City University of New York (CUNY)
CUNY Academic Works

Computer Science Technical Reports

Graduate Center

2008

TR-2008004: Additive Preconditioning for Matrix Computations

Victor Y. Pan

Dmitriy Ivolgin

Brian Murphy

Rhys Eric Rosholt

Yuqing Tang

See next page for additional authors

Follow this and additional works at: http://academicworks.cuny.edu/gc_cs_tr

Part of the Computer Sciences Commons

Recommended Citation

Pan, Victor Y.; Ivolgin, Dmitriy; Murphy, Brian; Rosholt, Rhys Eric; Tang, Yuqing; and Yan, Xiaodong, "TR-2008004: Additive Preconditioning for Matrix Computations" (2008). *CUNY Academic Works.*
http://academicworks.cuny.edu/gc_cs_tr/309

This Technical Report is brought to you by CUNY Academic Works. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of CUNY Academic Works. For more information, please contact AcademicWorks@gc.cuny.edu.

Authors
Victor Y. Pan, Dmitriy Ivolgin, Brian Murphy, Rhys Eric Rosholt, Yuqing Tang, and Xiaodong Yan

This technical report is available at CUNY Academic Works: http://academicworks.cuny.edu/gc_cs_tr/309

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
{victor.pan}{brian.murphy}{rhys.rosholt}@lehman.cuny.edu

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

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

Abstract

Versus the customary preconditioners, our weakly random ones are generated more readily and for a much larger class of input matrices. Furthermore our preconditioners have a wider range of applications, in particular to linear systems with rectangular and rank deficient coefficient matrices and to eigen-solving. We study the generation of such preconditioners and their impact on conditioning of the input matrix. Our analysis and experiments show the power of this approach even where we use weak randomization, with fewer random parameters, and choose sparse and structured preconditioners.

2000 Math. Subject Classification: 65F22, 65F35, 65A12

Key words: Matrix computations, Additive preconditioning, Weak randomization

1 Introduction

1.1 Background: multiplicative preconditioning

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

*Supported by PSC CUNY Awards 66437-0035, 67297-0036 and 68291–0037 Some results of this paper have been presented at the International Conferences on the Matrix Methods and Operator Equations in Moscow, Russia, in June of 2005 and July 2007, on the Foundations of Computational Mathematics (FoCM'2005) in Santander, Spain, in July 2005, and on Industrial and Applied Mathematics, in Zürich, Switzerland, in July 2007, as well as at the SIAM Annual Meeting, in Boston, in July 2006, at the Third International Workshop on Symbolic-Numeric Computation (SNC'07) in London, Ontario, Canada, in July 2007, and at the Third International Computer Science Symposium in Moscow, Russia, in June of 2008.

$MAN\mathbf{x} = M\mathbf{b}$ for $\mathbf{y} = N\mathbf{x}$ and readily computable nonsingular matrices M and N, called preconditioners (see [1]–[4] and the bibliography therein). Such systems can be solved faster and more accurately. Generally, however, computing the desired multiplicative preconditioners M and N can be as costly as Gaussian elimination, and so preconditioning florishes only for large but special classes of input matrices A.

1.2 Weakly random additive preprocessing

As an alternative or complementary tool, we propose *weakly random additive preprocessing* $A \leftarrow C = A + P$, i.e., we add a matrix P (having a smaller rank or structured) to the input matrix A to obtain its *additive modification* C with a smaller condition number. Hereafter we use the abbreviations "A-" for "additive", "*APPs*" for A-preprocessors, "AC" for "A-complements", and "APCs" for "A-preconditioners". ACs (resp. APCs) are the APPs P such that the input matrix A is rank deficient (resp. ill conditioned), whereas the matrix $C = A + P$ is not.

For ill conditioned matrices A with at most r small singular values and random APPs P of ranks of at least r such that the ratios $||A||/||P||$ are neither large nor small, we quite regularly arrive at well conditioned matrices $C = A + P$. As usual we assume that the concepts "large", "small", "well" and "ill" are quantified in the context of the computational tasks and computer environment, and we assume the customary IEEE model of numerical computing with rounding. We write M^H to denote the Hermitian (complex conjugate) transpose of a matrix M (which is the transpose M^T if M is a real matrix), and we let $A = S\Sigma T^H$ denote the SVD of a matrix A where S and T are the unitary (orthogonal) matrices of the left and right singular vectors of the matrix A, respectively.

We can explain the above phenomenon based on two properties below.

1. Random matrices tend to be well conditioned. This claim has been proved for various large classes of matrices and is generally supported by huge empirical evidence.

2. Assume an APP $P = UV^H$, a pair of properly scaled $n \times r$ matrices U and V, and an SVD $A = S\Sigma T^H$ of an $n \times n$ matrix A having at most r singular values that are small relative to the norm $||A||$. Then we prove that the matrix $C = A + P$ is well conditioned if so are the $r \times r$ tailing (southern) submatrices of the matrices $S^H U$ and $T^H V$.

It follows that the A-modification C tends to be well conditioned for scaled random matrices U and V of rank r, and empirically even *weak randomization* is sufficient where the APPs have some fixed patterns of structure and sparseness and depend on fewer random parameters. This is in contrast with random multiplicative preconditioning because random matrices tend to be well conditioned and because $\operatorname{cond} A \le \prod_i \operatorname{cond} F_i$ if $A = \prod_i F_i$.

To summarize, our APCs and ACs are generated more readily and for a much larger class of matrices than multiplicative preconditioners. Furthermore they better preserve matrix structure and sparseness and have a wider range of applications. In particular they remain effective for rectangular and rank deficient matrices A.

The papers [5]–[11] cover effective applications of such APCs and ACs to the solution of singular and nonsingular linear systems of equations, eigen-solving, and the computation of determinants. In this paper (which is the journal version of the proceedings paper [9]) we generate ACs and APCs and study their impact on conditioning.

1.3 Organization of our paper

We organize our paper as follows. We begin with the definitions in the next section. We generate random AC and APCs in Section 3 and sparse and structured APCs in Section 6. In Section 5 we generate, refine and compress the APCs. In Section 7 we cover dual APPs. We study conditioning of A-modifications of an input matrix A theoretically in Section 4 and experimentally in Section 8. In Section 9 we briefly comment on preconditioning by expansion and its link to A-preconditioning.

Our numerical tests have been designed by the first author and performed by his coauthors. Otherwise this work with all typos and other errors is due to the first author.

We present our study for square matrices but a large part of it can be extended to rectangular matrices.

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 [12]. Helpful and encouraging were the interest and comments to our work from the participants of the International Conferences on the Matrix Methods and Operator Equations in Moscow, Russia in 2005 and 2007 and on FoCM 2005 in Santander, Spain in 2005 (particularly from J. W. Demmel, G. H. Golub, V. Olshevsky, and M. Van Barel).

2 Basic definitions

Most of our basic definitions reproduce or slightly modify the customary definitions in [13]–[19] for matrix computations, in particular, for Hermitian, unitary (orthogonal), singular, full-rank and rank deficient matrices, the $k \times k$ identity matrices $I = I_k$, $k \times l$ matrices $0 = 0_{k,l}$ filled with zeros, the transpose A^T and the Hermitian transpose A^H of an $n \times n$ matrix A, its rank $\rho = \operatorname{rank} A$, nullity $\operatorname{nul} A = n - \rho$, singular values $\sigma_j(A)$, in the nonincreasing order for $j = 1, \ldots, \rho$; 2-norm $||A|| = \sigma_1(A)$, the condition number $\operatorname{cond} A = \frac{\sigma_1(A)}{\sigma_\rho(A)}$, and the *Moore-Penrose generalized inverse* A^+ (also called *pseudo inverse* and equal to the inverse A^{-1} for a nonsingular matrix A). A matrix A is normalized if $||A|| = 1$. We write $M \geq 0$ for a nonnegative definite Hermitian matrix M. We write $\alpha \gg \gamma$ where the ratio α/γ is large. We say that $r = \operatorname{nnul} A$ is the numerical nullity and $n - r = \operatorname{nrank} A$ is the numerical rank of the matrix A if the ratio $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$ is not large, whereas $\sigma_1(A) \gg \sigma_{n-r+1}(A)$, that is if the matrix has exactly r singular values that are small relatively to its norm $||A|| = \sigma_1(A)$, e.g. are less than $\tau||A||$ for a fixed small positive tolerance value τ. $(B_1, \ldots, B_k)$ and $\operatorname{diag}(B_i)_{i=1}^k$ denote the $1 \times k$ block matrix with the blocks $B_1, \ldots, B_k$ and $k \times k$ block diagonal matrix with the diagonal blocks $B_1, \ldots, B_k$, respectively. We write $Q(M)$ for the Q-factor of the size $m \times n$ in the thin QR factorization of an $m \times n$ matrix M of the full rank where the R-factor has positive diagonal entries. $\mathbb{C}$ is the field of complex numbers.

3 Generation of ACs and APCs

3.1 Error-free A-preprocessing

We represent an $n \times n$ APPs P of a rank r by a pair of $n \times r$ *generators* U and V such that $P = UV^H$. If the input size $n \times n$ is reasonable and the entries of the $n \times n$ matrix A are not full precision numbers, we can control or avoid rounding errors in numerical computation of APPs UV^H and A-modifications $C = A + UV^H$ with rounding. To achieve this, we can fill the generator matrices U and V with short (lower precision) numbers (possibly just with the integers -2, -1, 0, 1, and 2 or even -1, 0, and 1) or apply the expansion approach in Section 9.

3.2 The basic theorem for ACs

Suppose $A, C \in \mathbb{C}^{n\times n}$, $U, V \in \mathbb{C}^{n\times r}$, and U and V have full rank r. Then

$$\{\operatorname{rank} C = n\} \Longrightarrow \{r \geq \operatorname{nul} A\},$$

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

Let us formalize these simple relationships.

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

Lemma 3.1. *[20] (cf. also [21], [22]). For a finite set Δ of cardinality $|\Delta|$ in a ring $\mathbf{R}$, let a polynomial in m variables not vanish identically on the set Δ^m, let it have total degree d, and let the values of its variables be randomly sampled from the set Δ. Then the polynomial vanishes with a probability of at most $\frac{d}{|\Delta|}$.*

Theorem 3.1. *For a finite set Δ of cardinality $|\Delta|$ in a ring $\mathbf{R}$ and four matrices $A \in \mathbf{R}^{n\times n}$ of a rank ρ, U and V in $\Delta^{r\times n}$, and $C = A + UV^T$, we have*

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

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

c) $\operatorname{rank} C = n$ *with a probability of at least* $1 - \frac{r}{|\Delta|}$ *if* $r + \rho \geq n$, *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 $r + \rho \geq n$. Then clearly, $\operatorname{rank} C = n$ if $U = V$ and if the entries of the matrix U are indeterminates. Since $\det C$ is a polynomial of a total degree of at most $2(n - \rho) \leq 2r$ in these entries, part b) follows from Lemma 3.1. Part c) is proved similarly to part b). □

3.3 Generation of randomized ACs and APCs

In virtue of Theorem 3.1 a random APP UV^H of a rank r is likely to be an AC if $r \geq \operatorname{nul} A$, whereas an APP of a rank r is never an AC otherwise. Randomized linear or binary search for the value $\operatorname{nul} A$ can rely on these properties.

Likewise, assuming $M \in \mathbb{C}$ and $\operatorname{nnul} M = r$, we can generate APCs based on the following (sketched) extension of Theorem 3.1:

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

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

Seeking $\operatorname{nnul} A$, however, we should choose properly scaled well conditioned APPs for which the ratio $\frac{||UV^H||}{||A||}$ is neither large nor small, and in such a search we should test the candidate A-modifications for being well conditioned rather than having full rank. The algorithm only requires a random number generator and crude estimates for the ratio $\frac{||UV^H||}{||A||}$. Formally we also ought to estimate the condition numbers of the matrices U, V and C (see [13, Sections 2.3.2, 2.3.3, 3.5.4, and 12.5], [14, Sections 5.3 and 5.4], and [16, Chapter 15] on the norm and condition estimators), but practically this stage can be relaxed because random matrices U and V tend to be well conditioned.

4 APPs and conditioning

4.1 Sharp lower estimates

In this section we estimate the ratio $\frac{\operatorname{cond} A}{\operatorname{cond} C}$ from above but first recall the following sharp lower bounds from [12].

Theorem 4.1. *For any $n \times n$ matrix $A \geq 0$, we have*

$$\min_{P \geq 0, \operatorname{rank} P \leq k} \operatorname{cond}(A + P) = \frac{\sigma_1(A)}{\sigma_{n-k}(A)}.$$

The minimum is reached where

$$A = \operatorname{diag}(\sigma_j)_{j=1}^n \ \ \textit{and} \ \ P = \operatorname{diag}(0, \ldots, 0, \sigma_{n-k} - \sigma_{n-k+1}, \ldots, \sigma_{n-k} - \sigma_n).$$

Theorem 4.2. *For any $n \times n$ nonsingular matrix A,*

$$\min_{\operatorname{rank} P \leq k} \operatorname{cond}(A + P) = \begin{cases} \frac{\sigma_{k+1}(A)}{\sigma_{n-k}(A)}, & k < \frac{n}{2} \\ 1, & k \geq \frac{n}{2} \end{cases}$$

If we know the SVD of the input matrix $A = S^H \Sigma T$, we can compute an APC supporting this theorem as follows: first bring the matrix A into the diagonal form Σ and then recursively apply the following result [12].

Theorem 4.3. *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 (respectively real) if so is the input matrix.

In contrast to the above restriction on the dynamics of singular values, any prescribed change of the eigenvalues can be obtained even with a rank-one modification, e.g., for a Frobenius (companion) matrix.

4.2 Randomized upper estimates (the objective and the two main steps)

Even without costly computation of the SVD, we can achieve quite strong preconditioning power. Namely our analysis and extensive tests show that the value $\operatorname{cond} C$ is likely to be roughly of the order of $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$ provided A is an $n \times n$ matrix and an APP UV^H of a rank r is weakly random and scaled so that the ratio $\frac{||UV^H||}{||A||}$ is neither large nor small. We first show this property for a singular well conditioned matrix A with a nullity r. Then in Sections 4.5 and 4.6 we extend our study to nonsingular ill conditioned matrices A with numerical nullity $r = \operatorname{nnul} A$.

4.3 ACs and conditioning: the basic estimates

We first factorize the A-modification C.

Theorem 4.4. *Let $A = S\Sigma T^H$ be the SVD of an $n \times n$ matrix A, where S and T are $n \times n$ unitary matrices, so that $S^H S = SS^H = T^H T = TT^H = I_n$, $\Sigma = \operatorname{diag}(\Sigma_A, 0_r)$ is an $n \times n$ diagonal matrix of a rank $\rho = n - r$, and $\Sigma_A = \operatorname{diag}(\sigma_j)_{j=1}^{\rho}$ is the diagonal matrix of the positive singular values of the matrix A (cf. our comment 3 in Section 4.6). Let U and V be $n \times r$ matrices for $r < n$ such that the $n \times n$ matrix $C = A + UV^H$ is nonsingular. Write*

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

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

Proof. Observe that $S^H CT = \Sigma + S^H UV^H T$, $R_U \Sigma R_V^H = \Sigma$, $S^H U = R_U \begin{pmatrix} 0 \\ I_r \end{pmatrix}$, $T^H V = R_V \begin{pmatrix} 0 \\ I_r \end{pmatrix}$, and $\begin{pmatrix} 0 \\ I_r \end{pmatrix} (0, I_r) = \operatorname{diag}(0, I_r)$. Deduce that

$$\tilde{S}^H CT = R_U \Sigma R_V^H + R_U \operatorname{diag}(0, I_r) R_V^H = R_U \operatorname{diag}(\Sigma_A, I_r) R_V^H$$

and arrive at part a). Part b) follows because the matrix C is nonsingular. □

Corollary 4.1. *Under the assumptions of Theorem 4.4 we have*

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

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

so that

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

Proof. The corollary follows from Theorem 4.4 because $\operatorname{cond} M = ||M||\ ||M^+||$ and $||M^H|| = ||M||$ for any matrix M and because S and T are unitary matrices. □

4.4 ACs and conditioning: refined estimates

Lemma 4.1. *For any pair of matrices X and Y of compatible sizes we have*

$$\max\{||X||, ||Y||\} \leq ||(X, Y)|| = ||(X, Y)^H|| \leq \sqrt{||X||^2 + ||Y||^2}.$$

Proof. Let $||(X, Y) \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} || = ||(X, Y)||$ for two vectors $\mathbf{u}$ and $\mathbf{v}$ such that

$$|| \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} ||^2 = ||\mathbf{u}||^2 + ||\mathbf{v}||^2 = 1.$$

Recall that $(X,Y)\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix} = X\mathbf{u}+Y\mathbf{v}$ and deduce that

$$||(X,Y)|| = ||X\mathbf{u}+Y\mathbf{v}|| = ||(X,Y)\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix}||.$$

Apply Cauchy–Schwartz bound and obtain that

$$||(X,Y)||^2 \le (||X||^2+||Y||^2)(||\mathbf{u}||^2+||\mathbf{v}||^2) = ||X||^2+||Y||^2,$$

which is the claimed upper bound. Now let $||X\mathbf{w}|| = ||X||$ where $||\mathbf{w}|| = 1$. Then $||X|| = ||(X,Y)\begin{pmatrix}\mathbf{w}\\ \mathbf{0}\end{pmatrix}|| \le ||(X,Y)||$. Similarly we obtain that $||Y|| \le ||(X,Y)||$. Finally recall that $||L|| = ||L^H||$ for any matrix L. □

Theorem 4.5. *Suppose the matrices U and V have full rank. Then we have*

$$\begin{array}{rcccl} \max\{1,||U||^2\} & \le & ||R_U||^2 & \le & 1+||U||^2, \\ \max\{1,||V||^2\} & \le & ||R_V||^2 & \le & 1+||V||^2, \\ 1 & \le & ||R_U^{-1}||^2 & \le & 1+(1+||U||^2)||U_r^{-1}||^2, \\ 1 & \le & ||R_V^{-H}||^2 = ||R_V^{-1}||^2 & \le & 1+(1+||V||^2)||V_r^{-1}||^2. \end{array}$$

Proof. The bounds on the norms $||R_U||$ and $||R_V||$ follow from Lemma 4.1 because $R_U = (I_{n,\rho}, U)$ and $R_V = (I_{n,\rho}, V)$ where $I_{n,\rho} = \begin{pmatrix}I_\rho\\ 0\end{pmatrix}$. The lower bounds on the norms $||R_U^{-1}||$ and $||R_V^{-1}||$ are obvious. To bound the norm $||R_U^{-1}||$ from above, first observe that $R_U^{-1} = \operatorname{diag}(I_\rho, 0) + \begin{pmatrix}-U_\rho\\ I_r\end{pmatrix} U_r^{-1}$. Now apply Lemma 4.1 at first to this matrix and then to the matrix $\begin{pmatrix}-U_\rho\\ I_r\end{pmatrix}$ and obtain that

$$||R_U^{-1}||^2 \ \le\ 1+||\begin{pmatrix}-U_\rho\\ I_r\end{pmatrix} U_r^{-1}||^2 \ \le\ 1+||\begin{pmatrix}-U_\rho\\ I_r\end{pmatrix}||^2||U_r^{-1}||^2$$

and

$$||\begin{pmatrix}-U_\rho\\ I_r\end{pmatrix}||^2 \ \le\ 1+||U||^2.$$

By combining the latter bounds obtain the desired estimate for the norm $||R_U^{-1}||$. The norm $||R_V^{-1}||$ is estimated similarly. □

The next two theorems are immediately verified,

Theorem 4.6. *Under the assumptions of Theorem 4.4, suppose that*

$$\sigma_{n-r} \le 1 \le \sigma_1. \tag{4.1}$$

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

Theorem 4.7. *Let us write $\theta = \frac{||UV^H||}{||A||}$. Then we have*

$$|1-\theta| \le \frac{||C||}{||A||} \le 1+\theta.$$

Corollary 4.2. *Write*

$$\theta = ||UV^H||/||A||, \quad q = ||R_U||\ ||R_V|| \quad \text{and} \quad p = ||R_U^{-1}||\ ||R_V^{-1}||,$$

so that

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

Then under the bounds (4.1) and the assumptions of Theorems 4.4 and 4.7 we have

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

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

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

Proof. Parts a) and b) follow from Corollary 4.1 and Theorems 4.5-4.7. Part c) follows from parts a) and b). □

The corollary shows that the transition $A \rightarrow C$ (which tends to yield the full rank property) changes the norms and condition numbers of the matrices only within the factor $p\min\{1+\theta, q\}$. Clearly we can nicely bound the parameters θ and q by properly scaling the matrices A, U and V. We estimate the bound p in Section 4.6.

Now suppose we represent an ill conditioned matrix of full rank as the sum $A+E$ where E is a small norm matrix and A is a well conditioned and rank deficient matrix. Then perturbation by the matrix E little affects our analysis, and so our next results extend it to the A-modification $C = A + E + UV^H$ for a random APP UV^H.

4.5 The impacts of A-modification on full rank matrices

We extend the bounds of Corollary 4.2 in the cases where $||E||$ is small or $C \geq 0$ and $E \geq 0$.

Theorem 4.8. *Let the matrices C and $\tilde{C} = C + E$ be nonsingular. Write $\delta = ||E||$ and $\delta_C = \delta||C^{-1}||$. Then we have*

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

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

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

Proof. Parts a) and c) follow immediately. Part b) follows because $||\tilde{C}^{-1}|| = \frac{1}{\sigma_n(\tilde{C})} \leq \frac{1}{\sigma_n(C)-\delta}$. □

4.6 Further comments

1. Bounds (4.1) in Corollary 4.2 are no loss of generality. Indeed scale the matrices A, UV^H, and $C = A + UV^H$ by the same scalar s and observe that the ratios $\frac{||C||}{||A||}$, $\frac{||C^{-1}||}{||A^+||}$, and $\frac{\operatorname{cond} C}{\operatorname{cond} A}$ do not change, whereas $\sigma_j \rightarrow s\sigma_j$.

2. An APP UV^H cannot be an APC if the ratio $\theta = \frac{||UV^H||}{||A||}$ is small because $\sigma_n(C) \le \sigma_n(A) + ||UV^H||$. Furthermore, an APP UV^H cannot be an APC if the ratio θ is large and if $\operatorname{rank}(UV^H) < \operatorname{rank} A$. Corollary 4.2 provides us, however, with reasonable bounds on the ratio $\frac{\operatorname{cond} C}{\operatorname{cond} A}$ as long as the norms $||U||$, $||V||$, $||U_r^{-1}||$, and $||V_r^{-1}||$ are reasonable. We can assume that $||U|| = ||V|| = 1$ and then obtain that $1 \le q \le 2$ and $1 \le p^2 \le (1+2||U_r^{-1}||)(1+2||V_r^{-1}||)$ in Corollary 4.2.

3. The value p in Corollary 4.2 is not large where the matrices U_r and V_r are well conditioned. Now recall about huge empirical evidence and some formal support in [23]–[26] and the references therein that random matrices tend to be well conditioned. Thus in the case of random generators U and V we can expect that the value p is not large.

   Can we expect to have the same property when we restrict randomness of the matrices U and V by imposing various patterns of structure and sparseness on these matrices and using fewer random parameters? Yes, according to our extensive tests (see Sections 6 and 8). We speculate that premultiplication by matrices S^H and T^H strengthens the randomness of the matrices U and V because typically the latter matrices U and V are generated independently of the former matrices S and T of the singular vectors of the matrix A.

4. In virtue of Theorem 3.1, the A-modifications $\tilde{C} = \tilde{A} + P$ of a singular matrix $\tilde{A}$ having nullity r are likely to be nonsingular for random APPs P of rank r. Corollary 4.2 shows that preprocessing $\tilde{A} \rightarrow \tilde{C}$ with scaled random APPs of rank r tends to keep the matrix A well conditioned, and then preprocessing $A \rightarrow C = A + P$ with the same APPs P would turn the nearby ill conditioned matrices A with $\operatorname{nnul} A = r$ into well conditioned nonsingular matrices C.

5. According to our extensive tests (see Section 8), the estimated impact of A-preconditioning on the singular values is quite regular so that random APPs can be used for detecting large jumps in the spectra of the singular values and for computing numerical rank and numerical nullity. This application can be reinforced with the techniques in the next section.

5 Generation, refinement, and compression of APCs

In the unlikely case where our randomization works poorly, we can just reapply A-preconditioning with a new scaled weakly random APP. This has a good chances for success, according to our study in the previous section and test results, but let us next fix our APPs without generation of new random APPs. Suppose for an ill conditioned matrix A, we have arrived at a substantially better conditioned but still too crude A-modification $C = A + UV^H$ of a rank r, so that $\operatorname{cond} A \gg \operatorname{cond} C \gg \frac{\sigma_1}{\sigma_{\rho-r}}$. Then the following transform serves as a remedy, according to our extensive tests (cf. Table 8.2) and the analysis in [7, Section 6] and [12]:

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

where we use the notation $Q(M)$ in Section 2.

Now suppose we have an upper bound r_+ on the unknown number r of small and zero singular values of an input matrix A. Then we can generate a random scaled APC UV^H of rank r_+ and compress it into APCs of recursively decreasing ranks r by extending transform (5.1) as follows.

Procedure 5.1. Generation of an APC via inflation and compression.

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 suitably scaled and well conditioned matrix $T(U)$ (resp. $T^H(V)$) of full rank whose columns (resp. rows) generate the right (resp. left) singular space of the matrix AC^+U (resp. V^HC^+A) associated with the r smallest singular values of these matrices. (Here we count the singular values with their multiplicity and include the vanishing singular values zero.)*

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

The ranges of the resulting matrices $U = Q(C^+UT(U))$ and $V = Q((T^H(V)V^HC^+)^H)$ approximate the bases for the left and right singular spaces, respectively, associated with the r smallest singular values of the matrix A (cf. [7], [11], [12]). According to our tests and the ones in [12], this yields highly effective preconditioners UV^H of rank r.

6 Structured and sparse APPs

All APPs of small ranks are structured, but next we supply various examples of sparse and structured APPs of any rank. In our extensive tests, these APPs were typically APCs for all classes of tested input matrices. We welcome more such examples of weakly random APCs from the readers.

Example 6.1. 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 r fixed sets $\mathbb{S}_1, \ldots, \mathbb{S}_r$ 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., [27, 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 $\mathbb{S}_1, \ldots, \mathbb{S}_r$ are real sets, then the APP is Hermitian. If the sets $\mathbb{S}_1, \ldots, \mathbb{S}_r$ lie in the annulus $\{x : \ d_- \le |x| \le d_+\}$, then $\mathrm{cond}(UV^H) = \mathrm{cond}\, D_r \le d_+/d_-$.

Example 6.2. f-circulant APPs *[27, 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 6.3. Toeplitz-like APPs I. *Define an $n \times r$ well conditioned Toeplitz matrix U of full rank. 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. Define a matrix V a) either similarly or b) set $V = U$ (to produce a Hermitian APP). The APP UV^H 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 (cf. [27]).*

Example 6.4. Structured or sparse APPs I. *Define a matrix $U = PW$, P for a fixed or random $n \times n$ permutation matrix P (in the simplest case $P = I_n$) and a fixed or random $n \times r$ block W of the $n \times n$ matrix of the discrete Fourier, sine or cosine transform [27, Section 3.11], or of another well conditioned matrix with a fixed structure such as the sparseness structure or the displacement structure of Toeplitz, Hankel, Vandermonde, or Cauchy types. Example 6.3 is the special case where $P = I_n$ and W is a Toeplitz matrix. Define a matrix V a) either similarly or b) set $V = U$ (to produce a Hermitian APP). Define an APP UV^H. The complexity of its multiplication by a vector can be linear or nearly linear, depending on its structure.*

Example 6.5. Toeplitz-like APPs II. *Define 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 $kr + n_1 + \dots + n_k = n$. 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. Define a matrix V a) either similarly or b) set $V = U$ (to produce a Hermitian APP). For general Toeplitz matrices $T_1, \dots, T_k$ and the shift operators associated with the Toeplitz structure, the APP UV^H 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 $c_1^2 + \dots + c_k^2 = 1$.

Example 6.6. Structured or sparse APPs II. *Define a well conditioned matrix*

$$U = P(T_1, 0_{r,n_1}, \dots, T_k, 0_{r,n_k})^T$$

for an $n \times n$ permutation matrix P and integers k, $n_1, \dots, n_k$ chosen as in Example 6.5 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, or sparse matrices with fixed patterns of sparseness. Define a matrix V a) either similarly or b) set $V = U$ (to produce a Hermitian APP). Define an APP UV^H. Example 6.5 is the special case where $P = I_n$ and T_i are Toeplitz matrices.

Finally, we can generate APCs by appending pairs of (block) rows and (block) columns that preserve the structure of an input matrix.

7 Dual A-preprocessing

Let us next generate dual APCs by implicitly applying A-preconditioning to the (generalized) inverse A^+ matrix without computing this matrix. This option is valuable because it enables division-free reduction of solving linear systems and computation of determinants to the case of well conditioned input. Namely, we represent the dual A-modification $C_- = A^+ + VU^H$ by its (generalized) inverse

$$(C_-)^+ = (A^+ + VU^H)^+ = A - AVH^+U^HA, \quad H = I_q + U^HAV. \tag{7.1}$$

We call this equation *the dual SMW formula* (cf. the Sherman–Morrison–Woodbury formula in [13, page 50]).

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 our analysis to dual A-preprocessing. In particular, the matrix $(C_{-})^{+}$ is likely to be well conditioned where the ratio $\frac{||VU^{H}||}{||A^{+}||}$ is neither large nor small for a weakly random (well conditioned) APP VU^{H} of a sufficiently large rank. The latter ratio involves the norm $||A^{+}||$, which is a little harder to compute than the norm $||A||$, involved into the generation of APPs UV^{H}.

Finally, here is a natural extension of our policy (5.1) to dual APPs VU^{H},

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

8 Numerical tests for generating APCs

In our tests we first generated singular and nearly singular matrices of 16 classes, modified them with random and weakly 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.

Then we applied similar tests to the diagonal matrices with singular values forming a geometric progression.

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 8.1 and 8.2.

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. [13], [14]).

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

8.1 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, that is, 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 $j = 1, \ldots, n-1$, $\sigma_1 = 1$, the values $\sigma_2, \ldots, \sigma_{n-\nu-1}$ are randomly sampled in the semi-open interval $[0.1, 1)$, $\sigma_{n-\nu} = 0.1$, $\sigma_j = 0$ for $j = n-\nu+1, \ldots, n$, 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.

8.2 Generation of ill conditioned 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.* $A = \frac{W}{||W||} + \beta I_n$ 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 $j = n-\nu+1, \ldots, n$, whereas in the nonsymmetric case we iteratively computed a nonnegative scalar β such that $\sigma_1(A) \approx 1$ and

$$10^{-18} \le \sigma_{n-\nu+1}(A) \le 10^{-16}. \tag{8.1}$$

We initialized this iterative process with $\beta = 10^{-16}$, which implied that $\sigma_j(A) \le 10^{-16}$ for $j = n-\nu+1, \ldots, n$. If also $\sigma_{n-\nu+1}(A) > 10^{-18}$, so that bounds (8.1) held, we output this value of β and stopped. Otherwise we recursively set $\beta \leftarrow \frac{10^{-16}\beta}{\sigma_{n-\nu+1}(A)}$. We output the current value of β and stopped as soon as bounds (8.1) 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.

8.3 Generation of APPs and the data on conditioning

In Tables 8.1 and 8.2 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 6.6b) and their corrections $U_1V_1^H$ defined below and where $U = V$, $U_1 = V_1$, and 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 6.6b).

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 $\operatorname{cond} C \leq 10^5$. 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 8.1) or defined this APP by the formulae $U_1 \leftarrow Q(C^{-1}U)$, $V_1^H \leftarrow Q(V^HC^{-1})$ in equation (5.1), without changing the rank r (see the results in Table 8.2).

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

a) and b) for APPs from Example 6.6b) but with the sparse Toeplitz APCs, such that $T_i = c_iI_r$ 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 6.1,

e) and f) for real APPs from Example 6.3b) 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 $C^{(p)} = A + 10^pUV^H$ 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 8.1 and 8.2 we reported only the results for $p = 0$.

8.4 The case of diagonal input matrices

We applied A-preconditioning with APPs UV^T to $n \times n$ diagonal matrices

$$A = (\operatorname{diag} 2^{64\frac{i}{n}})_{i=0}^{n-1} \quad \text{for} \quad n = 64, 128.$$

We first generated the following $n \times r$ matrices U_1 and V_1 for $r = \frac{nj}{8}$, $j = 1$, 2, 3, 4, 5, 6, 7.

1. Random matrices U_1 and V_1
2. Random matrix U_1, $V_1 = U_1$
3. Random unitary matrices U_1 and V_1
4. Random unitary matrix U_1, $V_1 = U_1$
5. Random Toeplitz matrices U_1 and V_1
6. Random Toeplitz matrix U_1, $V_1 = U_1$.

Then we scaled the matrices U_1 and V_1 to yield the matrices U_2 and V_2 such that $||U_2V_2^T|| \approx ||A||$.

Finally we truncated all entries of the matrices U_2 and V_2 to eight bits and denoted the resulting matrices U and V. The truncation has ensured that the APPs UV^T had the desired ranks r and

Table 8.1: 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	

that $C - A = UV^T$, even though we computed these APPs and the A-modifications $C = A + UV^T$ with floating point and with rounding to the standard IEEE double precision.

Our Tables 8.3–8.14 display the test results observed in 1000 tests for each pair of n and r. They show that the ratio $\frac{(n-r)\log\operatorname{cond} A}{n\log\operatorname{cond} C}$ was consistently in a rather narrow range between 0.5 and 1.

Table 8.2: 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	

9 Further work

[10, Section 12] describes *preconditioning by expansion*, that is by appending to an input matrix A new rows and columns. Namely, in [10, Section 12] the matrix A is embedded into a matrix $M = \begin{pmatrix} \eta I & 0 \\ B & A \end{pmatrix}$, and then A-preconditioning $M \rightarrow M + P$ is applied for $P = \begin{pmatrix} I \\ 0 \end{pmatrix} (0, F) = \begin{pmatrix} \eta I & F \\ B & A \end{pmatrix}$, $\eta \approx ||A||$, and weakly random matrices B and F scaled so that $||B|| \approx ||A||$ and $||F|| \approx ||A||$. Because of the latter rstriction on the values η, $||B||$, and $||F||$, embedding does not change the condition number $\mathrm{cond}\, A$ substantially, whereas the impact of this special A-preconditioning is

Table 8.3: $A = \text{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 64$, $r = 8$, $\text{cond}(A) = 9.223372e + 18$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	1.08361e+018	1.20196e+019	4.58727e+016	2.83277e+020	9.51630e-001
Random Sym	3.90744e+017	1.61882e+018	2.86030e+016	3.79442e+019	9.60910e-001
Unitary	2.29417e+017	6.12078e+017	2.06627e+016	9.75414e+018	9.71501e-001
Unitary Sym	2.60456e+017	1.07133e+018	1.93541e+016	2.70620e+019	9.71356e-001
Toeplitz	6.79930e+017	6.38405e+018	2.82636e+016	1.96062e+020	9.57215e-001
Toeplitz Sym	6.03878e+017	9.20230e+018	2.91564e+016	2.88241e+020	9.65704e-001

Table 8.4: $A = \text{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 64$, $r = 16$, $\text{cond}(A) = 9.223372e + 18$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	2.36668e+016	1.30529e+017	7.11146e+014	2.99944e+018	8.96245e-001
Random Sym	3.21429e+015	2.91459e+015	5.65028e+014	5.82154e+016	9.23034e-001
Unitary	1.98699e+016	5.29672e+016	9.30829e+014	6.63291e+017	8.93742e-001
Unitary Sym	5.42922e+015	1.63799e+016	6.94890e+014	3.70468e+017	9.15474e-001
Toeplitz	1.31361e+016	1.63651e+017	4.11882e+014	5.10288e+018	9.15238e-001
Toeplitz Sym	2.05219e+015	2.15409e+015	3.51954e+014	2.71915e+016	9.36637e-001

Table 8.5: $A = \text{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 64$, $r = 24$, $\text{cond}(A) = 9.223372e + 18$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	1.34201e+014	5.63648e+014	3.14062e+012	1.35806e+016	8.70034e-001
Random Sym	2.04270e+013	1.86455e+013	2.45158e+012	1.85291e+014	8.98305e-001
Unitary	6.63454e+014	1.05562e+016	4.69794e+012	3.18549e+017	8.63487e-001
Unitary Sym	2.90021e+013	3.34354e+013	4.41430e+012	7.09881e+014	8.88235e-001
Toeplitz	6.65849e+013	2.88842e+014	2.28106e+012	6.63841e+015	8.90152e-001
Toeplitz Sym	1.14082e+013	1.39032e+013	1.46568e+012	2.70185e+014	9.17782e-001

similar to the usual impact.

Preconditioning by expansion requires limited increase of the input size, but is error-free. Moreover, because of the special form of the generators $U = \begin{pmatrix} I \\ 0 \end{pmatrix}$ and $V = (0, F)$ of the APP, we can simplify the subsequent solution of the original ill conditioned problem.

Table 8.6: $A = \mathrm{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 64$, $r = 32$, $\mathrm{cond}(A) = 9.223372e + 18$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	1.03706e+012	7.84853e+012	2.29102e+010	1.64716e+014	8.35496e-001
Random Sym	1.02168e+011	8.45415e+010	1.30935e+010	8.59375e+011	8.69133e-001
Unitary	8.77737e+011	4.06730e+012	3.34894e+010	9.13614e+013	8.28184e-001
Unitary Sym	1.53717e+011	1.25814e+011	2.18668e+010	1.23496e+012	8.55484e-001
Toeplitz	3.94729e+011	2.80367e+012	1.05051e+010	7.98411e+013	8.63554e-001
Toeplitz Sym	5.26510e+010	4.92329e+010	6.64650e+009	4.19800e+011	8.95642e-001

Table 8.7: $A = \mathrm{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 64$, $r = 40$, $\mathrm{cond}(A) = 9.223372e + 18$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	3.64023e+009	2.07266e+010	1.04842e+008	5.14567e+011	7.88829e-001
Random Sym	5.21963e+008	4.20833e+008	8.05181e+007	3.87211e+009	8.25645e-001
Unitary	5.40394e+009	3.65762e+010	1.71588e+008	9.17457e+011	7.74924e-001
Unitary Sym	8.23576e+008	6.90037e+008	9.71479e+007	5.67202e+009	8.07622e-001
Toeplitz	1.33295e+009	5.23862e+009	5.37118e+007	1.20304e+011	8.21210e-001
Toeplitz Sym	2.73046e+008	3.57683e+008	3.58696e+007	4.89314e+009	8.58927e-001

Table 8.8: $A = \mathrm{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 64$, $r = 48$, $\mathrm{cond}(A) = 9.223372e + 18$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	1.95017e+007	8.99977e+007	3.50659e+005	1.30401e+009	7.06263e-001
Random Sym	2.51244e+006	1.99124e+006	3.10802e+005	1.80586e+007	7.52744e-001
Unitary	1.74074e+007	5.74201e+007	5.64628e+005	1.14934e+009	6.94290e-001
Unitary Sym	4.07143e+006	3.60756e+006	3.81003e+005	3.63847e+007	7.30504e-001
Toeplitz	5.38182e+006	1.68598e+007	2.01315e+005	2.82572e+008	7.52190e-001
Toeplitz Sym	1.32076e+006	1.67708e+006	1.31192e+005	2.79892e+007	7.96992e-001

Table 8.9: $A = \mathrm{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 128$, $r = 16$, $\mathrm{cond}(A) = 1.304382e + 19$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	9.00893e+017	3.68726e+018	9.44887e+016	7.83928e+019	9.49794e-001
Random Sym	7.54539e+017	3.69515e+018	6.76430e+016	8.14035e+019	9.55289e-001
Unitary	3.82162e+017	1.57830e+018	2.93430e+016	2.45520e+019	9.72428e-001
Unitary Sym	2.83650e+017	6.81346e+017	3.53846e+016	1.10999e+019	9.71706e-001
Toeplitz	1.52550e+018	2.65657e+019	6.31877e+016	8.36544e+020	9.54695e-001
Toeplitz Sym	5.22631e+017	2.89616e+018	4.47650e+016	8.68359e+019	9.60338e-001

Table 8.10: $A = \mathrm{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 128$, $r = 32$, $\mathrm{cond}(A) = 1.304382e + 19$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	4.43929e+016	2.19833e+017	1.85261e+015	4.32220e+018	8.85779e-001
Random Sym	4.41166e+015	2.08244e+015	1.22571e+015	1.69976e+016	9.18730e-001
Unitary	5.56929e+016	3.02392e+017	2.78328e+015	7.77269e+018	8.81699e-001
Unitary Sym	7.33220e+015	7.35583e+015	1.78143e+015	1.06167e+017	9.08649e-001
Toeplitz	2.34964e+016	8.49708e+016	1.04679e+015	1.59832e+018	9.00039e-001
Toeplitz Sym	2.83160e+015	1.88393e+015	5.80915e+014	2.05292e+016	9.31854e-001

Table 8.11: $A = \mathrm{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 128$, $r = 48$, $\mathrm{cond}(A) = 1.304382e + 19$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	3.26233e+014	1.85918e+015	1.15524e+013	4.14182e+016	8.52546e-001
Random Sym	2.84455e+013	1.41842e+013	7.50276e+012	1.14344e+014	8.91000e-001
Unitary	8.98233e+014	1.14687e+016	1.82448e+013	3.36247e+017	8.45462e-001
Unitary Sym	4.20903e+013	2.11494e+013	9.02804e+012	1.64270e+014	8.79872e-001
Toeplitz	1.49435e+014	5.70442e+014	4.08301e+012	8.70108e+015	8.73919e-001
Toeplitz Sym	1.57012e+013	1.34157e+013	2.57722e+012	1.83051e+014	9.11638e-001

Table 8.12: $A = \mathrm{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 128$, $r = 64$, $\mathrm{cond}(A) = 1.304382e + 19$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	1.55610e+012	4.05980e+012	7.26960e+010	5.89482e+013	8.11041e-001
Random Sym	1.51627e+011	7.43880e+010	3.78576e+010	6.24300e+011	8.58232e-001
Unitary	3.31336e+012	3.82383e+013	8.62790e+010	1.09007e+015	8.05163e-001
Unitary Sym	2.33668e+011	1.33690e+011	7.05725e+010	1.65499e+012	8.44506e-001
Toeplitz	7.32596e+011	4.46140e+012	2.58278e+010	1.22060e+014	8.42763e-001
Toeplitz Sym	7.79676e+010	6.07700e+010	1.35590e+010	6.35448e+011	8.84444e-001

Table 8.13: $A = \mathrm{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 128$, $r = 80$, $\mathrm{cond}(A) = 1.304382e + 19$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	3.23207e+010	7.67489e+011	3.19383e+008	2.42563e+013	7.57295e-001
Random Sym	7.65959e+008	4.29077e+008	1.93689e+008	3.98671e+009	8.11606e-001
Unitary	1.67399e+010	1.02946e+011	4.59358e+008	1.74678e+012	7.46439e-001
Unitary Sym	1.21236e+009	6.30760e+008	2.67350e+008	5.13002e+009	7.93357e-001
Toeplitz	4.44073e+009	4.06987e+010	1.18967e+008	1.23209e+012	7.92921e-001
Toeplitz Sym	3.73584e+008	3.32574e+008	7.11037e+007	5.43107e+009	8.45268e-001

Table 8.14: $A = \mathrm{diag}(\sqrt{2}^{128i/n})_{i=1}^{n}$, $n = 128$, $r = 96$, $\mathrm{cond}(A) = 1.304382e + 19$

APP	mean(cond C)	std(cond C)	min(cond C)	max(cond C)	Ratio
Random	2.95471e+007	8.97563e+007	1.47168e+006	1.55489e+009	6.73631e-001
Random Sym	3.73307e+006	1.99705e+006	1.12101e+006	2.70990e+007	7.32157e-001
Unitary	4.83564e+007	1.89337e+008	2.20168e+006	4.43300e+009	6.58656e-001
Unitary Sym	5.72358e+006	2.79523e+006	1.76388e+006	2.44323e+007	7.11635e-001
Toeplitz	1.21819e+007	5.33440e+007	5.71416e+005	9.04293e+008	7.20065e-001
Toeplitz Sym	1.72635e+006	1.61914e+006	3.39188e+005	3.06641e+007	7.78179e-001

References

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

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

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

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

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

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

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

[8] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning and Aggregation in Matrix Computations, *Computers and Math. with Applications*, **55**, **8**, 1870–1886, 2008.

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

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

[11] V. Y. Pan, G. Qian, Solving Homogeneous Linear Systems with Weakly Randomized Additive Preprocessing, Technical Report TR 2008009, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2008. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

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

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

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

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

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

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

[18] 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.

[19] 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.

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

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

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

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

[24] D. Spielman, S.-H. Teng, Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time, *Proc. 33rd Annual ACM Symposium on the Theory of Computing (STOC 2001)*, 296–305, 2001.

Full version available at http://math.mit.edu/ spielman/SmoothedAnalysis.

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

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

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

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

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