{b}$, $AN\mathbf{x} = \mathbf{b}$, or more generally $MAN\mathbf{x} = M\mathbf{b}$ for $\mathbf{y} = N\mathbf{x}$ and readily computable nonsingular matrices M and/or N, called preconditioners (see [1]–[3] and the bibliography therein). Such systems can be solved faster and/or more accurately (see Appendix).

Multiplicative preconditioners M and N are closely linked to the Singular Value Decomposition (SVD) of the input matrix, namely to the costly computation of the smallest singular values and the associated singular vectors of an ill conditioned matrix. Furthermore, the SVD-based preconditioners can easily destroy matrix structure.

As an alternative or complementary tool, we propose *additive preprocessing* $A \leftarrow C = A + UV^H$, i.e., we add a matrix UV^H (having a smaller rank and/or structured) to the input matrix A to obtain its *additive modification* C with a smaller condition number. Here and hereafter M^H denotes the Hermitian (that is complex conjugate) transpose of a matrix M, which is just its transpose M^T for a real matrix M. Hereafter I_k is the $k \times k$ identity matrix, $\sigma_j(A)$ is the jth largest singular value of a matrix A, $\rho = \operatorname{rank} A$ is its rank, $||A|| = \sigma_1(A)$ is its 2-norm, and $\operatorname{cond} A = \sigma_1(A)/\sigma_\rho(A)$ is its condition number, and we use the abbreviations *MPPs*, *APPs*, *MPCs*, *APCs*, *A-modification*, and *M-* and *A-preconditioning* for multiplicative and additive preprocessors and preconditioners, additive modification, and multiplicative and additive preconditioning, respectively. We call an APP UV^H an APC if the ratio $\operatorname{cond} A / \operatorname{cond} C$ is large, and respectively call its generation A-preconditioning. We call an APP UV^H an *additive complement* or AC if the matrix C has full rank, whereas the matrix A does not.

In this paper we study A-preprocessing for general, sparse, and structured matrices, both theoretically and experimentally. In the Appendix we point out its applications to some fundamental matrix computations. In other papers we cover the latter subject in depth, presenting technical details and the results of numerical tests.

We stay with the original goal of decreasing the condition number of an input matrix and do not pursue the more recent alternative goal of compressing the spectrum of the singular values of an input matrix A into a smaller number of clusters to support faster convergence of the Conjugate Gradient and GMRES algorithms.

Given a nonsingular $n \times n$ matrix A, a positive integer $r < n$, and the *r-tail* (resp. *r-head*) of its SVD, that is the r smallest (resp. largest) singular values of the matrix A together with the associated singular spaces, one can immediately define an APC UV^H of a rank r and the A-modification $C = A + UV^H$ such that $\operatorname{cond} C = \sigma_1(A)/\sigma_{n-r}(A)$ (resp. $\operatorname{cond} C = \sigma_{r+1}(A)/\sigma_n(A)$). If both r-head and r-tail are known and if $2r < n$, one can readily obtain the optimal APCs UV^H

of a rank $r < n/2$, such that $\operatorname{cond} C = \sigma_{r+1}(A)/\sigma_{n-r}(A)$ (see Section 3.2). This can help even if we just approximate the r-tail and/or r-head because APCs are quite stable in their small-norm perturbation.

One can obtain the r-head and r-tail of the SVD by applying the Lanczos algorithm [4, Chapter 9], [5, Chapter 5], but we prefer some less costly ways to A-preconditioning, which also better preserve matrix structure. According to our analysis and extensive experiments, for a nonsingular $n \times n$ matrix A we are likely to arrive at an A-modification $C = A + UV^H$ with $\operatorname{cond} C$ of the order of $\sigma_1(A)/\sigma_{n-r}(A)$ as long as an APP UV^H of a rank r is

a) random (general, sparse, or structured),

b) well conditioned, and

c) properly scaled so that the ratio $||A||/||UV^H||$ is neither very large nor very small.

The decrease of the condition number of the matrix A is dramatic where $\sigma_{n-r}(A) \gg \sigma_n(A)$. In contrast, random M-preprocessing cannot help much against ill conditioning because $\operatorname{cond} A \le \prod_i \operatorname{cond} F_i$ if $A = \prod_i F_i$.

For computing APPs with properties b) and c), as well as at the other stages of A-preconditioning, we can apply the effective norm and condition estimators in [4, Section 3.5.4] and [6, Section 5.3].

With some additional techniques we can

- refine APCs UV^H of a rank r wherever $\operatorname{cond}(A+UV^H) \gg \sigma_1(A)/\sigma_{n-r}(A)$
- compress reasonably good APCs of a rank exceeding r into highly effective APCs of rank r and
- define dual APCs of a rank r whose associated dual A-modifications are expected to have condition numbers of the order of $\sigma_{r+1}(A)/\sigma_n(A)$.

In some applications we can generate the desired (e.g., sparse and/or structured) APCs by using neither SVD nor randomization. For example (see Acknowledgements), with a rank-one APC we can increase the absolute value of a small pivot entry in the Gaussian elimination and Cyclic Reduction algorithms without destroying matrix structure. Likewise, with rank-r APCs we can improve conditioning of $r \times r$ pivot blocks of block Gaussian elimination and block Cyclic Reduction.

1.2 The preceding study

Small-rank modification is a known tool for decreasing the rank of a matrix [7], [8], fixing its small-rank deviations from the Hermitian, positive definite, and displacement structures, and supporting the divide-and-conquer eigen-solvers [4], [5], [9], but these isolated efforts have not been identified as additive preconditioning in matrix computations. The discussions that followed the presentations by the first author at the International Conferences on the Matrix

Methods and Operator Equations in Moscow, Russia, in June 20–25, 2005, and on the Foundations of Computational Mathematics (FoCM'2005) in Santander, Spain, June 30–July 9, 2005, revealed only a few other touches to what we call A-preconditioning. They were sporadic and rudimentary versus our present work. We are aware of no earlier use of the nomenclature of A-preconditioning and APCs as well as of no attempts of devising and employing random and/or structured APCs, studying primal and dual APCs and their affect on conditioning systematically, applying them to a wide range of matrix computational problems, and relating them to aggregation and other fundamental techniques of matrix computations.

We introduced A-preconditioning to accelerate the steps of the inverse iteration for the algebraic eigenproblem [10]. (This occurred when we applied such an iteration to a semiseparable generalized companion matrix to compute the roots of a polynomial [11]. This was the first work that exploited semiseparable matrix structure for polynomial root-finding (cf. [12]–[14]).) While elaborating upon the application of APCs to eigen-solving, we also observed applications to the null space computations, constructed random and structured pseudo random APCs, estimated their affect on conditioning, defined various classes of primal and dual APCs, and devised our aggregation techniques for the transition from APCs to the solution of linear systems and computing eigensystems and determinants.

1.3 The contents and the organization of our paper

The study of A-preconditioning consists of two closely related areas, that is

- computing the APCs and
- employing them for facilitating the solution of the original problem of matrix computations.

In other papers we study the latter subject by applying aggregation and some other techniques of matrix computations. In the application of APCs to solving linear systems of equations and computing determinants we also use some advanced multiplication and summation algorithms in [15]. Hereafter we refer to them as *MSAs*. In this paper we cover the former subject of computing APCs as well as their affect on conditioning. Our analysis and experiments show the efficiency of our approach.

We study general case of $m \times n$ matrices rectangular input matrices, but on first reading one can assume dealing just with square matrices.

We organize our paper as follows. We begin with the definitions and preliminary results in Section 2. We define the SVD-based APCs in Section 3, but in Section 4 we comment on their deficiencies and generate SVD-free APCs, including structured APCs in Section 4.5. We study conditioning of the resulting A-modifications of the input matrix A theoretically in Section 5 and experimentally in Section 7. In Section 6 we briefly cover the advanced techniques of A-preconditioning for refining APCs and for generating dual APCs.

Our numerical tests have been designed by the first author and performed by his coauthors, mostly by D. Ivolgin, X. Yan, and Y. Tang. Otherwise this work as well as all typos and other errors are due to the first author.

Acknowledgements. E. E. Tyrtyshnikov, S. A. Goreinov, and N. L. Zamarashkin from the Institute of Numerical Analysis of the Russian Academy of Sciences in Moscow, Russia, and B. Mourrain from the INRIA in Sophia Antipolis, France, provided the first author of this paper with the access to the computer and library facilities during his visits to their Institutes in 2005/06. X. Wang was the first reader of our papers on A-preconditioning and responded with his original contribution [16]. Helpful and encouraging were the interest and comments to our work from the participants of the cited Conferences in Moscow and Santander (particularly from J. W. Demmel, G. H. Golub, V. Olshevsky, L. Reichel, M. Van Barel, and a participant of FoCM'2005 who proposed the substitution of APCs for pivoting in the Gaussian elimination algorithm).

2 Basic definitions and preliminaries

2.1 Some basic definitions for matrix computations

Most of our basic definitions reproduce or slightly modify the customary definitions in [4]–[6], [17]–[20] for matrix computations, in particular, for Hermitian, unitary, singular, full-rank and rank deficient matrices, the $k \times k$ identity matrices I_k, $k \times l$ matrices $0_{k,l}$ filled with zeros, the transpose A^T and the Hermitian transpose A^H of an $m \times n$ matrix A, its 2-norm $||A||$ and condition number $\operatorname{cond} A$, its range (that is the span of its column vectors), its (right) null space $N(A) = RN(A)$ and left null space $LN(A)$, rank $\rho = \operatorname{rank} A$, left nullity $\operatorname{lnul} A = m - \rho$, right nullity $\operatorname{rnul} A = n - \rho$, and nullity $\operatorname{nul} A = \min\{m, n\} - \rho$.

A matrix A is normalized if $||A|| = 1$.

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

$(B_1, \ldots, B_k)$ is the $1 \times k$ block matrix with the blocks $B_1, \ldots, B_k$.

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

We write $Q(M)$ for the Q-factor of the size $m \times n$ in the QR factorization of an $m \times n$ matrix M of the full rank.

2.2 The SVDs, conditioning, and the generalized inverse

Next we cover some concepts that are more closely related to preconditioning. The *Singular Value Decomposition* (hereafter *SVD*, also called the *full SVD*) of an $m \times n$ matrix A of a rank ρ is given by the equation $A = S\Sigma T^H$. Here $S = (S^{(\rho)}, S^{(\mathrm{nul})}) = (\mathbf{s}_j)_{j=1}^m$ and $T = (T^{(\rho)}, T^{(\mathrm{nul})}) = (\mathbf{t}_j)_{j=1}^n$ are square unitary

matrices; $S^{(\mathrm{nul})} = (\mathbf{s}_j)_{j=\rho+1}^{m}$ and $T^{(\mathrm{nul})} = (\mathbf{t}_j)_{j=\rho+1}^{n}$ are left and right unitary null matrix bases for the matrix A, respectively; $S^{(\rho)H}S^{(\rho)} = I_\rho$, $T^{(\rho)H}T^{(\rho)} = I_\rho$; $\Sigma = \mathrm{diag}(\Sigma^{(\rho)}, 0_{l,r})$ is the $m \times n$ matrix, $l = \mathrm{lnul}\, A = m - \rho$ and $r = \mathrm{rnul}\, A = n - \rho$ are the left and right nullity of the matrix A, respectively; $\Sigma^{(\rho)} = \mathrm{diag}(\sigma_j)_{j=1}^{\rho}$ is a diagonal matrix; $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\rho > 0$, $\sigma_j = 0$ for $j > \rho$, and we write $\sigma_j = +\infty$ for $j < 1$.

The scalars σ_j for $j \geq 1$ are the *singular values* of the matrix A. The vectors $\mathbf{s}_j$ for $j = 1, \ldots, m$ and $\mathbf{t}_j$ for $j = 1, \ldots, n$ are the associated left and right *singular vectors*, respectively, so that the null vectors are the singular vectors associated with the singular value zero. The decomposition $A = S^{(\rho)}\Sigma^{(\rho)}T^{(\rho)H} = \sum_{j=1}^{\rho} \sigma_j \mathbf{s}_j \mathbf{t}_j^H$ is the *compact SVD* of the matrix A.

Recall that $A\mathbf{t}_j = \sigma_j \mathbf{s}_j$ and $\mathbf{s}_j^H A = \sigma_j \mathbf{t}_j^H$ for all j, $A^H = T\Sigma^T S^H$, $A^H A = T\Sigma^T \Sigma T^H$, and $AA^H = S\Sigma\Sigma^T S^H$. Furthermore, $||A|| = \sigma_1$.

The *Moore-Penrose generalized inverse* of an $m \times n$ matrix A of a rank ρ (also called the *pseudo inverse*) is the matrix $A^- = \sum_{j=1}^{\rho} \sigma_j^{-1} \mathbf{t}_j \mathbf{s}_j^H$. We write A^- instead of the customary A^+ in [4], and we consistently write A^{-H} for $(A^H)^- = (A^-)^H$. We have $A^- = A^{-1}$ if $m = n = \rho$,

$$A^- = (A^H A)^{-1} A^H \quad \text{if} \quad m \geq n = \rho, \tag{2.1}$$

$$A^- = A^H (AA^H)^{-1} \quad \text{if} \quad m = \rho \leq n. \tag{2.2}$$

We write $n \gg d$ where the ratio n/d is large.

$\mathrm{cond}\, A = \sigma_1(A)/\sigma_\rho(A) = ||A||\, ||A^-||$ is the condition number of a matrix A of a rank ρ (under the 2-norm).

A matrix A of a rank ρ is *ill conditioned* if $\sigma_1 \gg \sigma_\rho$ and is *well conditioned* otherwise. A matrix A of any rank $\rho > 1$ can be ill conditioned, e.g., where $\sigma_1(A) \approx \sigma_j(A) \gg \sigma_{j+1}(A) \approx \sigma_\rho(A)$ for some j, $1 \leq j < \rho$, but for a larger rank ρ it can be ill conditioned even if $\sigma_j(A)/\sigma_{j+1}(A) \leq c$ for all j and a smaller bound $c > 1$. E.g., we can have $\mathrm{cond}\, A = 2^{100}$ for $c = 2$ and $\rho = 101$.

2.3 The g-heads, (extended) h-tails, and (g, h)-matrices

Preconditioning is linked to the parts of the SVD represented by its largest and its smallest singular values, and its effect is measured in terms of the ratios of singular values. This motivates our next definitions.

The *g-head*, *(g, h)-residue*, *h-tail*, and *extended h-tail* of the SVD (cf. Section 1.1) are the triples

$$(S^{(g)}, \Sigma^{(g)}, T^{(g)}),\ (S_{g,h}, \Sigma_{g,h}, T_{g,h}),\ (S_h, \Sigma_h, T_h),\ \text{and}\ (S_h^{(\mathrm{ext})}, \Sigma_h^{(\mathrm{ext})}, T_h^{(\mathrm{ext})}),$$

respectively, where g and $h < \rho - g$ are two nonnegative integers, $S^{(g)} = (\mathbf{s}_j)_{j=1}^{g}$,

$$S_{g,h} = (\mathbf{s}_j)_{j=g+1}^{\rho-h},\quad S_h = (\mathbf{s}_j)_{j=\rho-h+1}^{\rho},\ S_h^{(\mathrm{ext})} = (S_h, S^{(nul)}) = (\mathbf{s}_j)_{j=\rho-h+1}^{m},$$

$$T^{(g)} = (\mathbf{t}_j)_{j=1}^{g},\quad T_{g,h} = (\mathbf{t}_j)_{j=g+1}^{\rho-h},\quad T_h = (\mathbf{t}_j)_{j=\rho-h+1}^{\rho},$$

$$T_h^{(\mathrm{ext})} = (T_h, T^{(nul)}) = (\mathbf{t}_j)_{j=\rho-h+1}^{n},$$
$$\Sigma^{(g)} = \mathrm{diag}(\sigma_j)_{j=1}^{g}, \Sigma_{g,h} = \mathrm{diag}(\sigma_j)_{j=g+1}^{\rho-h},$$
$$\Sigma_h = \mathrm{diag}(\sigma_j)_{j=\rho-h+1}^{\rho}, \quad \Sigma_h^{(\mathrm{ext})} = \mathrm{diag}(\Sigma_h, 0_{l,r}),$$

and $M^{(0)}$ and M_0 are empty matrices for M denoting S, Σ, or T.

A matrix A of a rank ρ is a (g,h) *matrix* if its (g,h)-residue is well conditioned, whereas its $(g+1)$-head and $(h+1)$-tail are ill conditioned or equivalently if $\sigma_1 \gg \sigma_{g+1}$ and $\sigma_{\rho-h+1} \gg \sigma_\rho$, whereas the ratio $\sigma_{g+1}/\sigma_{\rho-h+1}$ is not large. A matrix is a *strictly* (g,h) *matrix* if it is a (g,h) matrix and if its g-head and h-tail are well conditioned, whereas $\sigma_g \gg \sigma_{g+1}$ and $\sigma_{\rho-h} \gg \sigma_{\rho-h+1}$ or equivalently if the ratios σ_1/σ_g, $\sigma_{g+1}/\sigma_{\rho-h+1}$, and $\sigma_{\rho-h}/\sigma_\rho$ are not large, whereas

$$\sigma_g \gg \sigma_{g+1} \quad \text{and} \quad \sigma_{\rho-h+1} \gg \sigma_{\rho-h}.$$

In this paper the (g,h)-matrices are usually the $(g,0)$ or $(0,h)$ matrices.

A strictly $(0,h)$ matrix A of a rank ρ has exactly h small positive singular values. We call h its *numerical nullity* and $\rho - h$ its *numerical rank* and write $h = \mathrm{nnul}\, A$, $\rho - h = \mathrm{nrank}\, A$.

3 SVD-based APPs

3.1 Null-based ACs and SVD-based APCs

We begin with two straightforward algorithms for computing ACs and APCs provided we know some bases for the respective null spaces and singular spaces.

Algorithm 3.1. Computing a null-based AC.

INPUT: *an $m \times n$ matrix A of a rank ρ and the left and right unitary null matrix bases $S^{(null)} = (\mathbf{s}_j)_{j=\rho+1}^{m}$ and $T^{(null)} = (\mathbf{t}_j)_{j=\rho+1}^{n}$ of the matrix A.*

OUTPUT: *two matrices U and V and the matrix of full rank $u = \min\{m,n\}$,*

$$C = A + UV^H = \sum_{j=1}^{\rho} \sigma_j \mathbf{s}_j \mathbf{t}_j^H + \sigma \sum_{j=\rho+1}^{u} \mathbf{s}_j \mathbf{t}_j^H. \tag{3.1}$$

COMPUTATIONS:

Fix a positive σ and compute and output a pair of matrices

$$U = (\sigma \mathbf{s}_j)_j, \ V = (\mathbf{t}_j)_j \quad \text{for} \quad j = \rho+1, \dots, u. \tag{3.2}$$

Algorithm 3.2. Computing an SVD-based APC.

INPUT: *an $m \times n$ matrix A of a rank ρ, two nonnegative integers g and h such that $g + h \le \rho$, the g-head and the h-tail of the compact SVD $A = S^{(\rho)}\Sigma^{(\rho)}T^{(\rho)H}$ (that is the g first and the h last columns of the matrices $S^{(\rho)}$, $\Sigma^{(\rho)}$, and $T^{(\rho)H}$), and a positive σ in the range $[\sigma_{g+1}, \sigma_{\rho-h}]$.*

OUTPUT: *a pair of matrices U of size $m \times r$ and V of size $n \times r$ such that*

$$\operatorname{cond}(A + UV^H) = \max\{\sigma, \sigma_{g+1}\}/\min\{\sigma, \sigma_{\rho-h}\}. \tag{3.3}$$

Here $r = g + h$ for $\sigma_g > \sigma > \sigma_{\rho-h+1}$, $r = g + h - 1$ for $\sigma = \sigma_g > \sigma_{\rho-h+1}$ and for $\sigma_g > \sigma = \sigma_{\rho-h+1}$, and $r = \rho - h - 2$ for $\sigma = \sigma_g = \sigma_{\rho-h+1}$.

COMPUTATIONS:

Compute and output a pair of matrices U of size $m \times r$ and V of size $n \times r$ such that

$$UV^H = \sum_{j=1}^{g} (\sigma - \sigma_j)\mathbf{s}_j\mathbf{t}_j^H + \sum_{j=\rho-h+1}^{\rho} (\sigma - \sigma_j)\mathbf{s}_j\mathbf{t}_j^H,$$

$$U = ((\sigma - \sigma_j)\mathbf{s}_j)_j,\ V = (\mathbf{t}_j)_j \quad \text{for} \quad j = 1, \dots, g; \rho - h + 1, \dots, \rho. \tag{3.4}$$

Observe that

$$C = A + UV^H = \sum_{j=g+1}^{\rho-h} \sigma_j\mathbf{s}_j\mathbf{t}_j^H + \sigma\widehat{\sum}_j \mathbf{s}_j\mathbf{t}_j^H \tag{3.5}$$

where the symbol $\widehat{\sum}_j$ stands for a sum over j ranging from one to g and from $\rho - h + 1$ to ρ.

3.2 APCs of the optimal rank

It is quite surprising that the same decrease of $\operatorname{cond} A$ as with Algorithm 3.2 for $r = g + h$ can be always achieved with an APC of rank $\max\{g, h\}$, and this is the optimum decrease of the rank [16]. To compute such an APC, first bring the input matrix $A = S^H\Sigma T$ to the diagonal form Σ and then recursively apply the following result [16].

Theorem 3.1. *For any numbers $a_1 \geq b_1 \geq b_2 \geq a_2 > 0$, there exist real numbers u and v such that the 2×2 matrix*

$$\begin{pmatrix} a_1 - u^2 & -uv \\ -uv & a_2 - v^2 \end{pmatrix}$$

has singular values b_1 and b_2.

This APC is Hermitian and/or real if so is the input matrix.

Remark 3.1. *Theorem 3.1 also supports computing an APC of a rank r that collapses $2r+1$ distinct singular values given with the associated singular vectors into a single median value.*

4 SVD-free ACs and APCs

4.1 SVD-free APPs versus SVD-based APPs and MPPs

We run into two problems with the SVD-based preconditioning.

a) Realistically, approximations to the g-heads of the SVDs of an input matrix A for smaller positive g are readily available, but the task is more costly for the h-tails of the SVDs [4, Sections 9.1 and 9.2], [5, pages 366 and 367], [20].

b) The SVD does not preserve matrix structure and sparseness.

We avoid both problems by using SVD-free low-cost random APPs.

4.2 Error-free A-preconditioning

Given a $(g, 0)$ matrix A and its APC UV^H, the rounding errors of computing the A-modification $C = A + UV^H$ are of the order of $\epsilon||A|| = \epsilon\sigma_1(A)$ where ϵ is the unit roundoff (the machine epsilon). They are large relatively to the output norm $\tilde{\sigma} = ||A + UV^H|| = \max\{\sigma, \sigma_{g+1}\}$ if $\sigma_1 \gg \tilde{\sigma}$ and can ruin the effect of preconditioning. No such a problem arises for APCs for $(0, h)$ matrices A.

Even for $(g, 0)$ matrices, however, we can compute the APC error-free by filling the generators U and V with shorter numbers and/or applying the MSAs in [15]. (Recall that the small-norm perturbations of the generators caused by the truncation of their entries keep the matrix C well conditioned.)

4.3 Randomized ACs

Our next theorem links the ranks of a random APP UV^H and of the A-modification $C = A + UV^H$ of an $m \times n$ matrix A. We first sketch the main claims of this theorem by writing $u = \min\{m, n\}$ and "$\Longrightarrow$" for "implies" and then state and prove it formally.

$$\{\text{rank}\, C = u\} \Longrightarrow \{r \geq \text{nul}\, A\},$$

$$\{r \geq \text{nul}\, A \text{ for random unitary } U \text{ and } V\} \Longrightarrow \{\text{rank}\, C = u \text{ (likely)}\}.$$

We recall some basic definitions and a basic result for randomized algebraic computations.

Random sampling of elements from a finite set Δ is their selection from the set Δ at random, independently of each other, and under the uniform probability distribution on Δ. A matrix is *random* if its entries are randomly sampled (from a fixed finite set Δ).

An $k \times l$ *random unitary* matrix is the $k \times l$ Q-factor $Q(M)$ in the QR factorization of random $k \times l$ matrix M of the full rank.

Lemma 4.1. *[21] (cf. also [22], [23]). For a finite set Δ of cardinality $|\Delta|$, let a polynomial in m variables have total degree d, let it not vanish identically on the set Δ^m, and let the values of its variables be randomly sampled from the set Δ. Then the polynomial vanishes with a probability of at most $d/|\Delta|$.*

Theorem 4.1. *For a finite set Δ of cardinality $|\Delta|$ in a ring $\mathbb{R}$, $u = \min\{m, n\}$, and four matrices $A \in \mathbb{R}^{m\times n}$ of a rank ρ, $U \in \Delta^{m\times r}$, $V^T \in \Delta^{r\times n}$, and $C = A + UV^T$, we have*

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

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

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

Proof. Part a) is verified immediately. Now let $r+\rho \ge u$, let a $u\times u$ submatrix A_u of the matrix A have rank ρ, and let $C_u = A_u + (UV^T)_u$ denote the respective $u \times u$ submatrix of the matrix C. Then clearly, $\operatorname{rank} C = \operatorname{rank} C_u = u$ if $U = V$ and if the entries of the matrix U are indeterminates. Since $\det C_u$ is a polynomial of a total degree of at most $2(u - \rho) \le 2r$ in these entries, part b) follows from Lemma 4.1. Part c) is proved similarly to part b). □

We can recursively generate random APPs UV^H of ranks $r = 0, 1, \dots$ and stop where we either arrive at an AC or exceed a fixed upper bound $r(+)$ on r. Foreseeing an extension to numerical computation of APCs, we choose random unitary matrices U and V for a normalized matrix A.

Algorithm 4.1. Randomized computation of a unitary AC.

INPUT: *a normalized $m \times n$ matrix A of an unknown rank $\rho \le u = \min\{m, n\}$, an integer $r(+) \ge u - \rho$, and a black box Subroutine FULL RANK that tests if a matrix has full rank.*

OUTPUT: *FAILURE or an integer r such that $u - \rho \le r \le r(+)$ and a pair of unitary matrices U of size $m \times r$ and V of size $n \times r$ such that the matrix $C = A + UV^H$ has full rank u.*

INITIALIZATION:

Set $r \leftarrow 0$, $U \leftarrow \emptyset^{m\times 0}$, and $V \leftarrow \emptyset^{n\times 0}$ where $\emptyset^{i\times 0}$ is the empty $i \times 0$ matrix.

COMPUTATIONS:

1. If r exceeds $r(+)$, stop and output FAILURE.

2. Otherwise apply Subroutine FULL RANK to test if the A-modification $C = A + UV^H$ has full rank. If so, output the integer r and the matrices U and V and stop.

3. Otherwise sample two normalized random vectors $\mathbf{u}$ and $\mathbf{v}$ of dimension n such that $U^H\mathbf{u} = \mathbf{0}$ and $V^H\mathbf{v} = \mathbf{0}$ (unless $r = 0$), set $r \leftarrow r + 1$, $U \leftarrow (U, \mathbf{u})$, and $V \leftarrow (V, \mathbf{v})$, and go to Stage 1.

In virtue of Theorem 4.1, Algorithm 4.1 is correct and is likely to output $r = u - \rho = \operatorname{nul} A$.

Remark 4.1. *The latter feature enables uncertified randomized computation of the rank and the nullity of a matrix. Observe that $r = \operatorname{rnul} A$ if and only if $AC^{-}U = 0$ provided U is a full rank matrix.*

The most costly stage of the algorithm is the applications of the Subroutine FULL RANK to the matrices C. We expect to apply the Subroutine $u-\rho$ times when we proceed as above but at most $2\lceil \log_2(u-\rho)\rceil$ times if we incorporate the binary search for the nullity.

4.4 Extension to computing APCs and their consistent scaling

To implement Algorithm 4.1 numerically, one should just replace Subroutine FULL·RANK with estimating whether $\operatorname{cond} C$ exceeds a fixed tolerance bound. Applied to a $(0, h)$ matrix A, the resulting algorithm either fails (in some pathological cases) or outputs a well conditioned A-modification $C = A+UV^H$. Here is a sketch of an extension of Theorem 4.1 that supports the algorithm,

$$\{\operatorname{nrank} C = u\} \Longrightarrow \{r \geq \operatorname{nnul} A\},$$

$$\{r \geq \operatorname{nnul} A \text{ and random unitary } U \text{ and } V\} \Longrightarrow \{\operatorname{nrank} C = u \text{ (likely)}\}.$$

Suitably scaled and well conditioned, rather than unitary, random APPs can be more readily consistent with a desired matrix structure and are still likely to be APCs according to our analysis in Section 5 and the test results in Section 7.

Let us specify suitable scaling. An APP UV^H is *scaled consistently* with a $(0, h)$ matrix A, and the pair of A and UV^H is scaled consistently, if the ratio $||A||/||UV^H||$ is neither large nor small. We can scale APPs by the powers of two to avoid rounding errors.

4.5 Structured and sparse APPs

All APPs of small ranks are structured, but next we supply various examples of sparse and/or structured APPs of any rank. In our extensive tests, these APPs were typically APCs for all classes of tested input matrices. Hereafter we call these APPs *pseudo random*. More examples of them are expected and welcome from the readers.

Example 4.1. Circulant APPs. *$UV^H = F^{-1}D_rF$ for the $n \times n$ unitary matrix $F = \frac{1}{\sqrt{n}}(\exp\frac{2\pi ij\sqrt{-1}}{n})_{i,j=0}^{n-1}$ of the discrete Fourier transform at the n-th roots of unity and for the $n \times n$ diagonal matrix $D_r = \mathrm{diag}(d_i)_{i=0}^{n-1}$ that has exactly r nonzero entries fixed or sampled at random in a fixed set $\mathbb{S}$ and placed at r fixed or random positions on the diagonal. Such an APP UV^H is a circulant matrix of the rank r that has the first column $F^{-1}\mathbf{d}$ for $\mathbf{d} = (d_i)_{i=0}^{n-1}$ (cf., e.g., [24, Theorem 2.6.4]). It is sufficient to perform $O(n\max\{r, \log n\})$ ops to multiply it by a vector. The bound decreases to $O(n\log r)$ where the r nonzeros occupy r successive positions on the diagonal. If $\mathbb{S}$ is a real set, then the APP is Hermitian. If the set $\mathbb{S}$ lies in the annulus $\{x:\ d_- \le |x| \le d_+\}$, then $\mathrm{cond}(UV^H) = \mathrm{cond}\, D_r \le d_+/d_-$.*

Example 4.2. f-circulant APPs *[24, Section 2.6]. In the previous example replace the matrix F with the matrix FD_- where $D_- = \mathrm{diag}(g^i)_{i=0}^{n-1}$ and g is a primitive n-th root of a nonzero scalar f. In this case the APP is f-circulant. (It is circulant for $f = 1$ and skew-circulant for $f = -1$.) As in the previous example, one can readily bound the condition number of the APP and the arithmetic cost of its multiplication by a vector.*

Example 4.3. Hermitian Toeplitz-like APPs I. *Define a Hermitian and nonnegative definite Toeplitz-like APP UU^H for an $n \times r$ well conditioned Toeplitz matrix U of full rank (we assume the definitions of Toeplitz-like matrices in [24]). Either fix such a matrix or define it by varying u random parameters for a nonnegative integer $u < n + r$ until you yield well conditioning. Output FAILURE if this does not work. The APP has a rank of at most r and a displacement rank of at most four and can be multiplied by a vector in $O(n\log r)$ ops.*

Example 4.4. Structured or sparse Hermitian APPs I. *Define an APP UU^H where $U = PW$, P is a fixed or random $n \times n$ permutation matrix (in the simplest case $P = I_n$) and W is a fixed or random $n \times r$ block of the $n \times n$ matrix of the discrete Fourier, sine or cosine transform [24, Section 3.11], or of another well conditioned matrix with a fixed structure such as the sparseness structure [25], [26], the displacement structure of Toeplitz, Hankel, Vandermonde, or Cauchy types (cf. [24] and the bibliography therein), or the semi-separable (rank) structure (cf. the bibliography in [27]). We can apply random diagonal scaling to sparse and semiseparable matrices. Example 4.3 is the special case where $P = I_n$ and W is a Toeplitz matrix. The complexity bounds for multiplication by a vector are extended with the adjustment to the structure of the APP.*

Example 4.5. Hermitian Toeplitz-like APPs II. *Define a Hermitian and nonnegative definite Toeplitz-like APP UU^H for an $n \times r$ Toeplitz matrix $U = (T_1, 0_{r,n_1}, \dots, T_k, 0_{r,n_k})^T$. Here T_i are $r \times r$ Toeplitz matrices, $0_{r,n_i}$ are $r \times n_i$ matrices filled with zeros for $i = 1, \dots, k$, and $k, n_1, \dots, n_k$ are positive integers (fixed or random) such that $kr + n_1 + \cdots + n_k = n$. Fix or choose at random the Toeplitz matrices T_i such that the resulting matrix U is well conditioned.*

T_i can denote general Toeplitz matrices or special, e.g., circulant, f-circulant, triangular Toeplitz or banded Toeplitz matrices. For general Toeplitz matrices $T_1, \dots, T_k$, the APP has a displacement rank of at most $2k \le 2\lfloor n/r \rfloor$ and can be multiplied by a vector in $O(kr \log r)$ flops. For banded Toeplitz matrices T_i with a constant bandwidth we only need $O(kr)$ flops to multiply the APP by a vector. For $T_i = c_i I_r$ the matrix U has orthogonal columns, and we make it unitary by choosing the scalars $c_1, \dots, c_k$ such that $c_1^2 + \cdots + c_k^2 = 1$.

Example 4.6. Structured or sparse Hermitian APPs II. *Define a well conditioned APP UU^H where $U = P(T_1, 0_{r,n_1}, \dots, T_k, 0_{r,n_k})^T$ for an $n \times n$ permutation matrix P and integers $k, n_1, \dots, n_k$ chosen as in Example 4.5 but for all i let T_i be $r \times r$ fixed or random structured matrices, e.g., the matrices of the discrete Fourier, sign or cosine transforms, matrices with a fixed displacement structure, semi-separable (rank structured) matrices, or sparse matrices with fixed patterns of sparseness (see the bibliography listed in Example 4.4). Example 4.5 is the special case where $P = I_n$ and T_i are Toeplitz matrices.*

5 APPs and conditioning

In this section we assume a square matrix A. Part of our study extends to its rectangular submatrices and matrices embedding it.

First we consider a normalized singular well conditioned matrix A with nullity r and a random unitary APP UV^H of rank r and show that the A-modification $C = A + UV^H$ is expected to be nonsingular and well conditioned. Then we extend this result to $(0, r)$ matrices lying near such a matrix A and to a consistently scaled random and well conditioned APP UV^H of rank r.

In [16] this result is rederived directly for such a neighbourhood of a rank deficient matrix and is further refined in the Hermitian and Hermitian positive definite cases. The results in the latter cases can be viewed as a quantitative complement to the interlacing property of the eigenvalues of the Hermitian matrices A and $C = A + UU^H$ [4, Theorem 8.5.3], [9, Section 3.2.1], [28], [29].

5.1 ACs and conditioning

We first factorize the A-modification C.

Theorem 5.1. *Let A be an $n \times n$ matrix of rank $\rho = n - r$, so that $r = \operatorname{nul} A$. Let U and V be $n \times r$ matrices such that the $n \times n$ matrix $C = A + UV^H$ is nonsingular. Let $A = S\Sigma T^H$ be the SVD of the matrix A, where the matrices S^H and T^H are unitary, $\Sigma = \operatorname{diag}(\Sigma_A, 0_r)$, and $\Sigma_A = \operatorname{diag}(\sigma_j)_{j=1}^{\rho}$ is the diagonal matrix of the singular values of the matrix A. Write*

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

where the matrices U_r and V_r have size $r \times r$. Then

a) $C = SR_U \operatorname{diag}(\Sigma_A, I_r) R_V^H T^H$ and

b) the matrices R_U, R_V, U_r, and V_r are nonsingular.

Proof. Write $\tilde{C} = \Sigma + S^H U V^H T$. Observe that $C = A + UV^H = S\Sigma T^H + SS^H UV^H TT^H = S\tilde{C}T^H$, $R_U \Sigma R_V^H = \Sigma$, $S^H U = R_U \begin{pmatrix} 0 \\ I_r \end{pmatrix}$, and $T^H V = R_V \begin{pmatrix} 0 \\ I_r \end{pmatrix}$. Deduce that $\tilde{C} = R_U \Sigma R_V^H + R_U \operatorname{diag}(0, I_r) R_V^H = R_U \operatorname{diag}(\Sigma_A, I_r) R_V^H$. Substitute this expression into the equation $C = S\tilde{C}T^H$ to arrive at part a). Part b) follows because the matrix C is nonsingular. □

Corollary 5.1. *Under the assumptions of Theorem 5.1 we have*

$$\frac{||\operatorname{diag}(\Sigma_A, I_r)||}{||R_U^{-1}||\ ||R_V^{-1}||} \le ||C|| \le ||\operatorname{diag}(\Sigma_A, I_r)||\ ||R_U||\ ||R_V||,$$

$$\frac{||\operatorname{diag}(\Sigma_A^{-1}, I_r)||}{||R_U||\ ||R_V||} \le ||C^{-1}|| \le ||\operatorname{diag}(\Sigma_A^{-1}, I_r)||\ ||R_U^{-1}||\ ||R_V^{-1}||,$$

so that

$$\frac{\operatorname{cond}\operatorname{diag}(\Sigma_A, I_r)}{(\operatorname{cond} R_U)\operatorname{cond} R_V} \le \operatorname{cond} C \le (\operatorname{cond} R_U)(\operatorname{cond} R_V)\operatorname{cond}\operatorname{diag}(\Sigma_A, I_r).$$

Proof. The corollary follows from Theorem 5.1 because $||S|| = ||S^{-1}|| = ||T|| = ||T^{-1}|| = 1$, $\operatorname{cond} M = ||M||\ ||M^-||$ and $||M^H|| = ||M||$ for any matrix M. □

Let us specify the estimate for $\operatorname{cond} C$ provided the matrices U and V are unitary and the matrix A is scaled properly.

Theorem 5.2. *If the matrices U and V are unitary, then we have*

$$||R_U|| \le \sqrt{2},\ ||R_V|| \le \sqrt{2},$$

$$||R_U^{-1}|| \le 1 + \sqrt{2}||U_r^{-1}||,\ ||R_V^{-H}|| = ||R_V^{-1}|| \le 1 + \sqrt{2}||V_r^{-1}||.$$

Proof. The first two bounds of the theorem follow because $R_U = (I_{n,\rho}, S^H U)$, $R_V = (I_{n,\rho}, T^H V)$ where $I_{n,\rho} = \begin{pmatrix} I_\rho \\ 0 \end{pmatrix}$ and because $||(X, Y)|| = ||(X, Y)^H|| \le \sqrt{2}$ for a pair of matrices X and Y with the 2-norms of at most one.

To deduce the two other bounds, observe that

$$R_U = \operatorname{diag}(I_\rho, 0) + (0, S^H U)\operatorname{diag}(0, I_r),$$

$$R_V = \operatorname{diag}(I_\rho, 0) + (0, T^H V)\operatorname{diag}(0, I_r),$$

$$R_U^{-1} = \operatorname{diag}(I_\rho, 0) + \begin{pmatrix} 0 & -U_\rho \\ 0 & I_r \end{pmatrix} \operatorname{diag}(0, U_r^{-1}),$$

$$R_V^{-H} = \operatorname{diag}(I_\rho, 0) + \begin{pmatrix} 0 & -V_\rho \\ 0 & I_r \end{pmatrix} \operatorname{diag}(0, V_r^{-1}),$$

the 2-norms of the matrices S, T, U, and V are equal to one, $||U_\rho|| \le ||U|| = 1$ and $||V_\rho|| \le ||V|| = 1$. □

Theorem 5.3. *Under the assumptions of Theorem 5.1, suppose that*

$$\sigma_{n-r} \le 1 \le \sigma_1. \tag{5.1}$$

Then $||\operatorname{diag}(\Sigma_A, I_r)|| = ||A||$ *and* $||(\operatorname{diag}(\Sigma_A, I_r))^{-1}|| = \sigma_{n-r}$.

Proof. Immediate by inspection. □

Corollary 5.2. *Write* $p_r = ||R_U^{-1}||\ ||R_V^{-1}|| \le (1+\sqrt{2}||U_r^{-1}||)(1+\sqrt{2}||V_r^{-1}||)$. *Under bounds (5.1) and the assumptions of Theorem 5.1, suppose the matrices* U *and* V *are unitary. Then*

a) $||A||/p_r \le ||C|| \le 1 + ||A|| \le 2||A||$,

b) $1/(2\sigma_{n-r}) \le ||C^{-1}|| \le p_r/\sigma_{n-r}$, *and*

c) $(\operatorname{cond} A)/(2p_r) \le \operatorname{cond} C \le 2p_r \operatorname{cond} A$.

Proof. Part a) follows immediately from part a) of Theorem 5.1 because $||C|| \le ||A|| + ||UV^H||$ and $||UV^H|| = 1 \le \sigma_1 = ||A||$.

To prove part b), combine Corollary 5.1 with Theorems 5.2 and 5.3.

Part c) immediately follows from parts a) and b). □

5.2 Small-norm perturbation of an AC and conditioning

Any ill conditioned matrix $\tilde{A}$ is a small-norm perturbations of a singular matrix $A = \tilde{A} - E$. To extend Corollary 5.2 to ill conditioned matrices $\tilde{A}$, it remains to bound the estimates in the corollary in a small-norm perturbation $A \to \tilde{A}$.

Theorem 5.4. *Under the assumptions of Corollary 5.2, let the matrix* $\tilde{C} = C + E$ *be nonsingular. Write* $\delta = ||E||$, *and* $\delta_C = \delta||C^{-1}||$. *Then we have*

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

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

c) if the matrices C *and* E *are Hermitian and nonnegative definite, then* $||\tilde{C}^{-1}|| \le ||C^{-1}||$, *so that* $\operatorname{cond} \tilde{C} \le (1 + \delta/||C||) \operatorname{cond} C$.

Proof. Part a) is proved immediately, similarly to part a) of Corollary 5.2.

To prove part b), apply the SMW formula of Sherman, Morrison, and Woodbury [4, page 50], [6, Corollary 4.3.2] to the matrices $C + UV^H = \tilde{C}$, $U = -E$, and $V = I_n$ and obtain that $\tilde{C}^{-1} = C^{-1}(I_n + E(I_n - C^{-1}E)^{-1}C^{-1})$. Part b) follows from this equation and the assumption that $\delta_C = \delta||C^{-1}|| < 1$.

Part c) follows because the matrix $E = \tilde{C} - C$ is nonnegative definite and the matrix C is positive definite. □

5.3 The impact of scaling APPs

Under the assumptions of Corollary 5.2, the ratio $\frac{\operatorname{cond} C}{\operatorname{cond} A}$ is not large unless the product $p_r = ||U_r^{-1}||\ ||V_r^{-1}||$ is large. Moreover, if we relax assumption (5.1), assume that $\sigma_1 = 1/\theta \ge \sigma_{n-r}$ or $\sigma_1 \ge \sigma_{n-r} = \theta$ for $\theta > 1$, and ignore the resulting dynamics in the factors of $||U_r^{-1}||$ and $||V_r^{-1}||$, then the upper

estimate $2p_r$ on this ratio in Corollary 5.2c would increase by the factor of θ. In our extensive tests, the value of θ tended to be nicely bounded for random, well conditioned, and consistently scaled APPs.

For random unitary $n \times r$ matrices U and V, we can also view the $n \times r$ unitary matrices $S^H U$ and $T^H V$ in Theorem 5.1 as random, so that the norms of the inverses of their $r \times r$ southern-most submatrices U_r and V_r are likely to be reasonably bounded for smaller r. (If these norms are large, then $\operatorname{cond} C$ is large, and if we detect this, we can resample the random matrices U and V.)

To define consistent scaling we must estimate the 2-norms of the matrix A and the APP UV^H. The effective algorithms in [6, Section 5.3.3] compute quite tight bounds on both values $\sigma_1(A) = ||A||$ and $\sigma_\rho(A) = 1/||A^-||$. We also recall the following cruder bounds in [4, Section 2.3.2 and Corollary 2.3.2],

$$1 \le ||A||/\max_{i,j} |a_{i,j}| \le \sqrt{mn},$$

$$1 \le \sqrt{||A||_1 ||A||_\infty}/||A|| \le (mn)^{1/4}.$$

6 Advanced A-preconditioning

6.1 Improving APCs

Suppose for an ill conditioned matrix A, an APC UV^H of a rank r has turned out to be too crude, $\operatorname{cond} C \gg \sigma_1/\sigma_{\rho-r}$ for the A-modification $C = A + UV^H$. Then the following transform (where we use the notation $Q(M)$ in Section 2.1) serves as a remedy, according to our extensive tests (cf. Table 7.2) and the analysis in [16] and [30],

$$(U \leftarrow Q(C^- U), \quad V \leftarrow Q(C^{-H} V)). \tag{6.1}$$

The transform (6.1) can be extended to the compression of the APCs of larger ranks. We can generate effective APCs of larger ranks more readily, and then yield effective APCs of smaller ranks by combining such an inflation with the subsequent compression as follows.

1. *(Generation of an inflated APC.)* Select an integer $h > r$, e.g., $h = 2r$, and generate an APC UV^H of rank h.

2. Compute two suitably scaled and well conditioned matrix bases $T(U)$ and $T(V)$ for the right singular spaces in the extended r-tails of the matrices $AC^- U$ and $A^H C^{-H} V$, respectively.

3. *(Compression.)* Compute and output the new generators $U \leftarrow Q(C^- U T(U))$ and $V \leftarrow Q(C^{-H} V T(V))$ and the new APC UV^H.

X. Wang in [16] has applied a similar algorithm to 10×10 Hilbert input matrices $A = (\frac{1}{i+j-1})_{i,j=1}^{10}$ and has consistently arrived at $\operatorname{cond} C \approx \frac{\sigma_1(A)}{\sigma_{10-h}(A)}$ in his extensive tests for various choices of positive $h \le 10$ and $r < h$.

6.2 Dual A-preprocessing

Given a smaller positive integer g and a $(g, 0)$ matrix A of the full rank, we can readily compute the g-head of its SVD and then apply Algorithm 3.2 to compute an effective APC of rank g (cf. Section 4.2). For larger g, however, computing the g-head becomes a problem, which we can avoid by applying dual A-preprocessing, that is by applying A-preprocessing to the matrix A^- without computing this matrix. Namely, we represent the dual A-modification $C_- = A^- + VU^H$ by its (generalized) inverse

$$(C_-)^- = (A^- + VU^H)^- = A - AVH^-U^HA, \quad H = I_q + U^HAV. \tag{6.2}$$

We call this equation *the dual SMW formula*.

Having the matrix $(C_-)^-$ available, we can compute the vector $\mathbf{y} = A^-\mathbf{b}$ as follows,

$$\mathbf{y} = A^-\mathbf{b} = \mathbf{z} - VU^H\mathbf{b}, \quad (C_-)^-\mathbf{z} = \mathbf{b}.$$

We readily extend the results of our analysis from the primal to dual A-preprocessing. In particular, the matrix $(C_-)^-$ is likely to be well conditioned if an APC VU^H of a sufficiently large rank satisfies requirements a) and b) in Section 1.1 together with the following counterpart of requirement c),

d) the ratio $||A^-||/||VU^H||$ is neither large nor small.

Furthermore, we can choose a dual APC that preserves the structure of the input matrix.

Finally, here is a natural extension of our policy (6.1) to dual APCs VU^H,

$$V \leftarrow Q((C_-)^-V), \quad U \leftarrow Q((C_-)^{-H}U)).$$

7 Numerical tests for generating APCs

In our tests we generated singular and ill-conditioned nonsingular matrices of 16 classes, modified them with random and pseudo random APPs of eight classes, and computed the condition numbers of the input and modified matrices. We run such tests for over 100,000 input instances and observed quite similar statistics for all selected classes of input matrices A and APPs. Moreover, the test results varied little with the matrix size.

In all tests we used the following CPU and memory configuration, operating system, mathematical application software, and random number generator.

CPU	AMD Athlon XP 2800+ 2.09GHZ
Memory	512MB
OS	Microsoft Windows XP Professional Version 2002 Service Pack 2
Platform	Matlab Version 7.0.0.19920(R14)
Random Number Generator	Matlab Statistics Toolbox's Uniform Distribution

Unless we specify otherwise, we sampled the entries of random matrices in the closed line interval $[-1, 1]$.

We display sample data in Tables 7.1 and 7.2.

Dealing with real (in particular integer or rational) matrices, we use the nomenclatures "orthogonal", "symmetric", and "nonsymmetric" rather than "unitary", "Hermitian", and "non-Hermitian" (cf. [4], [6]).

Throughout this section we assign the values $n = 100$ and $\nu = 1, 2, 4, 8$ to the parameters n and ν.

7.1 Generation of singular input matrices A

In our tests we used the following real singular input matrices A with $\operatorname{nul} A = \nu$ for $\nu = 1, 2, 4, 8$. ("s" is our abbreviation for "symmetric" and "n" for "nonsymmetric".)

1n. *Nonsymmetric matrices of type I with nullity ν.* $A = G\Sigma_\nu H^T$ are $n \times n$ matrices where G and H are $n \times n$ random orthogonal matrices, the Q-factors in the QR factorizations of random real matrices; $\Sigma_\nu = \operatorname{diag}(\sigma_j)_{j=1}^n$ is the diagonal matrix filled with zeros and the singular values of the matrix A such that $\sigma_{j+1} \le \sigma_j$ for $j = 1, \dots, n-1$, $\sigma_1 = 1$, the values $\sigma_2, \dots, \sigma_{n-\nu-1}$ are randomly sampled in the semi-open interval $[0.1, 1)$, $\sigma_{n-\nu} = 0.1$, $\sigma_j = 0$ for $j = n-\nu+1, \dots, n$, and therefore $\operatorname{cond} A = 10$.

1s. *Symmetric matrices of type I with nullity ν.* The same as in part 1n, but for $G = H$.

2n. *Nonsymmetric matrices of type II with nullity ν.* $A = (W, WZ)$ where W and Z are random orthogonal matrices of sizes $n \times (n-\nu)$ and $(n-\nu) \times \nu$, respectively.

2s. *Symmetric matrices of type II with nullity ν.* $A = WW^H$ where W are random orthogonal matrices of size $n \times (n-\nu)$.

3n. *Nonsymmetric Toeplitz-like matrices with nullity ν.* $A = c(T, TS)$ for random Toeplitz matrices T of size $n \times (n-\nu)$ and S of size $(n-\nu) \times \nu$ and for a positive scalar c such that $||A|| \approx 1$.

3s. *Symmetric Toeplitz-like matrices with nullity ν.* $A = cTT^H$ for random Toeplitz matrices T of size $n \times (n-\nu)$ and a positive scalar c such that $||A|| \approx 1$.

4n. *Nonsymmetric Toeplitz-like matrices with nullity one.* $A = (a_{i,j})_{i,j=0}^{n-1}$ is an $n \times n$ Toeplitz matrix. Its entries $a_{i,j} = a_{i-j}$ are random for $i - j < n - 1$. The entry $a_{n-1,0}$ is selected to ensure that the last row is linearly expressed through the other rows.

4s. *Symmetric Toeplitz-like matrices with nullity one.* $A = (a_{i,j})_{i,j=0}^{n-1}$ is an $n \times n$ Toeplitz matrix. Its entries $a_{i,j} = a_{i-j}$ are random for $|i - j| < n - 1$, whereas the entry $a_{0,n-1} = a_{n-1,0}$ is a root of the quadratic equation $\det A = 0$. We have repeatedly generated the matrices A until we arrived at the quadratic equation having real roots.

7.2 Generation of a nonsingular strictly $(0, \nu)$ input matrices A

We modified the above matrices with nullity ν to turn them into nonsingular matrices with numerical nullity ν in two ways. (To our previous abbreviations "s" and"n", we add another "n" for "nonsingular".)

1nn and 1ns. *Matrices of type I having numerical nullity* ν. The same matrices as in parts 1n and 1s in the previous subsection except that now $\sigma_j = 10^{-16}$ for $j > n - \nu$, so that $\operatorname{cond} A = 10^{16}$.

2nn, 3nn, 4nn, 2ns, 3ns, and 4ns. *Matrices of type II and Toeplitz-like matrices having numerical nullity* ν. $A = W/||W|| + \beta I_n$ where we defined the matrices W in the same way as the matrices A in the previous subsection. We set the scalar β equal to 10^{-16} in the symmetric case, so that $\sigma_1(A) = 1 + 10^{-16}$, $\sigma_j(A) = 10^{-16}$ for $j = n - \nu + 1, \dots, n$, whereas in the nonsymmetric case we iteratively computed a nonnegative scalar β such that $\sigma_1(A) \approx 1$ and

$$10^{-18} \le \sigma_{n-\nu+1}(A) \le 10^{-16}. \tag{7.1}$$

We initialized this iterative process with $\beta = 10^{-16}$, which implied that $\sigma_j(A) \le 10^{-16}$ for $j = n - \nu + 1, \dots, n$. If also $\sigma_{n-\nu+1}(A) > 10^{-18}$, so that bounds (7.1) held, we output this value of β and stopped. Otherwise we recursively set $\beta \leftarrow 10^{-16}\beta/\sigma_{n-\nu+1}(A)$. We output the current value of β and stopped as soon as bounds (7.1) were satisfied for the resulting matrix A. If they were not satisfied in 100 recursive steps, we restarted the process for a new input W.

7.3 Generation of APPs and the data on conditioning

In Tables 7.1 and 7.2 we display the data on generating APPs UV^H and on the conditioning of the A-modifications $C = A + UV^H$ and $C_1 = A + U_1V_1^H$ where we use APPs from Example 4.6 and their corrections $U_1V_1^H$ defined below and where we write $T_i = cI_r$ for all i with scalar c chosen to normalize the matrix U.

In the first column of each table we display the type of the input matrix A.

The second and the third columns show the values of ν, denoting the nullity (or numerical nullity) of the basic matrix A, and $\operatorname{cond} A$, denoting its condition number.

The fourth columns show the rank r of the APP UV^H from Example 4.6.

The fifth columns show the condition numbers $\operatorname{cond} C$ of the A-modifications $C = A + UV^H$.

The sixth columns have blank entries where $\operatorname{cond} C \le 10^5$. Wherever we had $\operatorname{cond} C > 10^5$, we computed a new APP $U_1V_1^H$ and the matrix $C_1 = A + U_1V_1^H$ and then displayed the condition number $\operatorname{cond} C_1$ in the sixth column and the rank of the new APP $U_1V_1^H$ in the fourth column.

To generate the APP $U_1V_1^H$, we either reapplied the same rules as before but with the APP's rank r incremented by one (see the results in Table 7.1) or defined this APP by the formulae $U_1 \leftarrow Q(C^{-1}U)$, $V_1^H \leftarrow Q(V^HC^{-1})$ in equation (6.1), without changing the rank r (see the results in Table 7.2).

We applied the same tests and obtained quite similar results for APPs of seven other types, namely,

a) and b) for APPs from Example 4.6 but with the sparse Toeplitz APCs, such that $T_i = c_i I_r$ where we first randomly sampled the coefficients c_i from one of the sets $\{-1, 1\}$ for type a) or $\{-2, -1, 1, 2\}$ for type b) and then normalized the matrix U by scaling,

c) for APPs from the same example but with T_i being real circulant matrices with random first columns,

d) for APPs from Example 4.1,

e) and f) for real APPs from Example 4.3 with random parameters from the line intervals $[-1, 1]$ for type e) or $[-10, 10]$ for type f), and

g) random real APPs.

For every selected APP UV^H we computed the matrices $C^{(p)} = A + 10^p UV^H$ for $p = -10, -5, 0, 5, 10$. In all tests, the values $\operatorname{cond} C^{(p)}$ were minimized for $p = 0$ and grew steadily (within the factor of $|p|$) as the integer $|p|$ grew. In Tables 7.1 and 7.2 we reported only the results for $p = 0$.

Table 7.1: APPs and conditioning I

Type	ν	Cond(A)	r	Cond(C)	Cond(C_1)
1n	1	8.40E+16	1	3.21E+2	
1n	2	4.56E+16	2	4.52E+3	
1n	4	3.90E+18	5	2.09E+5	1.81E+3
1n	8	5.69E+16	8	6.40E+2	
1s	1	1.98E+16	1	5.86E+2	
1s	2	3.69E+16	2	1.06E+4	
1s	4	2.91E+16	4	1.72E+3	
1s	8	3.36E+16	8	5.60E+3	
2n	1	3.48E+16	1	8.05E+1	
2n	2	1.53E+17	2	6.82E+3	
2n	4	2.73E+16	4	2.78E+4	
2n	8	1.23E+17	8	3.59E+3	
2s	1	4.13E+16	1	1.19E+3	
2s	2	4.67E+16	2	1.96E+3	
2s	4	4.40E+16	4	1.09E+4	
2s	8	1.33E+18	8	9.71E+3	
3n	1	3.96E+16	1	2.02E+4	
3n	2	2.18E+17	2	1.53E+3	
3n	4	1.37E+18	4	6.06E+2	
3n	8	4.24E+17	8	5.67E+2	
3s	1	1.69E+17	1	2.39E+4	
3s	2	4.58E+16	2	2.38E+3	
3s	4	1.39E+17	4	1.69E+3	
3s	8	1.60E+17	8	6.74E+3	
4n	1	1.22E+17	1	4.93E+2	
4n	2	3.26E+16	2	4.48E+2	
4n	4	5.99E+16	4	2.65E+2	
4n	8	1.23E+17	8	1.64E+2	
4s	1	3.22E+15	1	1.45E+3	
4s	2	2.34E+16	2	5.11E+2	
4s	4	1.09E+17	4	7.21E+2	
4s	8	2.29E+16	8	2.99E+2	

Table 7.2: APPs and conditioning II

Type	ν	Cond(A)	r	Cond(C)	Cond(C_1)
1n	1	2.63E+16	1	2.81E+2	
1n	2	2.98E+16	2	1.66E+3	
1n	4	3.85E+16	4	4.26E+3	
1n	8	3.55E+17	8	8.60E+2	
1s	1	5.10E+16	1	5.29E+2	
1s	2	2.22E+16	2	3.24E+4	
1s	4	2.96E+16	4	3.96E+4	
1s	8	2.88E+16	8	1.69E+3	
2n	1	1.06E+17	1	1.86E+2	
2n	2	3.58E+16	2	4.05E+2	
2n	4	9.90E+16	4	5.84E+3	
2n	8	8.29E+16	8	1.10E+4	
2s	1	1.25E+16	1	8.34E+2	
2s	2	2.71E+16	2	9.63E+2	
2s	4	5.91E+16	4	8.90E+3	
2s	8	5.49E+16	8	1.81E+4	
3n	1	1.85E+17	1	3.63E+3	
3n	2	9.71E+16	2	2.13E+4	
3n	4	1.76E+17	4	2.49E+3	
3n	8	3.70E+17	8	7.61E+2	
3s	1	1.30E+17	1	6.03E+3	
3s	2	1.03E+17	2	2.15E+4	
3s	4	7.20E+16	4	1.46E+4	
3s	8	8.98E+16	8	1.73E+6	9.93E+2
4n	1	1.74E+18	1	1.08E+3	
4n	2	9.08E+16	2	2.04E+2	
4n	4	2.57E+16	4	5.81E+1	
4n	8	7.66E+15	8	3.33E+1	
4s	1	2.60E+16	1	4.21E+2	
4s	2	2.55E+16	2	1.88E+2	
4s	4	7.80E+16	4	8.95E+2	
4s	8	1.81E+16	8	3.83E+2	

Appendix

Two impacts of preconditioning

1. ***Preconditioning as a means of convergence acceleration***

Suppose the Conjugate Gradient algorithm is applied to a linear system $A\mathbf{y} = \mathbf{b}$ where A is a Hermitian matrix whose spectrum is not limited to a small number of clusters. Then k iteration steps add the order of $k\sqrt{\operatorname{cond} A}$ new correct bits per a variable (cf. [4, Theorem 10.2.6]), and so A-preconditioning enables convergence acceleration.

How much does this acceleration increase the computational cost per iteration? The basic operation of the algorithm is the multiplication of an input matrix by a vector, whose computational cost is little affected by small-rank modifications of the input as well as by its large-rank structured modifications.

Similar comments can be applied to the GMRES and various other algorithms of this kind [4, Sections 10.2–10.4], [19], [31], [32].

Likewise, preconditioning can accelerate the Wilkinson's iterative refinement algorithm in [4, Section 3.5.3], [6, Sections 3.3.4 and 3.4.5], and [17, Chapter11]. Indeed, its k iterations add the order of $k \log(\frac{1}{||E|| \operatorname{cond} A})$ correct bits per a variable to an initial approximate solution to a linear system $A\mathbf{y} = \mathbf{b}$ provided an approximate solution $\tilde{\mathbf{z}} = (A + E)^{-1}\mathbf{v}$ to a linear system $A\mathbf{z} = \mathbf{v}$ is readily available where $||A^{-1}E|| < 1/2$.

For another example, preconditioning can accelerate Newton's iteration for the approximation of the inverse or the Moore–Penrose generalized inverse of a matrix A because $\log_2 \operatorname{cond} A + \log_2 \log_2(1/\delta) + O(1)$ Newton's iteration steps are sufficient to yield the residual norm bound δ (cf. [24, Chapter 6]).

One can enhance the power of the above tecnhiques by combining them together. E.g., given a structured linear systems of equations, we can begin with A-preconditioning of its coefficient matrix, then approximate the inverse of this matrix by applying the algorithms of the Conjugate Gradient/GMRES type, and finally refine the computed approximation by applying a structured version of Newton's iteration (cf. [24, Chapter 6], [33], and the bibliography therein).

The progress in solving linear systems of equations can be extended to various related computations, for example, to the solution of a polynomial system of equations, which can be reduced to the solution of sparse multi-level Toeplitz linear systems [34]. One can multiply the coefficient matrix of such a system by a vector fast (and can hardly exploit this matrix structure otherwise), but the algorithms of the Conjugate Gradient/GMRES types are not much effective in this case because the matrices are typically ill conditioned and have singular values widely spread out. Structured APCs of larger ranks promise critical support.

We have similar situation with the matrix methods for polynomial root-finding that employ the inverse iteration or shift-and-invert enhancement to

approximate the eigenvalues of the associated companion or generalized companion matrices (cf. [10]).

2. ***Preconditioning for improving the accuracy of the output.***

With preconditioning we can obtain a more accurate output by computing with the same precision. Such a power of preconditioning is well known for discretized solution of PDEs, eigen-solving, etc., but in some areas has not been recognized yet.

E.g., the reduction of non-Hermitian and overdetermined linear systems of equations to normal linear systems is "the method of choice when the matrix is well conditioned" [18, page 118]. The users, however, are cautious about this reduction because it squares the condition number of the input matrix, which means the loss of the accuracy of the output. Here preconditioning can be a natural remedy.

Likewise, it can support numerical computation of the sign of the determinant of an ill conditioned matrix, which is critical for computing convex hulls and Voronoi diagrams and is required in many other fundamental geometric and algebraic computations (see [35] and the bibliography therein).

References

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

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

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

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

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

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

[7] M. T. Chu, R. E. Funderlic, G. H. Golub, A Rank-One Reduction Formula and Its Applications to Matrix Factorizations, *SIAM Review*, **37**, **4**, 512–530, 1995.

[8] L. Hubert, J. Meulman, W. Heiser, Two Purposes for Matrix Factorization: A Historical Appraisal, *SIAM Review*, **42**, **1**, 68–82, 2000.

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

[10] V. Y. Pan, X. Yan, Additive Preconditioning, Eigenspaces, and Inverse Iteration, Technical Report TR 2007, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, April 2007.

[11] D. A. Bini, L. Gemignani, V. Y. Pan, Inverse Power and Durand/Kerner Iteration for Univariate Polynomial Root-finding, *Computers and Mathematics (with Applications)*, **47**, **2/3**, 447–459, January 2004. (Also Technical Reports TR 2002 003 and 2002 020, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2002.)

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

[13] D. A. Bini, L. Gemignani, V. Y. Pan, Improved Initialization of the Accelerated and Robust QR-like Polynomial Root-finding, *Electronic Transactions on Numerical Analysis*, **17**, 195–205, 2004.

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

[15] V. Y. Pan, B. Murphy, G. Qian, R. E. Rosholt, Error-free Computations via Floating-Point Operations, Technical Report TR 2007, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, April 2007.

[16] X. Wang, Affect of Small Rank Modification on the Condition Number of a Matrix, *Computer and Math. (with Applications)*, in print.

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

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

[19] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van Der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1993.

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

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

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

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

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

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

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

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

[28] D. Bini, V. Y. Pan, Parallel Complexity of Tridiagonal Symmetric Eigenvalue Problem, *Proc. 2nd Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'91)*, 384–393, ACM Press, New York, and SIAM Publications, Philadelphia, 1991.

[29] D. Bini, V. Y. Pan, Computing Matrix Eigenvalues and Polynomial Zeros Where the Output Is Real, *SIAM J. on Computing*, **27, 4**, 1099–1115, 1998.

[30] V. Y. Pan, Computations in the Null Spaces with Additive Preconditioning, Technical Report TR 2007, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, April 2007.

[31] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Co., Boston, 1996 (first edition) and SIAM Publications, Philadelphia, 2003 (second edition).

[32] H. A. Van der Vorst, *Iterative Krylov Methods for Large Linear Systems,* Cambridge University Press, Cambridge, England, 2003.

[33] V. Y. Pan, M. Kunin, R. Rosholt, H. Kodal, Homotopic Residual Correction Processes, *Math. of Computation*, **75**, 345–368, 2006.

[34] D. Bondyfalat, B. Mourrain, V. Y. Pan, Computation of a Specified Root of a Polynomial System of Equations Using Eigenvectors, *Linear Algebra and Its Applications*, **319**, 193–209, 2000.

[35] H. Brönnimann, I. Z. Emiris, V. Y. Pan, S. Pion, Sign Determination in Residue Number Systems, *Theoretical Computer Science*, **210**, **1**, 173–197, 1999.