Numerically Safe Gaussian Elimination with No Pivoting *

Victor Y. Pan\textsuperscript{[1,2],[a]} and Liang Zhao\textsuperscript{[2],[b]}

\textsuperscript{[1]} Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA

\textsuperscript{[2]} Ph.D. Programs in Mathematics and Computer Science
The Graduate Center of the City University of New York
New York, NY 10036 USA

\[\textsuperscript{[a]}\] victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/

\[\textsuperscript{[b]}\] lzhao1@gc.cuny.edu

\textbf{Abstract}

Gaussian elimination with partial pivoting is performed routinely, millions times per day around the world, but partial pivoting (that is, row interchange of an input matrix) is communication intensive and has become the bottleneck of the elimination algorithm in the present day computer environment, in both cases of matrices of large and small size. Gaussian elimination with no pivoting as well as block Gaussian elimination are highly attractive alternatives. (Hereafter we use the acronyms GENP and BGE.) It has been proved decades ago that GENP and BGE are likely to proceed safely, that is, involving no division by 0 or inversion of singular blocks, if the input is pre-processed with random or even random structured matrix multipliers. Here and hereafter “likely” means with a probability 1 or close to 1. In 2015 we proved that, with Gaussian random multipliers (hereafter we say just Gaussian), both GENP and BGE are likely to be numerically safe, that is, are likely to be safe and to encounter no numerical stability problems. In all these results the multipliers are proven to be universal, that is, GENP and BGE with such randomized preprocessing

\begin{itemize}
  \item are safe with probability 1 if an input matrix is nonsingular and
  \item are likely to be numerically safe if an input matrix is also well-conditioned.
\end{itemize}

The latter assumptions about the input matrices is necessary for GENP, but not for BGE.

We prove some nontrivial positive and negative results about the universality of random structured preprocessing, including multiplicative and additive preprocessing and augmentation for GENP and BGE, but we also provide some new insights into the subject and motivate new policies of preprocessing, which are not universal but highly efficient according to the results of our extensive tests.

\textbf{2000 Math. Subject Classification:} 15A06, 15A52, 15A12, 65F05, 65F35

\textbf{Keywords:} Gaussian elimination; Pivoting; Block Gaussian elimination; Preconditioning; Random matrix algorithms; Linear systems of equations

\*Some results of this paper have been presented at the 17th Annual Conference on Computer Algebra in Scientific Computing (CASC'2014), September 10–14, 2015, Aachen, Germany (cf. [PZ15]).
1 Introduction

1.1 Gaussian elimination, pivoting policies, new bottleneck, safety and numerical safety

The history of Gaussian elimination can be traced back some 2000 years [G11]. Its modern version, called *Gaussian elimination with partial pivoting* (hereafter GEPP), is performed routinely, millions of times per day around the world, being a cornerstone for computations in linear algebra. For some samples of extensive and intensive applications of GEPP to Sciences, Technology, and Image Processing, the survey [DDF14] refers to fusion reactor modeling, aircraft design, acoustic scattering, antenna design, and radar cross-section studies and then recall that, e.g., in simulating fusion reactors, GEPP is applied to solving dense linear systems of equations with over half a million unknowns.

For an $n \times n$ input matrix, partial pivoting, that is, row interchange, involves only $(n-1)n/2$ comparisons, versus about $\frac{3}{2}n^3$ arithmetic operations involved into elimination. Clearly this is still a substantial share of the overall computational cost in the case of inputs of smaller size, but actually not only in this case. Progress in computer technology has made partial pivoting the bottleneck of present day Gaussian elimination. Here is a relevant citation from [BCD14]: “The traditional metric for the efficiency of a numerical algorithm has been the number of arithmetic operations it performs. Technological trends have long been reducing the time to perform an arithmetic operation, so it is no longer the bottleneck in many algorithms; rather, communication, or moving data, is the bottleneck.”

Pivoting is communication intensive and in modern computer environment takes quite a heavy toll: it interrupts the stream of arithmetic operations with foreign operations of comparison, involves book-keeping, compromises data locality, impedes parallelization of the computations, and increases communication overhead and data dependence.

According to [BDHT13], “pivoting can represent more than 40% of the global factorization time for small matrices, and although the overhead decreases with the size of the matrix, it still represents 17% for a matrix of size 10,000”. Because of the heavy use of GEPP, even its limited improvement is valuable.

*Gaussian elimination with no pivoting* (hereafter GENP) is an attractive alternative to the customary GEPP, but it can fail or produce a corrupted output more readily than GEPP. To specify this, call an input matrix unsafe and numerically unsafe for Gaussian elimination with or without pivoting if this algorithm applied to that matrix runs into a division by 0 or into numerical problems, respectively. A singular input matrix is unsafe for both GENP and GEPP, and an ill-conditioned matrix is numerically unsafe for both of them. Nonsingular matrices are safe for GEPP, and a very small fraction of them is unsafe for GENP.

Numerical safety or stability of Gaussian elimination is harder to study than just its safety, but since [W61] this study was alleviated by linking it to the growth factor $\rho = \max_{i,j} |a_{ij}|/\max_{i,j} |a_{ij}|$ in the PLUP’ factorization (with or without pivoting) of a matrix $A = (a_{ij})_{i,j} = PLUP'$ where $L = (l_{ij})_{i,j}$ and $U = (u_{ij})_{i,j}$ are lower and upper triangular factors, respectively, and $P$ and $P'$ are permutation matrices. In the case of GENP, both of $P$ and $P'$ turn into the identity matrix, $P = P' = I_n$. In the case of GEPP, $P' = I_n$, while $P$ is unrestricted allowing any row interchange. Both $P$ and $P'$ are unrestricted in Gaussian elimination with complete pivoting, hereafter referred to as GECP and allowing any interchange of rows and columns.

For the worst case input, $\rho$ is unbounded in the case of GENP, is as large as $2^{n-1}$ in the case of GEPP, and is in $O(n^{1+\log(\sqrt{n})}/2)$ in the case of GECP. Nevertheless GEPP has been universally preferred by the user in most cases. Its average growth factor is only $n^{2/3}$, which is just slightly larger than $n^{1/2}$ for GECP, and its pivoting is simpler than that of GECP, whereas GENP is considered numerically unsafe in practice and is little used, even though its average growth factor is just $n^{3/2}$, that is, only marginally larger than in the case of GEPP (cf. [TS90], [YC97]).

1 Gaussian elimination with rook pivoting [GL13], [H02] uses only $O(n^2)$ comparisons, like GEPP, and is as safe (stable) numerically as GECP, but the user still prefers GEPP because its numerical safety (stability) is satisfactory.
algorithms [GL13], but block Gaussian elimination (hereafter we keep using the acronym BGE) is impeded because pivoting cannot help us to avoid numerical stability problems for this algorithm. We can see in Section 2, however, that BGE is safe (resp., numerically safe) for any input matrix for which GENP is safe (resp., numerically safe). Thus we have yet another major motivation for studying safety and numerical safety of GENP.

1.2 Preprocessing versus pivoting

Preprocessing $A \rightarrow FA$, $A \rightarrow AH$, and $A \rightarrow FAH$, for nonsingular matrices $F$ and $H$, is a natural resource for supporting GENP and BGE because $A^{-1} = (FA)^{-1}F$, $A^{-1} = G(AG)^{-1}$, and $A^{-1} = G(FAG)^{-1}F$ and because empirically the matrices $FA$, $AH$, and $FAH$ tend to be safe and numerically safe for GENP and BGE when the matrix $A$ is nonsingular and well-conditioned and when the matrices $F$ and $H$ are random. The above observations and empirical data have recently convinced some leading experts in numerical matrix computations to implement GENP with randomized preprocessing (see [BBD12], [BDHT13], [BBBDD14], and [BLR15]). The resulting algorithm noticeably improves GEPP, even though preprocessing in the cited papers relies on ad hoc PRBT multipliers, which are recent amelioration by Baboulin of the ones from unpublished Technical Reports of 1995 by Parker and by Parker and Pierce. This encourages application of GENP with preprocessing, but leaves us with two challenges of (i) providing formal support for the observed empirical safety of GENP with random structured preprocessing and (ii) finding even more efficient multipliers and preprocessing policies. In this paper we advance in both directions.

Hereafter “likely” means “with a probability close to 1” and “Gaussian” stands for “standard Gaussian”. By virtue of our Theorems 3.2 and 3.3 combined, GENP pre-processed with a Gaussian multiplier is safe with probability 1 and is likely to be numerically safe. Moreover our analysis leads us to a practical recipe: according to Theorem 3.3 the probability of numerical failure of GENP with one-sided Gaussian preprocessing is small, but dramatically decreases if we apply two-sided Gaussian preprocessing.

The computational cost of generation and application of Gaussian multipliers is rather high, but we can apply sparse and structured multipliers. This derandomization of preprocessing is partly heuristic because Theorem 3.3 does not cover this application, but together with Theorem 3.4 motivates testing various multipliers successively or concurrently until success.

Namely Theorem 3.4 is dual to Theorem 3.3: it proves that GENP is safe with probability 1 and is likely to be numerically safe for a Gaussian input matrix $A$ pre-processed with any orthogonal or just nonsingular and well-conditioned multiplier. Consequently GENP is safe and numerically safe with such a multiplier for the input matrix $A$ average under the Gaussian probability distribution.

With which multipliers we have greater chances for quick success? Generally the chances are the same for any or two orthogonal $n \times n$ multipliers $H$ and $\hat{H}$ simply because $\hat{H} = QH$ for the orthogonal matrix $Q = HH^T$. Hence GENP applied to a matrix $A$ post-processed with the multiplier $\hat{H}$ fails or fails numerically if and only if it does so when it is applied to the matrix $AQ$ post-processed with the multiplier $H$.

We do not delve into the huge subject of the seeking most promising multipliers for preprocessing various specific input classes for GENP, but focus on the selection of orthogonal or nonsingular and well-conditioned multipliers generated and applied to a generic or structured matrix $A$ at a lower computational cost. In Section 4 we describe some promising classes of such multipliers and in Section 7 demonstrate their efficiency with extensive numerical tests.

It is of course desired to extend Theorem 3.3 to the case where Gaussian multipliers are replaced by some random sparse or structured multipliers. Such extensions are well-known for the task of supporting safe GENP, and in Section 5 we prove the strongest result of this kind: GENP is safe with probability 1 universally, that is, for any nonsingular input matrix pre-processed on the left or on the right with a Gaussian circulant multiplier, but no similar results are known that would support numerically safe GENP. Moreover, we prove that GENP fails numerically for a specific unitary input pre-processed with any (and therefore also with random) circulant multiplier (see

---

2Here and hereafter “PRBT” stands for “Partial Random Butterfly Transforms”.

3
Section 5.2), although in our tests GENP was consistently safe and numerically safe for various classes of input matrices pre-processed with one-sided random circulant multipliers.

Can we find random sparse or structured preprocessing with which GENP is likely to be numerically safe universally (for all nonsingular and well-conditioned input matrices)? In Section 4 we describe such preprocessing and in Section 7 support our formal proof with the results of our tests, although in both cases we apply random structured augmentation or additive preprocessing rather than random multipliers.

1.3 Our results versus related work

In this paper we reexamine the state of the art, the role of randomization, and the benefits of using sparse and structured multipliers. We supply formal probabilistic analysis based on our nonstandard and more general techniques of analysis. Our techniques are much simpler (the proof of our basic Theorem 3.3 occupies just ten lines).

The results of our analysis are consistent with the ones of the customary study in [PQY15] based on estimating the growth factor of LU factorization, but unlike that study, we cover BGE, Gaussian elimination with augmentation and additive preprocessing of the input. Moreover, the classical results of [YC97] have been only proven for Gaussian preprocessing for average input, whereas we prove universal efficiency of Gaussian preprocessing (for any input), efficiency of any nonsingular and well-conditioned preprocessing for the average input, and made specific estimates that show benefits of two-sided preprocessing.

We nontrivially prove that GENP with one-sided circulant preprocessing is universally safe with probability 1 under Gaussian randomization, but is never numerically unsafe for a specific input. We show, however, that it is likely to be numerically safe under random augmentation and additive preprocessing.

Our classes of sparse and structured multipliers, heuristic policies of their generation consistently supported by our extensive tests, and formal support for our recipes of combining two-sided preprocessing and block preprocessing in Section 3.4 can be of potential interest for the implementation of GENP.

1.4 Organization of the paper

In the rest of this section (and also in the Appendix) we cover some definitions. We devote Section 2 to BGE and its link to GENP. In Section 3 we study policies of preprocessing, partly for symbolic, but mostly for numerical GENP and BGE. In Section 4 we describe some efficient multipliers. In Section 5 we cover randomized circulant preprocessing for GENP and BGE, in particular (the contribution of the second author) under the symbolic model of computing. We devote Section 6 to augmentation and additive preprocessing and Section 7 to our numerical experiments. In the Appendix we cover some auxiliary subjects, in particular, random, DFT and circulant matrices. A number of our results stated under Gaussian sampling and preprocessing can be restated under the uniform probability distribution over a finite set (cf. Theorem B.1).

1.5 Some basic definitions

Hereafter “likely” and “unlikely” mean “with a probability close to 1” and, respectively, “to 0”. “Flop” stands for “floating point arithmetic operation”. We call an \( m \times n \) matrix Gaussian and denote it \( G_{m,n} \) if all its entries are i.i.d. standard Gaussian variables. \( G^{m \times n} \), \( \mathbb{R}^{m \times n} \), and \( \mathbb{C}^{m \times n} \) denote the classes of \( m \times n \) Gaussian, real and complex matrices, respectively. In order to simplify our presentation, we assume dealing with real matrices, unless we specify otherwise, but we can readily extend our study to the complex case.

In addition to the definitions below (cf. [GL13]), we recall some definitions and basic properties of general, random, circulant, and \( f \)-circulant matrices in the Appendix.

1. \( I_g \) is a \( g \times g \) identity matrix. \( O_{k,l} \) is the \( k \times l \) matrix filled with zeros.
2. \((B_1 \mid B_2 \mid \cdots \mid B_k)\) is a block vector of length \(k\), and \(\text{diag}(B_1, B_2, \ldots, B_k)\) is a \(k \times k\) block diagonal matrix, in both cases with blocks \(B_1, B_2, \ldots, B_k\).

3. \(W_{k,l}\) denotes the \(k \times l\) leading (that is, northwestern) block of an \(m \times n\) matrix \(W\) for \(k \leq m\) and \(l \leq n\). A matrix is strongly nonsingular if all its square leading blocks are nonsingular.

4. \(R(W)\), \(W^T\) and \(W^H\) denote its range (that is, column span), transpose and Hermitian transpose, respectively. \(W^H = W^T\) for a real matrix \(W\).

5. An \(m \times n\) matrix \(W\) is called unitary (in the real case also and preferably orthogonal) if \(W^H W = I_m\) or \(WW^H = I_n\).

6. \(Q(W)\) denotes the matrix obtained by means of column orthogonalization of a matrix \(W\), followed by the deletion of the columns filled with zeros (cf. [GL13, Theorem 5.2.3]).

7. \(\|W\|\) and \(\|W\|_F\) denote its spectral and Frobenius norms, respectively.

**2 BGE and GENP**

For a nonsingular \(2 \times 2\) block matrix \(A = \begin{bmatrix} B & C \\ D & E \end{bmatrix}\) of size \(n \times n\) with nonsingular \(k \times k\) pivot block \(B = A_{k,k}\), define \(S = S(A_{k,k}, A) = E - DB^{-1}C\), the Schur complement of \(A_{k,k}\) in \(A\), and the block factorizations,

\[
A = \begin{pmatrix} I_k & O_{k,r} \\ DB^{-1} & I_r \end{pmatrix} \begin{pmatrix} B & O_{k,r} \\ O_{r,k} & S \end{pmatrix} \begin{pmatrix} I_k & B^{-1}C \\ O_{k,r} & I_r \end{pmatrix} \tag{2.1}
\]

and

\[
A^{-1} = \begin{pmatrix} I_k & -B^{-1}C \\ O_{k,r} & I_r \end{pmatrix} \begin{pmatrix} B^{-1} & O_{k,r} \\ O_{r,k} & S^{-1} \end{pmatrix} \begin{pmatrix} I_k & O_{k,r} \\ -DB^{-1} & I_r \end{pmatrix}. \tag{2.2}
\]

We readily verify that \(S^{-1}\) is the \((n-k) \times (n-k)\) trailing (that is, southeastern) block of the inverse matrix \(A^{-1}\), and so the Schur complement \(S\) is nonsingular since the matrix \(A\) is nonsingular.

Factorization (2.2) reduces the inversion of the matrix \(A\) to the inversion of the leading block \(B\) and its Schur complement \(S\), and we can recursively reduce the inversion task to the case of the leading blocks and Schur complements of decreasing sizes as long as the leading blocks are nonsingular. After sufficiently many recursive steps of this process of BGE, we only need to invert matrices of small sizes, and then we can stop the process and apply a selected block box inversion algorithm, e.g., based on orthogonalization.

If we limit the number of recursive steps, we arrive at BGE dealing with large blocks and can use the benefits of block matrix algorithms, and if we keep recursive partitioning, then BGE eventually turn into GENP.

Namely, in \(
\lfloor \log_2(n) \rfloor
\) recursive steps all pivot blocks and all other matrices involved into the resulting factorization turn into scalars, all matrix multiplications and inversions turn into scalar multiplications and divisions, and we arrive at a complete recursive factorization of the matrix \(A\). If \(k = 1\) at all recursive steps, then the complete recursive factorization (2.2) defines GENP.

Moreover, any complete recursive factorizations turns into GENP up to the order in which we consider its steps. This follows because at most \(n-1\) distinct Schur complements \(S = S(A_{k,k}, A)\), for \(k = 1, \ldots, n-1\), are involved in all recursive block factorization processes for \(n \times n\) matrices \(A\), and so we arrive at the same Schur complement in a fixed position via GENP and via any other recursive block factorization (2.1).

Hence we can interpret factorization step (2.1) as the block elimination of the first \(k\) columns of the matrix \(A\), which produces the matrix \(S = S(A_{k,k}, A)\). If the dimensions \(d_1, \ldots, d_r\) and \(\tilde{d}_1, \ldots, \tilde{d}_r\) of the pivot blocks in two block elimination processes sum to the same integer \(k\), that is, if \(k = d_1 + \cdots + d_r = \tilde{d}_1 + \cdots + \tilde{d}_r\), then both processes produce the same Schur complement \(S = S(A_{k,k}, A)\). The following results extend this observation.
Theorem 2.1. In the recursive block factorization process based on (2.1), the diagonal block and its Schur complement in every block diagonal factor is either a leading block of the input matrix $A$ or the Schur complement $S(A_{h,h}, A_{k,k})$ for some integers $h$ and $k$ such that $0 < h < k \leq n$ and $S(A_{h,h}, A_{k,k}) = (S(A_{h,h}, A))_{h,h}$.

Corollary 2.1. The complete recursive block factorization process based on equation (2.1) can be computed by involving no singular pivot blocks (and, in particular, no pivot elements vanish) if and only if the input matrix $A$ is strongly nonsingular.

Proof. Combine Theorem 2.1 with the equation $\det A = (\det B) \det S$, implied by (2.1). \hfill \square

Remark 2.1. One can benefit from application of BGE to computations with structured matrices. E.g., the MBA superfast algorithm, by Morf [M74], [M80] and by Bitmead and Anderson [BA80], runs in nearly linear arithmetic time in the case of Toeplitz and Toeplitz-like inputs. This algorithm is precisely the recursive BGE, accelerated by means of exploiting the Toeplitz-like structure of the input matrix throughout the recursive process of BGE (cf. Remark D.5). The algorithm has been further extended to computations with structured matrices of other classes [P01, Chapter 5].

3 Preprocessing for GENP and BGE

In this section, $A$ denotes a nonsingular $n \times n$ matrix.

3.1 Verification by action and via formal study of the leading blocks

Suppose that the vector $y = Ab$ satisfies pre-processed linear systems $AHy = b$ and $FAHy = Fb$. Then the vector $x = Hy$ for $y = A^{-1}b$ satisfies both linear systems $Ax = b$ and $FAX = Fb$. We are going to study such preprocessing $A \rightarrow AH$ for GENP and BGE with random and fixed post-multipliers $H$. Our analysis is immediately extended to the preprocessing maps $A \rightarrow FA$, $A \rightarrow FAH$, and $A \rightarrow FAF^T$.

In order to verify whether a fixed preprocessing works, one can proceed by action, that is, by testing whether pre-processed GENP or BGE solves a linear system of equations $Ax = b$, but in our probabilistic analysis we estimate the probability of success in circuited ways, based on Theorem 2.1 and Corollary 2.1.

3.2 Criteria of safety and numerical safety

We call GENP and BGE safe if they proceed to the end with no divisions by 0.

Corollary 2.1 implies the following result for computations in any field (cf. Remark 3.1).

Theorem 3.1. GENP is safe if and only if the input matrix is strongly nonsingular.

Next assume that GENP and BGE are performed numerically, with rounding to a fixed precision, e.g., the IEEE standard double precision. Then extend the concept of safe GENP and BGE to numerically safe GENP and BGE by requiring that the input matrix be strongly nonsingular and strongly well-conditioned, that is, that the matrix itself and all its square leading blocks be nonsingular and well-conditioned.

Any inversion algorithm for a nonsingular matrix is highly sensitive to the input and rounding errors if the matrix is ill-conditioned [GL13]. GENP explicitly or implicitly involves the inverses of all its square leading blocks, and we arrive at the following Criterion of Numerical Safety of GENP implied by [PQZ13, Theorem 5.1]:

GENP applied to a strongly nonsingular matrix is highly sensitive to the input and rounding errors if and only if some of the square leading blocks are ill-conditioned.

Let us restate this criterion in the form similar to Theorem 3.1.

Theorem 3.2. GENP is safe and numerically safe if and only if the input matrix is strongly nonsingular and strongly well-conditioned.
Remark 3.1. BGE is safe if so does GENP. Likewise BGE is safe numerically if so does GENP. Thus our proofs of safety and numerical safety of GENP apply to BGE. The converse is not true: GENP fails (resp. fails numerically) if any square leading block of the input matrix is singular (resp. ill-conditioned), but BGE may bypass this block and be safe (resp. numerically safe).

3.3 GENP with Gaussian preprocessing is likely to be safe and numerically safe

Theorem 3.3. Assume that we are given a nonsingular and well-conditioned $n \times n$ matrix $A$ and a pair of $n \times n$ Gaussian matrices $F$ and $H$ and let $\nu_{k,k}^+, \nu_{k,n}^+$, and $\nu_{n,k}^+$ denote random variables of Appendix B. Then

(i) the matrices $FA$, $AH$, and $FAH$ are strongly nonsingular with a probability 1,

(ii) $\|((AH)_{k,k})^+\| \leq \nu_{k,k}^+ \|A_{k,n}\| \leq \nu_{k,k}^+ \|A^+\|$, $\|((FA)_{k,k})^+\| \leq \nu_{k,k}^+ \|A_{n,k}\| \leq \nu_{k,k}^+ \|A^+\|$, and

(iii) $\|((FAH)_{k,k})^+\| \leq \nu_{k,n}^+ \nu_{n,k}^+ \min\{|\|A_{k,n}\||, \|A_{n.k}\|\} \leq \nu_{k,n}^+ \nu_{n,k}^+ \|A^+\|$.

Proof. Part (i) follows from part (ii) of Theorem B.1.

Hereafter pairs of subscripts $p,q$ shows matrix size $p \times q$. The proof of part (ii) is similar for both products $AH$ and $FA$; we only cover the case of the former one. Note that

$$(AH)_{k,k} = A_{k,n}H_{n,k}$$

and substitute compact SVD

$$A_{k,n} = S_{k,k}\Sigma_{k,k}T_{n,k}^T$$

where $\Sigma_{k,k}$ is a diagonal matrix and $S_{k,k}$ and $T_{n,k}$ are orthogonal matrices. Obtain

$$(AH)_{k,k} = S_{k,k}\Sigma_{k,k}T_{n,k}^TH_{n,k} = S_{k,k}\Sigma_{k,k}G_{k,k}$$

where $G_{k,k} = T_{n,k}^TH_{n,k}$ is a $k \times k$ Gaussian matrix by virtue of Lemma B.1. Deduce that

$$((AH)_{k,k})^+ = G_{k,k}^+\Sigma_{k,k}^{-1}S_{k,k}^T.$$ 

Hence

$$\|((AH)_{k,k})^+\| = \|G_{k,k}^+\Sigma_{k,k}^{-1}\| \leq \|G_{k,k}^+\| \|\Sigma_{k,k}^{-1}\|.$$ 

Substitute the equations $\|G_{k,k}^+\| = \nu_{k,k}^+$ and $\|\Sigma_{k,k}^{-1}\| = \|A_{k,n}\| \leq \|A^+\|$ and obtain part (ii).

Let us prove part (iii). Note that

$$(FAH)_{k,k} = F_{k,n}AH_{n,k}$$

and substitute compact SVD

$$A_{k,n} = S_{k,k}\Sigma_{k,k}T_{n,k}^T.$$ 

Obtain that

$$(FAH)_{k,k} = F_{k,n}S_{k,k}\Sigma_{k,k}T_{n,k}^TH_{n,k} = G'\Sigma_{k,k}G''$$

where

$$G' = G_{k,n} = F_{k,n}S_{k,k}$$

and

$$G'' = G_{n,k} = T_{n,k}^TH_{n,k}$$

are Gaussian matrices by virtue of Lemma B.1.

Substitute compact SVDs

$$G' = S_{G',k,k}\Sigma_{G',k,k}T_{G',k,n}^T$$

and

$$G'' = S_{G'',n,k}\Sigma_{G'',n,k}T_{G'',k,n}^T$$

and obtain

$$(FAH)_{k,k} = S_{G',k,k}BT_{G'',k,n}^T$$

for $B = \Sigma_{G',k,k}W\Sigma_{G'',k,k}$ and $W = T_{G',k,n}^T\Sigma_{k,k}S_{G'',n,k}$. 

7
Observe that the latter equation is compact SVD, and so

\[ \|W^+\| = \|\Sigma_{k,k}^+\| = \|A_{k,n}^+\| \leq \|A^+\|. \]

Furthermore, clearly

\[ \|((FAH)_{k,k})^+\| = \|B^+\| \]

because \(G'_{k,k}\) and \(T_{G''n,k,n}\) are square orthogonal matrices.

Now observe that

\[ \|B^+\| \leq \|\Sigma_{G',k,k}^+\| \|W^+\| \|\Sigma_{G'',k,k}^+\| \]

because \(\Sigma_{G',k,k}^+\) and \(\Sigma_{G'',k,k}^+\) are square diagonal matrices. Note that

\[ \|\Sigma_{G',k,k}^+\| = \|G^+\| = \nu_{k,n}^+ \quad \text{and} \quad \|\Sigma_{G'',k,k}^+\| = \|G''^+\| = \nu_{n,k}^+. \]

Combine the above equations for the norms \((FAH)_{k,k}\), \(\|W^+\|\), \(\|\Sigma_{G',k,k}^+\|\) and \(\|\Sigma_{G'',k,k}^+\|\) and the above bound on the norm \(\|B^+\|\) \(\|((FAH)_{k,k})^+\|\) and obtain

\[ \|((FAH)_{k,k})^+\| \leq \nu_{k,n}^+ \nu_{n,k}^+ \|A_{k,n}^+\|. \]

By applying this estimate to the norm \(\|((FAH)_{k,k}^T)^+\|\) deduce that

\[ \|((FAH)_{k,k})^+\| = \|((FAH)_{k,k}^T)^+\| \leq \nu_{k,n}^+ \nu_{n,k}^+ \|A_{k,n}^+\|, \]

thus completing the proof of part (iii). \(\square\)

**Remark 3.2.** [PQY15, Corollary 5.2] provides a correct, although very long proof of part (ii) of Theorem 3.3 in the case of post-multiplication by \(H\), but states the result with an error, by writing \(\nu_{k,n}\) instead of the correct \(\nu_{k,k}\). We fix the statement and include a short proof for the sake of completeness of our presentation. Part (iii) of the theorem is new and reveals important benefits of using two-sided rather than one-sided Gaussian preprocessing (see the next subsection).

Theorems 3.1, 3.2, 3.3, B.2, and B.3 together imply the following result.

**Corollary 3.1.** GENP is safe with probability 1 and is likely to be numerically safe if it is applied to the matrices \(FA\), \(AH\) and \(FAH\) where \(A\) is a nonsingular and well-conditioned \(n \times n\) matrix and \(F\) and \(H\) are Gaussian \(n \times n\) matrices.

**3.4 Benefits of using two-sided Gaussian preprocessing and block preprocessing for GENP**

Combining Theorem 3.3 with the Central Limit Theorem implies that GENP is likely to be safe and numerically safe if it is applied to a nonsingular and well-conditioned matrix \(A\) pre-processed with multipliers \(F_i\) and/or \(H_i\) recursively sampled from two sufficiently large sets \(F = \{F_i\}\) and \(H = \{H_i\}\), of nonsingular and well-conditioned \(n \times n\) matrices generated independently of each other and of the matrix \(A\). We would continue the process until success or until we use all matrices in the sets \(F\) and/or \(H\), and we call this rule **Policy 1**.

Comparing the estimates of parts (ii) and (iii) of Theorem 3.3 shows that shifting from one-sided to two-sided Gaussian preprocessing of GENP is likely to enhance its power. Indeed suppose that the integer \(n - k\) is not small, say, exceeds 3. Then Theorem B.3 implies that the norm \(\nu_{k,k}^+\) is likely to exceed substantially the product \(\nu_{k,n}^+ \nu_{n,k}^+\), that is, GENP is substantially safer numerically with two-sided Gaussian preprocessing until the size of the GENP input has been reduced to \(4 \times 4\). At this point, however, it is inexpensive to complete Gaussian elimination by means of a reliable method such as GEPP, GECP, or orthogonalization.

Partitioning GENP into stages also enables us to use additional benefits of block matrix algorithms and, rather surprisingly, of saving random variables and flops. E.g., first pre-process the \(k \times k\) leading block of the input matrix for a proper integer \(k < n\) by using \(n \times k\) Gaussian multipliers. Having factored this block, decrease the input size from \(n\) to \(n - k\) and then re-apply Gaussian preprocessing. Already by using such a two-step block preprocessing for \(k = n/2\), we save 1/4 of all random variables and 3/8 of arithmetic operations involved.
3.5 GENP with any nonsingular and well-conditioned preprocessing is safe and numerically safe on the average input

**Theorem 3.4.** Assume that we are given a Gaussian $n \times n$ matrix $A$ and a pair of $n \times n$ nonsingular and well-conditioned matrices $F$ and $H$ and let $\nu_{k,k}^+$ denotes a random variable of Appendix B. Then

(i) the matrices $FA$, $AH$, and $FAH$ are strongly nonsingular with a probability 1,

(ii) $\|((AH)_{k,k})^+\| \leq \nu_{k,k}^+ \|H_{n,k}^+\|$, $\|((FA)_{k,k})^+\| \leq \nu_{k,k}^+ \|F_{k,n}^+\|$, and $\|((FAH)_{k,k})^+\| \leq \nu_{k,k}^+ \|H_{n,k}^+\| \leq \|F^+\| \|F_{k,k}^+\| \|H^+\|$.

Proof. The proof is similar to the proof of Theorem 3.3. We only specify the proof of part (iii). Recall that

$$(FAH)_{k,k} = F_{k,n}AH_{n,k},$$

substitute compact SVDs

$$F_{k,n} = S_{F,k,k}G_{k,k}H_{n,k} = S_{F,k,k}G_{k,k}H_{n,k}$$

and obtain that

$$(FAH)_{k,k} = S_{F,k,k}G_{k,k}S_{F,k,k}G_{k,k}H_{n,k}H_{n,k}$$

where $G_{k,k} = T_{F,n,k}^TAS_{H,n,k}$ is a $k \times k$ Gaussian matrix by virtue of of Lemma B.1. The matrices $S_{F,k,k}$, $G_{k,k}$, $H_{n,k}$, $T_{F,n,k}$ are nonsingular by assumption, and so is the matrix $G_{k,k}$ with probability 1 by virtue of part (ii) of Theorem B.1. Hence with probability 1

$$(FAH)_{k,k}^+ = T_{H,k,k}^T(\Sigma_{H,k,k}^{-1}G_{k,k}^{-1}S_{F,k,k}^{-1})S_{F,k,k}^T$$

and

$$\|((FAH)_{k,k})^+\| \leq \|T_{H,k,k}\| \|\Sigma_{H,k,k}^{-1}\| \|G_{k,k}^{-1}\| \|S_{F,k,k}^{-1}\| \|S_{F,k,k}^T\|.$$ 

Substitute $\|T_{H,k,k}\| = ||S_{F,k,k}^T|| = ||S_{F,k,k}^T|| = ||(F_{k,k}^+)\|$, $||\Sigma_{F,k,k}^{-1}\| = ||H_{n,k}^+\|$, $||G_{k,k}^{-1}\| = \nu_{k,k}^+$.

Theorems 3.2, 3.3, B.2, and B.3 together imply the following result.

Corollary 3.2. GENP is safe and numerically safe when it is applied to the matrices $FA$, $AH$ and $FAH$ where $A$ is average $n \times n$ matrix defined under the Gaussian probability distribution and $F$ and $H$ are $n \times n$ nonsingular and well-conditioned matrices.

3.6 Heuristic amelioration of Policy 1 of preprocessing for GENP

Heuristically one can hope that Policy 1 of Section 3.4 should succeed after a small number of application of GENP with sparse or structured multipliers generated by independent parameters. This hope has no formal support from Theorem 3.3, and for any pair of multipliers $F$ we can readily exhibit bad nonsingular and well-conditioned inputs $A$ for which GENP applied to the matrix $FA$ fails numerically, and similarly for GENP applied to the matrices $AH$ and $FAH$.

By virtue of Theorem 3.4, however, average $n \times n$ matrix $A$ and therefore most of $n \times n$ matrices are not bad in the above sense for any nonsingular and well-conditioned multipliers $F$, $H$ or pairs of such $F$ and $H$. Our extensive tests showed high efficiency of one-sided multipliers in such an application of Policy 1, in good accordance with Theorem 3.4, but one can prefer the two-sided preprocessing $A \rightarrow FAF^T$ for symmetric input matrices (our Theorems 3.3 and 3.4 hold unchanged for $H = F^T$).

In the next section we elaborate upon some promising choices of multipliers. Next we comment on some heuristic variations of Policy 1 of their generation towards accentuating their power.

Clearly the product of two sparse matrices has good chances to have singular square leading blocks, and so one can be cautious about preprocessing a sparse matrix with sparse multipliers. As a partial remedy we evenly distribute nonzero entries throughout a sparse multiplier in order to exclude undesirable blocks of large sizes filled with zeros.

Products of such multipliers with a sparse matrix $A$ can still have singular leading square blocks, and for a more radical means we can apply dense structured multipliers or the following heuristic
amelioration of Policy 1: if GENP has failed numerically for two matrices $AH_1$ and $AH_2$, then apply it to the sum $AH_3 = AH_1 + AH_2$ or more generally to $AH_3 = c_1 AH_1 + c_2 AH_2$ for two scalars $c_1$ and $c_2$. This recipe avoids generation of a new multiplier $H_3$ and replaces its multiplication with the matrix $A$ by a matrix addition. In our tests in Section 7, GENP has consistently succeeded where we simplified this variant of Policy 1 by skipping application of one-sided preprocessing with all multipliers, but applying it just with their sum.

4 Generation of Efficient Multipliers

4.1 What kind of multipliers do we seek?

Trying to support safe and numerically safe GENP we seek multipliers with the following properties:

1. Multipliers $F$ and $H$ must be nonsingular and well-conditioned.
2. The cost of the computation of the product $FA$, $AH$, $FAH$ or $FAF^H$ should be small.
3. Random multipliers should be generated by using fewer random variables.
4. For structured input matrices, the multipliers should have consistent structure.
5. The multipliers should enable GENP to produce accurate output with a probability close to 1 even with using no iterative refinement, at least for the inputs of reasonably small sizes.

These rules give general guidance but are a subject to trade-off: e.g., by filling multipliers with values 0, 1, and $-1$, we save flops when we compute their products with the input matrix $A$, but we increase the chances for success of our preprocessing if instead we fill the multipliers with random variables.

Remark 4.1. Property 5 was satisfied in our tests for most of our multipliers (unlike the PRBT multipliers of [BDHT13] and [DDF14]). It is important for $n \times n$ input matrices of smaller sizes: refinement iteration involves $O(n^2)$ flops versus cubic cost of $\frac{2}{3}n^3$ flops, involved in Gaussian elimination, but for small $n$ quadratic cost of refinement can make up a large share of the overall cost. A single refinement iteration was always sufficient (and usually was not even needed) in our tests in order to match or to exceed the output accuracy of GEPP (see also similar empirical data in [PQZ13], [BDHT13], [DDF14], and [PQY15] and see [H02, Chapter 12], [GL13, Section 3.5.3], and the references therein for detailed coverage of iterative refinement).

4.2 Some definitions

1. $P$ denotes an $n \times n$ permutation matrix.
2. $D$ denotes a unitary or real orthogonal diagonal matrix $\text{diag}(d_i)_{i=0}^{n-1}$, with fixed or random diagonal entries $d_i$ such that $|d_i| = 1$ for all $i$, and so each of $n$ entries $d_i$ lies on the unit circle $\{x : |z| = 1\}$, being either nonreal or $\pm 1$.
3. Define a Givens rotation matrix $G(i, j, \theta)$ (cf. [GL13]): for two integers $i$ and $j$, $1 \leq i < j \leq n$, and a fixed or random real $\theta$, $0 \leq \theta \leq 2\pi$, replace the submatrix $I_2$ in the $i$th and $j$th rows and columns of the identity matrix $I_n$ by the matrix $\begin{pmatrix} c & -s \\ s & c \end{pmatrix}$ where $c = \cos \theta$, $s = \sin \theta$, $c^2 + s^2 = 1$.
4. Define a Householder reflection matrix $I_n - \frac{2hh^T}{h^T h}$ by a fixed or random vector $h$ (cf. [GL13]).

We also use a DFT matrix $\Omega_n$ and transforms DFT, DFT($n$) and IDFT($n$) of Appendix C as well as circulant and $f$-circulant matrices of Appendix D. Besides unitary and orthogonal diagonal matrices $D$, defined above, we use a nonsingular and well-conditioned diagonal matrix $\text{diag}(\pm 2^{b_i})$ in Family 3 of Section 7, for random integers $b_i$ uniformly chosen from 0 to 3.
4.3 Dense structured multipliers: bidiagonal inverses, chains of Givens rotations, pairs of Householder reflections, and $f$-circulant matrices

**FAMILY 1.** The inverses of bidiagonal matrices:

\[ H = (I_n + DZ)^{-1} \text{ or } H = (I_n + Z^TD)^{-1} \]

for a diagonal matrix $D$ and the down-shift matrix $Z$ of Appendix D.

We can randomize a matrix $H$ by choosing $n - 1$ random diagonal entries of the matrix $D$ (whose leading entry makes no impact on $H$) and can pre-multiply it by a vector by using $2n - 1$ flops.

\[ ||H|| \leq \sqrt{n} \] because nonzero entries of the triangular matrix $H = (I_n + DZ)^{-1}$ have absolute values 1, and clearly $||H^{-1}|| = ||I_n + DZ|| \leq \sqrt{2}$. Hence $\kappa(H) = ||H|| ||H^{-1}||$ (the spectral condition number of $H$) cannot exceed $\sqrt{2n}$ for $H = (I_n + DZ)^{-1}$, and likewise for $H = (I_n + Z^TD)^{-1}$.

**FAMILY 2.** Chains of scaled and permuted Givens rotations with a DFT factor. A permutation matrix $P$ and a sequence of angles $\theta_1, \ldots, \theta_{n-1}$ together define a permuted chain of Givens rotations

\[ G(\theta_1, \ldots, \theta_{n-1}) = P \prod_{i=1}^{n-1} G(i, i + 1, \theta_i) \text{ for } n = 2^k. \]

Combine two such chains $G_1$ and $G_2$ with scaling and DFT into the following dense unitary matrix,

\[ H = \frac{1}{\sqrt{n}} D_1 G_1 D_2 G_2 D_3 \Omega_n \]

(cf. [HMT11, Section 4.6]), for diagonal matrices $D_1$, $D_2$ and $D_3$. This matrix can be multiplied by a vector by using about $10n$ flops plus the cost of performing DFT. By randomizing diagonal, permutation and Givens rotation matrices we can involve up to $7n - 2$ random variables.

**FAMILY 3.** Pairs of scaled and permuted Householder reflections with a DFT factor. Define the unitary matrix

\[ H = \frac{1}{\sqrt{n}} D_1 R_1 D_2 R_2 D_3 \Omega_n \]

where $D_1$, $D_2$, and $D_3$ are diagonal matrices, $R_1$ and $R_2$ are Householder reflections, and $\Omega_n$ is a DFT matrix of Appendix C. This matrix can be multiplied by a vector by using about $7n$ flops plus the cost of performing DFT. By randomizing diagonal, permutation and Householder reflection matrices we involve up to $7n$ random variables.

**FAMILY 4.** $f$-circulant matrices of Appendix D. $H = Z_f(\nu) = \sum_{i=0}^{n-1} \nu_i Z_f^i$, for the matrix $Z_f$ of $f$-circulant shift, defined by a scalar $f \neq 0$ and its first column vector $\nu = (\nu_i)_{i=0}^{n-1}$. By randomizing this vector we can involve up to $n$ random variables, and then Theorem D.2 and [PSZ15] enable us to bound the condition number $\kappa(H)$ of the matrix $Z_f$ by virtue of Theorem D.1, $Z_f(\nu) = (FD_f)^{-1} DGF_f$ where $D$ is a diagonal matrix, $D_f = \text{diag}(f_i)_{i=0}^{n-1}$, and $\Omega_n$ is the DFT matrix of Appendix D. Based on this expression, we can multiply the matrix $H$ by applying two DFT(n), an IDFT(n), and additionally $n + 2\delta_f n$ multiplications and divisions where $\delta_f = 0$ if $f = 1$ and $\delta_f = 1$ otherwise.

4.4 Sparse ARSPH matrices based on Hadamard’s processes

**FAMILY 5.** A $2^k \times 2^k$ Abridged Recursive Scaled and Permutated Hadamard (ARSPH) matrix $H = H_{2^k,d}$ of depth $d$ has $q = 2^d$ nonzeros entries in every row and in every column, for a fixed integer $d$, $1 \leq d \leq k$. Hence such a matrix is sparse unless $k - d$ is a small integer.

A special recursive structure of such a matrix allows highly efficient parallel implementation of its pre-multiplication by a vector (cf. Remark 4.2).

We recursively define matrices $H_{2^{h+1},d}$ for $h = k - d, \ldots, k - 1$, as follows:

\[ H_{2^{h+1},d} = D_{2^{h+1}} P_{2^{h+1}} \begin{pmatrix} H_{2^h,d} & H_{2^h,d} \\ H_{2^h,d} & -H_{2^h,d} \end{pmatrix} \tilde{P}_{2^{h+1}} D_{2^{h+1}}, \quad H_{2^{k-d},d} = \begin{pmatrix} I_{2^d} & I_{2^d} \\ I_{2^d} & -I_{2^d} \end{pmatrix}. \]  

(4.1)
Here $P_{2h+1}$ and $\tilde{P}_{2h+1}$ are $2^{h+1} \times 2^{h+1}$ permutation matrices, $D_{2h+1}$ and $\tilde{D}_{2h+1}$ are $2^{h+1} \times 2^{h+1}$ diagonal matrices.

We can pre-multiply a matrix $H_{2^d,d}$ by a vector by using at most $3dn$ flops.

If the matrices $D_{2h+1}$ or $\tilde{D}_{2h+1}$ are real, having nonzero entries $\pm 1$, then these flops are additions and subtractions, and matrix $H_{2^d,d}$ is orthogonal up to scaling by a constant; otherwise it is unitary.

For random permutation matrices $P_i$ and $\tilde{P}_i$ and the diagonal matrices $D_i$ and $\tilde{D}_i$, the matrix $\tilde{B} = H_{2^d,d}$ depends on up to $(1 + 1/2 + \cdots + 1/2^d)2^{d+1} = (1 - 1/2^d)2^{d+3}$ random variables.

For $d = k$, the matrix $H_{2^d,d}$ of (4.1) turns into a dense (unabridged) RSPH matrix.

By letting $D_{2h} = D_{2h} = I_{2^h}$, $P_{2h} = P_{2h} = I_{2^h}$, or $D_{2h} = D_{2h} = P_{2h} = P_{2h} = I_{2^h}$ for all $h$, we arrive at the three sub-families $ARPH$, $ARSH$, and $AH$ of the family of ARSPH matrices.

For $d = k$, an AH matrix turns into the dense (unabridged) matrix of Walsh-Hadamard transform.

Special sub-families of $2^k \times 2^k$ Abridged Scaled and Permuted Fourier (ASPF) and Abridged Scaled and Permuted Hadamard (ASPH) matrices use the same initialization of (4.1),

$$\Omega_{2^k-d,d} = H_{2^k-d,d} = \begin{pmatrix} I_{2^k-d} & I_{2^k-d} \\ I_{2^k-d} & -I_{2^k-d} \end{pmatrix},$$

and are defined by the following recursive processes, which specialize (4.1),

$$\Omega_{2^k+1,d} = \tilde{P}_{2h+1} \left( \begin{array}{cc} \Omega_{2^h,d} \\ \Omega_{2^h,d} \end{array} \right) \left( \begin{array}{cc} I_{2^h} \\ \tilde{D}_{2h} \end{array} \right) \left( \begin{array}{cc} I_{2^h} \\ I_{2^h} \end{array} \right) \left( \begin{array}{cc} I_{2^h} & -I_{2^h} \\ -I_{2^h} & I_{2^h} \end{array} \right)$$

and

$$H_{2^k+1,d} = \left( \begin{array}{cc} H_{2^k,d} \\ H_{2^k,d} \end{array} \right) \left( \begin{array}{cc} H_{2^k,d} \\ -H_{2^k,d} \end{array} \right),$$

for $h = k - d, \ldots, k - 1$ (cf. [P01, Section 2.3] and [M11, Section 3.1]), and output the matrices $PQ_{2^k,d}D$ or $PH_{2^k,d}D$ for fixed or random matrices $P$ and $D$ of primitive types 1 and 2, respectively.

Here $\tilde{D}_{2h} = \text{diag}(\omega_{2h}^i)_{i=0}^{2^h-1}$ and $\tilde{P}_{2h}$ is the $2^h \times 2^h$ odd/even permutation matrix, such that $\tilde{P}_{2h}(u) = v$, $v = (u_i)_{i=0}^{2^h-1}$, $v_j = u_{2j}$, $v_{j+2^{h-1}} = u_{2j+1}$, and $j = 0, 1, \ldots, 2^{h-1} - 1$.

The sub-families of ASPF and ASPH matrices in turn have sub-families of $ASF$, $APF$, $ASH$, and $APH$ matrices. The $n \times n$ AF matrix for $d = k$ is nothing else but the matrix of discrete Fourier transform (DFT) at $n = 2^k$ points,

$$\Omega_n = \Omega_{n,k} = (\omega_n)^{n-1}_{j=0},$$

for a primitive $n$th root of unity $\omega_n = \exp(2\pi \sqrt{-1}/n)$. (4.3)

Recursive process (4.2) defining the matrix $\Omega_n$ is known as the decimation in frequency (DIF) radix-2 representation of FFT; transposition turns it into the decimation in time (DIT) radix-2 representation of FFT. The numbers of random variables involved into generation of general ARSPF and ARSPH matrices decrease to at most $(1 - 1/2^d)2^{k+1}$ for ASPF and ASPH matrices, further decrease to at most $(1 - 1/2^d)2^{k-1}$ for ASF, APF, ASH, and APH matrices, and the AF and AH matrices involve no random variables. The estimated arithmetic cost of pre-multiplication of all these submatrices by a vector is the same as in the case of ARSPF and ARSPH matrices.

Another well-known special case is given by recursive two-sided Partial Random Butterfly Transforms (PRBTs), proposed in Technical Reports of 1995 by Parker and Parker and Pierce, which still remain unpublished (cf. [PP95]). These multipliers have been improved, carefully implemented, and then extensively tested in [BDHT13]. For an $n \times n$ input matrix and even $n = 2q$, that paper defines PRBT as follows,

$$B^{(n)} = \frac{1}{\sqrt{2}} \begin{pmatrix} R & S \\ R & -S \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} I_q & I_q \\ I_q & -I_q \end{pmatrix} \text{diag}(R, S)$$

(4.4)

where $R$ and $S$ are random diagonal nonsingular matrices. The paper [BDHT13] defines multipliers $F$ and $H$ recursively by using PRBT blocks. According to [BDHT13], the two-sided recursive processes of depth $d = 2$ with PRBT blocks are “sufficient in most cases”. In such processes $F = \text{diag}(B^{(n/2)}, B^{(n/2)})B^{(n)}$, and the multiplier $H$ is defined similarly. In the case of depth-$d$ recursion, $d \geq 2$, each of the multipliers $F$ and $H$ is defined as the product of $d$ factors made up of $2^j$ diagonal blocks of size $n/2^j \times n/2^j$, for $j = 0, \ldots, d - 1$, each block of the same type as above, and the two-sided multiplication by a vector involves $6dn$ flops for $n = 2^k$. 

12
4.5 Sparse multipliers: sparse and abridge \( f \)-circulant matrices, chains of Givens rotations, and pairs of Householder reflections

**FAMILY 6.** Sparse \( f \)-circulant matrices \( H = Z_f(v) \) are defined by a fixed or random scalar \( f \), \(|f| = 1\), and the first column \( v \) having exactly \( g \) nonzero entries, for \( g \ll n \). The positions and values of nonzeros can be randomized (and then the matrix depends on up to \( 2n + 1 \) random variables).

Such a matrix can be pre-multiplied by a vector by using at most \( (2q - 1)n \) flops or, in the real case where \( f = \pm 1 \) and \( v_i = \pm 1 \) for all \( i \), by using at most \( qn \) additions and subtractions.

**FAMILY 7.** Abridged \( f^n \)-circulant \( n \times n \) matrices, for \( n = 2^k \), positive integer \( k \), a recursion depth \( d \) such that the integer \( k - d \) is positive and not small, and a scalar \( f \) such that \(|f| = 1\), are matrices of the form

\[
H = (MD_f)^{-1} DMD_f.
\]

Here \( D = \text{diag}(d_i)_{i=1}^n \) and \( D_f = \text{diag}(f^i)_{i=0}^{n-1} \) are diagonal matrices, \(|d_i| = 1\) for all \( i \) and \( M \) is an AF or AH matrix of Family 5 of recursion depth \( d \).

Such a matrix \( H \) is unitary up to scaling by a constant (or orthogonal if the matrices \( M, D \) and \( D_f \) are real) and can be pre-multiplied by a vector by using at most \( 6dn \) flops. It involves up to \( n \) random variables, but would involve \( 3n \) (resp. \( 2n \)) such variables if we use an ASPF or ASPH (resp. ASF, ASH, ASF, or APH) rather than an AF or AH matrix \( F \). For \( d = k \) the AF matrix \( M \) turns into the DFT matrix \( \Omega_n \) and \( H \) turns into the \( g \)-circulant matrix (cf. Theorem D.1)

\[
Z_g(v) = D_f^{-1} \Omega_n^H D \Omega_n D_f, \quad \text{for } g = f^n, \ D_f = \text{diag}(f^{i})_{i=0}^{n-1}, \ D = \text{diag}(d_i)_{i=0}^{n-1}, \text{ and } (d_i)_{i=0}^{n-1} = \Omega_n D_f v.
\]

For \( f = 1 \), the above expressions are simplified: \( g = 1, D_f = I_n, M = \Omega_n, \text{ and } H = \sum_{i=0}^{n-1} u_i Z_i^1 \) is a circulant matrix:

\[
Z_1(v) = \Omega_n^H D \Omega_n, \quad D = \text{diag}(d_i)_{i=0}^{n-1}, \text{ for } (d_i)_{i=0}^{n-1} = \Omega_n v.
\]

**FAMILY 8.** Scaled and permuted chains of Givens rotations with an AF or AH factor. A permutation matrix \( P \) and \( n - 1 \) angles \( \theta_1, \ldots, \theta_{n-1} \) together define a permuted chain of Givens rotations,

\[
G(\theta_1, \ldots, \theta_{n-1}) = P \prod_{i=1}^{n-1} G(i, i + 1, \theta_i) \text{ for } n = 2^k.
\]

One can combine two such chains \( G_1 \) and \( G_2 \) with scaling and DFT into the dense unitary matrix

\[
H = D_1 G_1 D_2 G_2 D_3 \Omega_n
\]

(cf. [HMT11, Section 4.6]), for diagonal matrices \( D_1, D_2 \) and \( D_3 \) of primitive type 2.

For \( n = 2^k \gg 2^d \) we can make matrix \( H \) sparse by replacing the factor \( \Omega_n \) with an \( n \times n \) AF or AH matrix; then we can pre-multiply this matrix by a vector by using \((1.5d + 10)n\) flops.

A matrix \( H \) involves up to \( 7n \) random variables, but we can increase this bound by \( 2n \) (resp. by \( n \)) if we replace the factor \( \Omega_n \) with an ASPF or ASPH (resp. by APF, APH, ASF, or ASH) rather than AF or AH matrix.

**FAMILY 9.** Pairs of scaled and permuted Householder reflections with an AF or AH factor. Define the unitary matrix

\[
H = \frac{1}{\sqrt{n}} D_1 R_1 D_2 R_2 D_3 \Omega_n
\]

where \( D_1, D_2, \) and \( D_3 \) are diagonal matrices, \( R_1 \) and \( R_2 \) are Householder reflections, and \( \Omega_n \) is a DFT matrix. This matrix can be multiplied by a vector by using about \( 7n \) flops plus the cost of performing DFT. By randomizing diagonal, permutation and Householder reflection matrices we can involve up to \( 7n \) random variables. For \( n = 2^k \) and sparse vectors \( h \), defining Householder reflections \( R_i \), for \( i = 1, 2 \), we can make matrix \( H \) sparse by replacing the factor \( \Omega_n \) with an \( n \times n \) AF or AH matrix of recursion depth \( d \) such that \( 2^d \ll n \); then we can multiply this matrix by a vector by using at most \((1.5d + 7)n\) flops.
4.6 Estimated numbers of random variables and flops

Table 4.1 summarizes upper bounds on (a) the numbers of random variables involved into the matrices $\tilde{B}$ of Families 1–9 and (b) the numbers of flops for multiplication of such a matrix $\tilde{B}$ by a vector.\(^3\) Compare these data with $n^2$ random variables and $(2n - 1)n$ flops involved in the case of a Gaussian $n \times n$ multiplier and see more refined bounds in Sections 4.3–4.5.

| Family | 1   | 5   | 6   | 7   | 8   | 9   |
|--------|-----|-----|-----|-----|-----|-----|
| random variables | $n - 1$ | $2^k + 3$ | $2q + 1$ | $n$ | $7n - 2$ | $7n$ |
| flops   | $2n - 1$ | $3dn$  | $(2q - 1)n$ | $6dn$ | $(1.5d + 10)n$ | $(1.5d + 7)n$ |

Remark 4.2. Other observations besides flop estimates can be decisive. E. g., a special recursive structure of an ARSPH matrix $H_{2k}$ allows highly efficient parallel implementation of its multiplication by a vector based on Application Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs), incorporating Butterfly Circuits [DE].

4.7 Other basic families

There is a variety of other interesting basic matrix families. E. g., one can generalize Family 6 to the family of sparse matrices with $q$ nonzeros entries $\pm 1$ in every row and in every column for a fixed integer $q$, $1 \leq q \ll n$. Such matrices can be defined as the sums $\sum_{i=1}^q \tilde{D}_i P_1$ for fixed or random permutation matrices $P_i$ and diagonal matrices $\tilde{D}_i$, involve up to $qn$ random variables, and can be pre-multiplied by a vector at the same estimated cost as sparse $f$-circulant matrices.

For another example, one can modify a Givens chains of the form $D_1 G_1 D_2 G_2 D_3 F$, for $F$ denoting an DFT, AH, ASPF, ASPH, APF, APH, ASF, or ASH matrix, by replacing one of the matrices $G_1$ or $G_2$ with a permuted Householder reflection matrix or the inverse of a bidiagonal matrix.

The reader can find more families in our Section 7.

5 Symbolic and numerical GENP with one-sided randomized circulant preprocessing

5.1 GENP with one-sided randomized circulant preprocessing is likely to be safe universally (for any input)

In symbolic application of GENP one only cares about its safety rather than numerical safety. In this case the power of randomization is strengthened. Part (i) of Theorem 3.3 and its dual version imply that GENP pre-processed with any fixed nonsingular multiplier $F$ or $H$ (e.g., $H = I_n$) or with any fixed pair of such multipliers is unsafe only for an algebraic variety of lower dimension in the space of all inputs $A$ and that for a fixed input $A$ GENP is safe if pre-processed with almost any nonsingular multiplier or any pair of such multipliers apart from an algebraic variety of lower dimension. Therefore in symbolic computations one can quite confidently apply GENP with no preprocessing and in the very unlikely case of failure re-apply GENP with a random nonsingular multiplier.

For an interesting theoretical challenge, however, one can seek universal randomized multipliers that support safe GENP and BGE with probability 1, for any nonsingular input. This challenge has been met already in 1991 (see, e. g., [BP94, Section 2.13, entitled “Regularization of a Matrix via Preconditioning with Randomization”]). Among the known options, one-sided preprocessing with

\(^3\)The matrices of Families 2–4 involve up to $7n - 2$, $7n$, and $n$ random variables, respectively, and are multiplied by a vector by using $O(n \log(n))$ flops.
random Toeplitz multipliers of [KP91] is most efficient, but a little inferior two-sided preprocessing with random triangular Toeplitz multipliers of [KS91] has been advertised better and has become most popular.

Next we prove that even preprocessing with one-sided Gaussian or random uniform circulant multipliers (see Appendix D for definitions) is likely to make GENP safe, that is, involving no divisions by 0. Using circulant multipliers saves 50% of random variables and enables a 4-fold (resp. 2-fold) acceleration of the preprocessing of [KS91] (resp. [KP91]).

We need more than two pages + definitions in order to prove that result, but it enables us to improve substantially the popular and decades-old recipes for preprocessing GENP for symbolic computations. Namely, preprocessing of [KS91] requires pre- and post-multiplication of an $n \times n$ input matrix $A$ by an upper and a lower triangular Toeplitz matrices, respectively, at the overall cost dominated by the cost of performing twelve DFT$(n)$ per row of an input matrix $A$ (see Remark D.1), and in addition one must generate $2n - 1$ random variables. Preprocessing of [KP91] uses as many random variables and six DFT$(n)$ per row of $A$. We only need to post- or pre-multiply a matrix $A$ by a single circulant matrix, at the cost dominated by the cost of performing three DFT$(n)$ per row of an input matrix $A$, and we only generate $n$ random variables. Let us supply the details.

**Theorem 5.1.** Suppose $A = (a_{i,j})_{i,j = 1}^{n}$ is a nonsingular matrix, $T = (t_{i-j+1})_{i,j = 1}^{n}$ is a Gaussian $f$-circulant matrix, $B = AT = (b_{i,j})_{i,j = 1}^{n}$, $f$ is a fixed complex number, $t_{1}, \ldots , t_{n}$ are variables, and $t_{k} = ft_{n+k}$ for $k = 0, -1, \ldots , 1 - n$. Let $B_{l,j}$ denotes the $l$-th leading blocks of the matrix $B$ for $l = 1, \ldots , n$, and so det$(B_{l,j})$ are polynomial in $t_{1}, \ldots , t_{n}$, for all $l$, $l = 1, \ldots , n$. Then neither of these polynomials vanishes identically in $t_{1}, \ldots , t_{n}$.

**Proof.** Fix a positive integer $l \leq n$. With the convention $\alpha_{k\pm n} = f\alpha_{k}$, for $k = 1, \ldots , n$, we can write

$$B_{l,j} = \left( \sum_{k_{1} = 1}^{n} \alpha_{k_{1}}t_{k_{1}}, \sum_{k_{2} = 1}^{n} \alpha_{k_{2}+1}t_{k_{2}}, \ldots , \sum_{k_{l} = 1}^{n} \alpha_{k_{l}+l-1}t_{k_{l}} \right),$$

where $\alpha_{j}$ is the $j$th column of $A_{l,n}$. Let $a_{i,j+n} = f\alpha_{i,j}$, for $k = 1, \ldots , n$, and readily verify that

$$b_{i,j} = \sum_{k = 1}^{n} a_{i,j+k-1}t_{k},$$

and so det$(B_{l})$ is a homogeneous polynomial in $t_{1}, \ldots , t_{n}$.

Now Theorem 5.1 is implied by the following lemma.

**Lemma 5.1.** If det$(B_{l,j}) = 0$ identically in all the variables $t_{1}, \ldots , t_{n}$, then

$$\text{det}(\alpha_{i_{1}}, \alpha_{i_{2}}, \ldots , \alpha_{i_{l}}) = 0$$

for all $l$-tuples of subscripts $(i_{1}, \ldots , i_{l})$ such that $1 \leq i_{1} < i_{2} < \cdots < i_{l} \leq n$.

Indeed let $A_{l,n}$ denote the block submatrix made up of the first $l$ rows of $A$. Note that if (5.2) holds for all $l$-tuples of the subscripts $(i_{1}, \ldots , i_{l})$ above, then the rows of the block submatrix $A_{l,n}$ are linearly dependent, but they are the rows of the matrix $A$, and their linearly dependence contradicts the assumption that the matrix $A$ is nonsingular.

In the rest of this section we prove Lemma 5.1. At first we order the $l$-tuples $I = (i_{1}, \ldots , i_{l})$, each made up of $l$ positive integers written in nondecreasing order, and then we apply induction.

We order all $l$-tuples of integers by ordering at first their largest integers, in the case of ties by ordering their second largest integers, and so on.

We can define the classes of these $l$-tuples up to permutation of their integers and congruence modulo $n$, and then represent every class by the $l$-tuple of nondecreasing integers between 1 and $n$. Then our ordering of $l$-tuples of ordered integers takes the following form, $(i_{1}, \ldots , i_{l}) < (i'_{1}, \ldots , i'_{l})$ if and only if there exist a subscript $j$ such that $i_{j} < i'_{j}$ and $i_{k} = i'_{k}$ for $k = j + 1, \ldots , l$.

We begin our proof of Lemma 5.1 with the following basic result.
Lemma 5.2. It holds that
\[
\det(B_{l,t}) = \sum_{1 \leq i_1 \leq i_2 \leq \cdots \leq i_l \leq n} a_{\prod_{j=1}^l t_{i_j}} \prod_{j=1}^l t_{i_j}
\]
where a tuple \((i_1, \ldots, i_l)\) may contain repeated elements,
\[
a_{\prod_{j=1}^l t_{i_j}} = \sum_{(i'_1, \ldots, i'_l)} \det(\alpha_{i'_1}, \alpha_{i'_2+1}, \ldots, \alpha_{i'_l+1}),
\]
and \((i'_1, \ldots, i'_l)\) ranges over all permutations of \((i_1, \ldots, i_l)\).

Proof. By using (5.1) we can expand \(\det(B_{l,t})\) as follows,
\[
\begin{align*}
\det(B_{l,t}) &= \det \left( \sum_{k_1=1}^n \alpha_{k_1} t_{k_1}, \sum_{k_2=1}^n \alpha_{k_2+1} t_{k_2}, \ldots, \sum_{k_l=1}^n \alpha_{k_l+1} t_{k_l} \right) \\
&= \sum_{i_1=1}^n t_{i_1} \det \left( \sum_{k=1}^n \alpha_{k+1} t_{k}, \ldots, \sum_{k_l=1}^n \alpha_{k_l+1} t_{k_l} \right) \\
&= \sum_{i_1=1}^n t_{i_1} \sum_{i_2=1}^n t_{i_2} \det \left( \alpha_{i_1}, \alpha_{i_2+1}, \sum_{k_2=1}^n \alpha_{k_2+2} t_{k_2}, \ldots, \sum_{k_l=1}^n \alpha_{k_l+1} t_{k_l} \right) \\
&= \ldots \\
&= \sum_{i_1=1}^n t_{i_1} \sum_{i_2=1}^n t_{i_2} \cdots \sum_{i_l=1}^n t_{i_l} \det(\alpha_{i_1}, \alpha_{i_2+1}, \ldots, \alpha_{i_l+1}).
\end{align*}
\]

Consequently the coefficient \(a_{\prod_{j=1}^l t_{i_j}}\) of any term \(\prod_{j=1}^l t_{i_j}\) is the sum of all determinants
\[
\det(\alpha_{i'_1}, \alpha_{i'_2+1}, \ldots, \alpha_{i'_l+1})
\]
where \((i'_1, \ldots, i'_l)\) ranges over all permutations of \((i_1, \ldots, i_l)\), and we arrive at (5.3).

\(\square\)

In particular, the coefficient of the term \(t_1^l\) is \(a_{t_1, \ldots, t_1} = \det(\alpha_1, \alpha_2, \ldots, \alpha_l)\). This coefficient equals zero because \(B_{l,t}\) is identically zero, by assumption of lemma 5.1, and we obtain
\[
\det(\alpha_1, \alpha_2, \ldots, \alpha_l) = 0.
\]

(5.5)

This is the basis of our inductive proof of Lemma 5.1. In order to complete the induction step, it remains to prove the following lemma.

Lemma 5.3. Let \(J = (i_1, \ldots, i_l)\) be a tuple such that \(1 \leq i_1 < i_2 < \cdots < i_l \leq n\).

Then \(J\) is a subcript tuple of the coefficient of the term \(\prod_{j=1}^l t_{i_j-j+1}\) in equation (5.3).

Moreover, \(J\) is the single largest tuple among all subscript tuples.

Proof. Hereafter \(\det(\alpha_{i'_1}, \alpha_{i'_2+1}, \ldots, \alpha_{i'_l+1})\) is said to be the determinant associated with the permutation \((i'_1, \ldots, i'_l)\) of \((i_1, \ldots, i_l)\) in (5.3). Observe that \(\det(\alpha_{i_1}, \ldots, \alpha_{i_l})\) is the determinant associated with \(I = (i_1, i_2 - 1, \ldots, i_l - l + 1)\) in the coefficient \(a_{\prod_{j=1}^l t_{i_j-j+1}}\).

Let \(J'\) be a permutation of \(I\). Then \(J'\) can be written as \(J' = (s_1 - s_1 + 1, s_2 - s_2 + 2, \ldots, s_l - s_l + l)\), where \((s_1, \ldots, s_l)\) is a permutation of \((1, \ldots, l)\). The determinant associated with \(J'\) has the subscript tuple \(J' = (s_1 - s_1 + 1, s_2 - s_2 + 2, \ldots, s_l - s_l + l)\). \(J'\) satisfies the inequality \(j \leq i_j \leq n - l + j\) because by assumption \(1 \leq i_1 < i_2 < \cdots < i_l \leq n\), for any \(j = 1, 2, \ldots, l\). Thus, \(i_{s_j} - s_j + j\) satisfies the inequality \(j \leq i_{s_j} - s_j + j \leq n - l + j \leq n\), for any \(s_j\). This fact implies that no subscript of \(J'\) is negative or greater than \(n\).
Let \( J'' = (i_{s_{r_1}} - s_{r_1} + r_1, i_{s_{r_2}} - s_{r_2} + r_2, \ldots, i_{s_{r_l}} - s_{r_l} + r_l) \) be a permutation of \( J \) such that its elements are arranged in the nondecreasing order. Now suppose \( J'' \geq J \). Then we must have \( i_{s_{r_1}} - s_{r_1} + r_l \geq i_l \). This implies that

\[
i_l - i_{s_{r_1}} \leq r_l - s_{r_1}.
\]

Observe that

\[
l - s_{r_1} \leq i_l - i_{s_{r_1}} \tag{5.7}
\]

because \( i_1 < i_2 < \cdots < i_l \) by assumption. Combine bounds (5.6) and (5.7) and obtain that \( l - s_{r_1} \leq i_l - i_{s_{r_1}} \leq r_l - s_{r_1} \) and hence \( r_l = l \).

Apply this argument recursively for \( l - 1, \ldots, 1 \) and obtain that \( r_j = j \) for any \( j = 1, \ldots, l \). Therefore \( J = J' \) and \( I' = I \). It follows that \( J \) is indeed the single largest subscript tuple. \( \square \)

By combining Lemmas 5.2 and 5.3, we support the induction step of the proof of Lemma 5.1, which we summarize as follows:

**Lemma 5.4.** Assume the class of \( l \)-tuples of \( l \) positive integers written in the increasing order in each \( l \)-tuple and write \( \det(I) = \det(\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_l}) \) if \( I = (\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_l}) \).

Then \( \det(I) = 0 \) provided that \( \det(J) = 0 \) for all \( J < I \).

Finally we readily deduce Lemma 5.2 by combining this result with equation (5.5). This completes the proof of Theorem 5.1. \( \square \)

**Corollary 5.1.** Assume any nonsingular \( n \times n \) matrix \( A \) and a finite set \( S \) of cardinality \(|S|\). Sample the values of the \( n \) coordinates \( v_1, \ldots, v_n \) of a vector \( v \) at random from this set. Fix a complex \( f \) and define the matrix \( H = Z_f(v) \) of size \( n \times n \), with the first column vector \( v = (v_i)_{i=1}^n \). Then GENP and BGE are safe for the matrix \( AH \)

(i) with a probability of at least \( 1 - 0.5(n - 1)n/|S| \) if the values of the \( n \) coordinates \( v_1, \ldots, v_n \) of a vector \( v \) have been sampled uniformly at random from a finite set \( S \) of cardinality \(|S| \) or

(ii) with probability 1 if these coordinates are i.i.d. Gaussian variables.

(iii) The same claims hold for the matrix \( FA \).

**Proof.** Theorems 3.1, 5.1, and B.1 together imply parts (i) and (ii) of the corollary. By applying transposition, extend them to part (iii). \( \square \)

### 5.2 GENP with any one-sided circulant preprocessing fails numerically for some specific inputs

By virtue of Corollary 5.1 random circulant preprocessing is a universal means for ensuring safe GENP, but is it also a universal means for ensuring numerically safe GENP?

By virtue of [Pa, Corollary 6.3.1], the answer is “No”, and moreover GENP is numerically unsafe when it is applied to the DFT matrix \( \Omega_n \) already for a reasonably large integer \( n \) as well as to the matrices \( \Omega_n Z_1(v) \) and any vector \( v \) of dimension \( n \), that is, for any circulant matrix \( Z_1(v) \), and consequently for a random circulant matrix \( Z_1(v) \). By combining the proof of [Pa, Corollary 6.3.1] with Theorem D.1 one can immediately extend the result to any \( f \)-circulant multiplier \( Z_f(v) \) for any \( f \neq 0 \). It follows that GENP also fails numerically for the input pairs \((A,H)\) where \( A = \Omega_n Q \) and \( H = Q^H Z_f(v) \) for any unitary matrix \( Q \).

Surely one does not need to use GENP in order to solve a linear system of equations with a DFT coefficient matrix, but the above results reveal the difficulty in finding universal classes of structured preprocessing for GENP.

Having specific bad pairs of inputs and multipliers does not contradict part (ii) of Corollary 3.1, and actually in extensive tests in [PQZ13] and [PQY15] very good numerical stability has been observed when we applied GENP to various classes of nonsingular well-conditioned input matrices with random circulant multipliers.

---

4This nontrivial result has been deduced by using the recent specialization in [P15] to Cauchy and Vandermonde matrices of the general techniques of transformation of matrix structure proposed in [F90].
6 Alternative techniques: augmentation and additive preprocessing

Other randomization techniques, besides multiplications, are also beneficial for various fundamental matrix computations (see [M11], [HMT11], [PGMQ], [PIMR10], [PQ10], [PQZC], [PQ12], [PQZa], and the bibliography therein). By following our report [PQZa], we next supports GENP and BGE by means of applying randomized augmentation of a matrix, that is, appending to it random rows and columns, as well as by means of an alternative and closely related technique of additive preprocessing (cf. (6.2)–(6.4)).

6.1 Gaussian augmentation

By virtue of part (i) of [PQZa, Theorem 10.1], properly scaled Gaussian augmentation of a sufficiently large size is likely to produce strongly nonsingular and strongly well-conditioned matrices. Namely, this holds when we augment an $n \times n$ matrix $A$ and produce the matrix $K = \begin{pmatrix} I_h & V^T \\ U & A \end{pmatrix}$. Here $U$ and $V$ are $n \times h$ Gaussian matrices filled with $2hn$ i.i.d. Gaussian random entries and $\nu \leq h \leq n$, for $\nu$ denoting an upper bound on the numerical nullity of the leading blocks, that is, on their numerical co-rank. In the dual version of that theorem, average matrix $K$ is strongly nonsingular and strongly well-conditioned if $A$ is a Gaussian matrix and if the matrices $U$ and $V$ have full rank and are scaled so that $||U|| \approx ||V|| \approx 1$, are well-conditioned.

6.2 SRFT augmentation

By virtue of [PQZa, Section 8] we are likely to succeed if we apply GENP to the matrix $K$ above obtained by augmenting a nonsingular and well-conditioned matrix $A$ with SRFT matrices $U$ and $V$ of a sufficiently large size replacing the Gaussian ones of the previous subsection.

Let us supply some details. Part (ii) of [PQZa, Theorem 10.1] implies that we are likely to produce a strongly nonsingular and strongly well-conditioned matrix $K$ if $U$ and $V$ are SRFT matrices such that $\nu \geq q = cn$ for a sufficiently large constant $c$ and if

$$4\left(\sqrt{\nu} + \sqrt{8 \log_2(\nu n)}\right)^2 \log_2(\nu) \leq h. \quad (6.1)$$

Indeed, the $q \times q$ leading blocks $K_{q,q}$ of the matrix $K$ are the identity matrices $I_q$ for $q = 1, \ldots, h$, and so we only need to estimate the probability that the leading blocks $K_{q,q}$ are well-conditioned for all $q > h$ because their nonsingularity with probability 1 readily follows from Theorem B.1.

It is proven in [PQZa, Section 8] that under (6.1) this property holds with a probability at least $1 - c'/q$ for a constant $c'$ and a fixed $q$. Therefore it holds for all $q, q = h + 1, \ldots, h + n$ with a probability at least $1 - c' \sum_{q=h+1}^{h+n} 1/q \approx 1 - c' \ln(h+n+1) = 1 - c' \ln(1 + n/(h+1)) \geq 1 - c' \ln(1 + 1/c)$. This is close to 1 for a sufficiently large constant $c$, and our claim about the matrix $K$ follows.

6.3 Linking augmentation to additive preprocessing

Consider an augmented matrix $K$ and its inverse $K^{-1}$. Then its $n \times n$ trailing (that is, southeastern) block is $C^{-1}$ for $C = A - UV^T$. Indeed

$$K = \begin{pmatrix} I_h & O_{h,n} \\ U & I_n \end{pmatrix} \begin{pmatrix} I_h & O_{h,n} \\ O_{n,h} & C \end{pmatrix} \begin{pmatrix} I_h & V^T \\ O_{n,h} & I_n \end{pmatrix}. \quad (6.2)$$

Consequently

$$K^{-1} = \begin{pmatrix} I_h & -V^T \\ O_{n,h} & I_n \end{pmatrix} \begin{pmatrix} I_h & O_{h,n} \\ O_{n,h} & C^{-1} \end{pmatrix} \begin{pmatrix} I_h & O_{h,n} \\ -U & I_n \end{pmatrix}, \quad (6.3)$$

$$C^{-1} = \text{diag}(O_{h,h}I_n)K^{-1}\text{diag}(O_{h,h}I_n), \text{ and the claim follows.}$$
Now deduce that \( ||K^{-1}||/N \leq ||C^{-1}|| \leq ||K^{-1}||, \ ||K||/N \leq ||C^{-1}|| \leq N ||K|| \), and hence
\[
\kappa(K)/N^2 \leq \kappa(C) \leq N\kappa(K), \text{ for } N = (1 + ||U||)(1 + ||V||).
\]
(6.4)
The above equations closely link augmentation \( A \rightarrow K \) with additive preprocessing \( A \rightarrow C \) and also link the leading blocks of the augmented matrices with those output by additive preprocessing.

Indeed extend readily our observations to obtain that the \( k \times k \) trailing submatrix of the leading block \( K_{h+k,h+k}^{-1} \) of the matrix \( K \) is the \( k \times k \) leading block \( C_{k,k}^{-1} \) of the matrix \( C^{-1} \) and that
\[
\kappa(K_{h+k,h+k})/N^2 \leq \kappa(C_{k,k}) \leq N\kappa(K_{h+k,h+k}),
\]
(6.5)
for \( k = 1, \ldots, n \). If the factor \( N \) is reasonably bounded, which is likely for Gaussian matrices \( U \) and \( V \), then the matrix \( C_{k,k} \) is nonsingular and well-conditioned if and only if so is the matrix \( K_{h+k,h+k} \).

6.4 Transition back to computations with the original matrix. Expansion and homotopy continuation

Having computed the inverses \( K^{-1} \) and \( C^{-1} \) by applying GENP or BGE to the augmented matrix \( K \), one can simplify computation of the inverse \( A^{-1} \) of the original matrix \( A \) by applying the Sherman–Morrison–Woodbury formula\(^5\)
\[
A^{-1} = C^{-1} - C^{-1}U(I_h + V^T C^{-1}U)^{-1}V^T C^{-1},
\]
(6.6)
for \( C = A - UV^T \) (cf. [GL13, page 65]).

Computing the inverse \( A^{-1} \) by means of SMW formula (6.6) may still cause numerical problems at the stages of computing and inverting the matrix \( I_h + V^T C^{-1}U \), but they are less likely to occur if the matrix \( C \) is nonsingular and well-conditioned, which we expect to be the case in this application.

In order to strengthen the chances for the success of this approach we can apply some heuristic recipes. In our tests with benchmark inputs, we succeeded by simply doubling the lower bound \( \nu \) on the dimension \( h \) of additive preprocessing or equivalently by keeping the same bound \( \nu \) but requiring that \( 2\nu \leq h \leq n \).

Another natural remedy is the well-known general technique of homotopy continuation, with which we proceed as follows. Fix two matrices \( U \) and \( V \) as before and define the matrices \( C(\tau) = A - \tau UV^T \) and \( S(\tau) = I_h + \tau V^T C(\tau)^{-1}U \) for a nonnegative parameter \( \tau \). Suppose that the matrix \( S(\tau) \) is diagonally dominant for some positive \( \tau \). Note that the matrices \( C(1) = A - UV^T \) and \( S(0) = I_h \) are readily invertible, fix the decreasing sequence of the values \( \tau_k, k = 0, 1, \ldots, l \), such that \( \tau_0 = 1 > \tau_1 > \cdots > \tau_l = 0 \), and compute the sequence of the matrices \( \tau_j V^T C(\tau_j)^{-1}U \), for \( j = 0, 1, \ldots, l \), by extending SMW formula (6.6) as follows,
\[
C(\tau_{j+1})^{-1} = C(\tau_j)^{-1} - \Delta_j C(\tau_j)^{-1}U(I_h + \Delta_j V^T C(\tau_j)^{-1}U)^{-1}V^T C(\tau_j)^{-1},
\]
for \( \Delta_j = \tau_{j+1} - \tau_j \). For sufficiently small values \( \Delta_j \), the matrices \( I_h + \Delta_j V^T C(\tau_j)^{-1}U \) are diagonally dominant and readily invertible, and then we can numerically safely perform \( l \) homotopy continuation steps for the transition from the inverse \( C(1)^{-1} = C^{-1} \) to \( C(0)^{-1} = A^{-1} \).

The SMW formula (6.6) can be readily generalized to an expression for the inverse
\[
K^{-1} = \begin{pmatrix} I_h & -V^T C^{-1} \\ O_{n,h} & C^{-1} \end{pmatrix} \begin{pmatrix} I_h & 0_{h,n} \\ -U & I_n \end{pmatrix}
\]
of the augmented matrix
\[
K = \begin{pmatrix} I_h & V^T \\ U & A \end{pmatrix} = \begin{pmatrix} I_h & 0_{h,n} \\ U & I_n \end{pmatrix} \begin{pmatrix} I_h & V^T \\ O_{n,h} & C \end{pmatrix} \text{ for } C = A - UV^T.
\]
\(^5\)Hereafter we use the acronym SMW.
7 Numerical Experiments

Numerical experiments have been designed by the first author and have been performed by Xiaodong Yan (see Tables 7.1–7.4) and by the second author (see the other tables). The tests have been run by using MATLAB in the Graduate Center of the City University of New York on a Dell computer with the Intel Core 2 2.50 GHz processor and 4G memory running Windows 7. Gaussian matrices have been generated by applying the standard normal distribution function randn of MATLAB. We refer the reader to [PQZ13], [PQY14], [PQY15], and [PQZa] for other extensive tests of GENP with randomized preprocessing.

Tables 7.1–7.4 show the maximum, minimum and average relative residual norms $||Ay - b||/||b||$ as well as the standard deviation for the solution of 1000 linear systems $Ax = b$ with Gaussian vector $b$ and $n \times n$ input matrix $A$ for each $n$, $n = 256, 512, 1024$. The linear systems have been solved by using GEPP, GENP, or GENP pre-processed with real Gaussian, real Gaussian circulant, and random circulant multipliers, each followed by a single loop of iterative refinement.

The tests have been applied to the matrices

$$A = \begin{pmatrix} A_k & B \\ C & D \end{pmatrix},$$

(7.1)

with $k \times k$ blocks $A_k, B, C$ and $D$, for $k = n/2$, scaled so that $||B|| \approx ||C|| \approx ||D|| \approx 1$, the $k - 4$ singular values of the matrix $A_k$ were equal 1 and the other ones were set to 0 (cf. [HO2, Section 28.3]), and with Gaussian Toeplitz matrices $B, C$, and $D$, that is, with Toeplitz matrices of (D.1), each defined by the i.i.d. Gaussian entries of its first row and first column. (The norm $||A^{-1}||$ ranged from $2.2 \times 10^3$ to $3.8 \times 10^5$ in these tests.) In the tests covered in Table 7.4, the matrix $A$ was set to equal $\Omega$, the matrix of DFT($n$). For preprocessing, either Gaussian or Gaussian unitary circulant matrices $C = \Omega^{-1}D(\Omega v)\Omega$ have been used as multipliers, with $v = (v_i)_{i=0}^{n-1}, v_i = \exp(2\pi i \phi_i \sqrt{-1}/n)$ and $n$ i.i.d. real Gaussian variables $\phi_i$, $i = 0, \ldots, n - 1$ (cf. Theorem D.1 and Remark D.4).

As should be expected, GEPP has always produced accurate solutions, with average relative residual norms ranging from $10^{-12}$ to $7 \times 10^{-13}$, but GENP with no preprocessing has consistently produced corrupted output with relative residual norms ranging from $10^{-3}$ to $10^{2}$ for the input matrices $A$ of equation (7.1). Even much worse was the output accuracy when GENP with no preprocessing or with Gaussian circulant preprocessing was applied to the matrix $A = \Omega$. In all other cases, however, GENP with random circulant preprocessing and with a single loop of iterative refinement has produced solution with desired accuracy, matching the output accuracy of GEPP. Furthermore GENP has performed similarly when it was applied to a nonsingular and well-conditioned input pre-processed with a Gaussian multiplier.

| dimension | iterations | mean     | max       | min       | std       |
|-----------|------------|----------|-----------|-----------|-----------|
| 256       | 0          | $6.13 \times 10^{-9}$ | $3.39 \times 10^{-6}$ | $2.47 \times 10^{-12}$ | $1.15 \times 10^{-7}$ |
| 256       | 1          | $3.64 \times 10^{-12}$ | $4.32 \times 10^{-12}$ | $1.91 \times 10^{-15}$ | $2.17 \times 10^{-13}$ |
| 512       | 0          | $5.57 \times 10^{-6}$ | $1.44 \times 10^{-5}$ | $1.29 \times 10^{-14}$ | $7.59 \times 10^{-7}$ |
| 512       | 1          | $7.36 \times 10^{-11}$ | $1.92 \times 10^{-10}$ | $3.32 \times 10^{-15}$ | $1.07 \times 10^{-14}$ |
| 1024      | 0          | $2.58 \times 10^{-14}$ | $2.17 \times 10^{-14}$ | $4.66 \times 10^{-15}$ | $6.86 \times 10^{-15}$ |
| 1024      | 1          | $7.53 \times 10^{-12}$ | $7.31 \times 10^{-9}$ | $6.75 \times 10^{-15}$ | $2.31 \times 10^{-10}$ |

Next we cover our tests of GENP pre-processed with some multipliers defined by means of combining matrices of Section 4. The test results are represented in Tables 7.5 and 7.6.

In this series of our tests we applied GENP to the above matrices of (7.1) and six families of benchmark matrices from [BDHT13], pre-processed with multipliers combining the ones of following three basic families.

**Family 1:** The matrices APF of depth 3 (with $d = 3$) and with a (single) random permutation.
Table 7.2: Relative residual norms: GENP with Gaussian circulant multipliers

| dimension | iterations | mean   | max    | min    | std     |
|-----------|------------|--------|--------|--------|---------|
| 256       | 0          | 8.97 × 10⁻¹³ | 1.19 × 10⁻⁸ | 6.23 × 10⁻¹⁵ | 4.85 × 10⁻¹⁰ |
| 256       | 1          | 2.88 × 10⁻¹⁴ | 2.89 × 10⁻¹² | 1.89 × 10⁻¹⁵ | 1.32 × 10⁻¹³ |
| 512       | 0          | 4.12 × 10⁻¹⁰ | 3.85 × 10⁻⁸ | 2.37 × 10⁻¹² | 2.27 × 10⁻⁹  |
| 512       | 1          | 5.24 × 10⁻¹⁴ | 5.12 × 10⁻¹⁲ | 2.95 × 10⁻¹⁵ | 2.32 × 10⁻¹³ |
| 1024      | 0          | 1.03 × 10⁻⁸  | 5.80 × 10⁻⁶  | 1.09 × 10⁻¹¹ | 1.93 × 10⁻⁷  |
| 1024      | 1          | 1.46 × 10⁻¹⁴ | 4.80 × 10⁻¹¹ | 6.94 × 10⁻¹⁰ | 1.60 × 10⁻¹² |

Table 7.3: Relative residual norms: GENP with circulant multipliers filled with ±1

| dimension | iterations | mean   | max    | min    | std     |
|-----------|------------|--------|--------|--------|---------|
| 256       | 0          | 2.37 × 10⁻¹² | 2.47 × 10⁻¹⁰ | 9.41 × 10⁻¹³ | 1.06 × 10⁻¹⁴ |
| 256       | 1          | 2.88 × 10⁻¹⁴ | 3.18 × 10⁻¹² | 1.83 × 10⁻¹⁵ | 1.36 × 10⁻¹³ |
| 512       | 0          | 7.43 × 10⁻¹² | 6.77 × 10⁻¹⁰ | 3.35 × 10⁻¹³ | 3.04 × 10⁻¹¹ |
| 512       | 1          | 5.22 × 10⁻¹⁴ | 4.97 × 10⁻¹² | 3.19 × 10⁻¹⁵ | 2.29 × 10⁻¹³ |
| 1024      | 0          | 4.43 × 10⁻¹¹ | 1.31 × 10⁻⁸  | 1.28 × 10⁻¹² | 4.36 × 10⁻¹⁰ |
| 1024      | 1          | 1.37 × 10⁻¹¹ | 4.33 × 10⁻¹¹ | 6.67 × 10⁻¹⁰ | 1.41 × 10⁻¹² |

Family 2: Sparse circulant matrices \( C = \Omega^{-1}D(\Omega\mathbf{v})\Omega \), where the vector \( \mathbf{v} \) has been filled with zeros, except for its ten coordinates filled with ±1. Here and hereafter each sign + or − has been assigned with probability 1/2.

Family 3: Sum of two inverse bidiagonal matrices. At first their main diagonals have been filled with the integer 101, and their first subdiagonals have been filled with ±1. Then each matrix have been multiplied by a diagonal matrix \( \text{diag}(\pm 2^{b_i}) \), where \( b_i \) were random integers uniformly chosen from 0 to 3.

We tested GENP on ten combinations of these three basic families of multipliers, listed below. The size of the linear system was 128. For each combination we have performed 1000 tests and have recorded the average relative error \( ||Ax - b||/||b|| \) with matrices \( A \) from the seven benchmark families and vectors \( b \) being standard Gaussian vectors. Here are these ten combinations.

1. \( F = I, H \) is a matrix of Family 1.
2. \( F = I, H \) is a matrix of Family 3.
3. \( F = H \) is a matrix of Family 1.
4. \( F = H \) is a matrix of Family 3.
5. \( F \) is a matrix of Family 1, \( H \) is a matrix of Family 3.
6. \( F = I, H \) is the product of two matrices of Family 1.
7. \( F = I, H \) is the product of two matrices of Family 2.
8. \( F = I, H \) is the product of two matrices of Family 3.
9. \( F = I, H \) is the sum of two matrices of Families 1 and 3.
10. \( F = I, H \) is the sum of two matrices of Families 2 and 3.

We tested these multipliers for the same linear systems as in our previous tests in this section and for six classes generated from Matlab, by following their complete description in Matlab and [BDHT13]. Here is the list of these seven test classes.

1. The matrices \( A \) of (7.1).
2. 'circul': circulant matrices whose first row is a standard Gaussian random vector.
3. 'condex': counter-examples to matrix condition number estimators.
4. 'fiedler': symmetric matrices generated with \((i, j)\) and \((j, i)\) elements equal to \(c_i - c_j\) where \(c_1, \ldots, c_n\) are i.i.d. standard Gaussian variables.
5. 'orthog': orthogonal matrices with \((i, j)\) elements \(\sqrt{\frac{2}{n+1}} \sin \frac{ij\pi}{n+1}\).
Table 7.4: Relative residual norms: GENP for DFT($n$) with Gaussian multipliers

| dimension | iterations | mean     | max       | min    | std      |
|-----------|------------|----------|-----------|--------|----------|
| 256       | 0          | 2.26e-12 | 4.23e-11  | 2.83e-13| 4.92e-12 |
| 256       | 1          | 1.05e-15 | 1.26e-15  | 9.14e-16| 6.76e-17 |
| 512       | 0          | 1.11e-14 | 6.23e-10  | 6.72e-13| 6.22e-14 |
| 512       | 1          | 1.50e-15 | 1.69e-15  | 1.33e-15| 6.82e-17 |
| 1024      | 0          | 7.57e-10 | 7.25e-8   | 1.89e-12| 7.25e-10 |
| 1024      | 1          | 2.13e-12 | 2.29e-12  | 1.96e-13| 7.15e-12 |

6. 'randcorr': random $n \times n$ correlation matrices with random eigenvalues from a uniform distribution. (A correlation matrix is a symmetric positive semidefinite matrix with 1's on the diagonal.)

7. 'toeppd': $n \times n$ symmetric, positive semi-definite (SPSD) Toeplitz matrices $T$ equal to the sums of $m$ rank-2 SPSD Toeplitz matrices. Specifically,

$$T = w(1) \ast T(\theta(1)) + ... + w(m) \ast T(\theta(m))$$

where $\theta(k)$ are i.i.d. Gaussian variables and $T(\theta(k)) = (\cos(2\pi(i-j)\theta(k)))_{i,j=1}^n$.

In our tests, for some pairs of inputs and multipliers, GENP has produced no meaningful output. In such cases we filled the respective entries of Tables 7.5 and 7.6 with $\infty$.

GENP pre-processed with our multipliers of the 9th combination of three basic families, has produced accurate outputs without iterative refinement for all seven benchmark classes of input matrices. With the other combinations of the three basic families of our multipliers, this was achieved from 4 to 6 (out of 7) benchmark input classes. For comparison, the 2-sided preprocessing with PRBT-based multipliers of [BDHT13] and [BBBDD14] always required iterative refinement.

Table 7.5: Relative residual norms output by pre-processed GENP with no refinement iterations

| class | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|-------|-----|-----|-----|-----|-----|-----|-----|
| 1     | 2.61e-13 | 6.09e-15 | $\infty$ | 2.62e+02 | 7.35e-15 | 1.38e-12 | 3.04e-13 |
| 2     | 2.02e+02 | 4.34e-14 | 5.34e-16 | $\infty$ | 7.35e+02 | 5.27e-15 | 3.23e-15 |
| 3     | 4.34e-13 | 8.36e-15 | $\infty$ | 3.03e+02 | 1.94e-14 | 3.04e-13 | 3.21e-13 |
| 4     | 1.48e+01 | 1.36e-12 | 2.39e-16 | 1.01e-11 | 4.71e+01 | 5.09e-15 | 5.12e-15 |
| 5     | 3.71e-11 | 2.21e-14 | $\infty$ | 2.85e+01 | 5.83e-10 | 2.23e-12 | 1.34e-12 |
| 6     | 3.33e-13 | 9.36e-15 | $\infty$ | 3.66e-05 | 7.04e-15 | 3.75e-13 | 2.11e-13 |
| 7     | 7.76e-12 | 3.55e-14 | 9.91e+01 | 7.90e+00 | 7.75e+00 | 7.11e+00 | 1.05e+01 |
| 8     | 7.95e+00 | 9.55e-14 | 7.56e-16 | $\infty$ | 5.74e+03 | 6.51e-15 | 3.57e-15 |
| 9     | 5.36e-13 | 1.51e-14 | 4.26e-16 | 2.24e-11 | 3.68e-13 | 6.47e-15 | 4.92e-15 |
| 10    | 3.50e-12 | 8.43e-14 | 3.43e-13 | 2.90e-10 | 1.36e+01 | 3.53e-13 | 1.67e-13 |

We also tested GENP with additive preprocessing applied to the same $n \times n$ test matrices $A$ of (7.1), but for $n = 32, 64, 128, 256$. In this case we applied GENP to the matrix $C = A - UV^T$ where $U$ and $V$ were $n \times h$ random Gaussian subcircular matrices each defined by the $n$ i.i.d. Gaussian entries of its first column and scaled so that $||A|| = 2||UV^T||$. Then we computed the solution $x$ to the linear system $Ax = b$ for a Gaussian vector $b$ by substituting the SMW formula (6.6) into the equation $x = A^{-1}b$.

We present the test results in Table 7.7. The results changed little when we scaled the matrices $U$ and $V$ to increase the ratio $||A||/||UV^T||$ to 10 and 100.

Finally we present the results of our tests of GENP with additive preprocessing applied to the same $n \times n$ test matrices $A$ of (7.1), but for $n = 32, 64, 128, 256$. In this case we applied GENP to
Table 7.6: Relative residual norms output by pre-processed GENP followed by a single refinement iteration

| class | 1      | 2      | 3     | 4     | 5     | 6     | 7     |
|-------|--------|--------|-------|-------|-------|-------|-------|
| 1     | 1.13e-15 | 6.90e-17 | ∞     | 1.12e+00 | 5.23e-17 | 2.10e-16 | 1.05e-15 |
| 2     | 5.07e-04 | 7.71e-17 | 1.05e-16 | ∞     | 4.40e+02 | 1.99e-16 | 1.19e-15 |
| 3     | 1.14e-15 | 7.34e-17 | ∞     | 5.43e-13 | 5.15e-17 | 2.24e-16 | 1.10e-15 |
| 4     | 1.55e-03 | 6.19e-17 | 1.31e-16 | 5.69e-13 | 2.69e+02 | 2.13e-16 | 1.17e-15 |
| 5     | 9.80e-16 | 6.96e-17 | ∞     | 6.75e+01 | 5.55e-17 | 2.47e-16 | 9.84e-16 |
| 6     | 1.08e-15 | 6.13e-17 | ∞     | 6.35e-13 | 5.08e-17 | 1.86e-16 | 1.05e-15 |
| 7     | 3.47e+01 | 6.17e-17 | 2.61e+00 | 5.21e+00 | 5.31e-17 | 1.97e-16 | 9.97e-16 |
| 8     | 2.56e-04 | 6.67e-17 | 1.15e-16 | ∞     | 7.96e+02 | 1.98e-16 | 1.08e-15 |
| 9     | 9.81e-16 | 7.44e-17 | 3.99e-17 | 6.40e-13 | 5.09e-17 | 2.02e-16 | 1.15e-15 |
| 10    | 9.79e-16 | 8.32e-17 | 1.14e-16 | 7.34e-13 | 4.07e+01 | 2.23e-16 | 1.04e-15 |



Table 7.7: Relative residual norms of GENP with Gaussian subcirculant additive preprocessing

| n   | h | Refinement | mean    | std      |
|-----|---|------------|---------|----------|
| 16  | 4 | 0          | 1.67e-11 | 7.87e-11 |
| 16  | 4 | 1          | 6.15e-15 | 2.92e-14 |
| 32  | 4 | 0          | 8.42e-11 | 4.86e-10 |
| 32  | 4 | 1          | 1.49e-14 | 9.09e-14 |
| 64  | 4 | 0          | 9.23e-11 | 3.99e-10 |
| 64  | 4 | 1          | 1.63e-14 | 4.24e-14 |
| 128 | 4 | 0          | 6.55e-10 | 2.42e-09 |
| 128 | 4 | 1          | 7.50e-13 | 2.99e-13 |
| 256 | 4 | 0          | 1.13e-08 | 3.38e-08 |
| 256 | 4 | 1          | 1.10e-12 | 6.23e-13 |


Acknowledgements: This work has been supported by NSF Grant CCF 1116736 and PSC CUNY Award 67699-00 45.

References

[B85] J. R. Bunch, Stability of Methods for Solving Toeplitz Systems of Equations, *SIAM Journal on Scientific and Statistical Computing*, 6, 2, 349–364, 1985.

[BA80] R. R. Bitmead, B. D. O. Anderson, Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations, *Linear Algebra and Its Applications*, 34, 103–116, 1980.

[BBBDDD14] M. Baboulin, D. Becker, G. Bosilca, A. Danalis, J. Dongarra, An Efficient Distributed Randomized Algorithm for Solving Large Dense Symmetric Indefinite Linear Systems,
Parallel Computing (7th Workshop on Parallel Matrix Algorithms and Applications), 40, 7, 213–223, 2014.

[BBD12] D. Becker, M. Baboulin, J. Dongarra, Reducing the Amount of Pivoting in Symmetric Indefinite Systems, Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics, PPAM 2011, Lecture Notes in Computer Science, 7205, 133–142, Springer, 2012. Also INRIA, Research Report 7621 (05/2011) and University of Tennessee, Technical Report ICL-UT-11-06.

[BCD14] G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, O. Schwartz, Communication Lower Bounds and Optimal Algorithms for Numerical Linear Algebra, Acta Numerica (an invited paper), 23, 1–155, 2014.

[BDHT13] M. Baboulin, J. Dongarra, J. Herrmann, S. Tomov, Accelerating Linear System Solutions Using Randomization Techniques, ACM Transactions on Mathematical Software (TOMS), 39, 2, 2013.

[BLR15] M. Baboulin, X. S. Li, F-H. Rouet, Using Random Butterfly Transformations to Avoid Pivoting in Sparse Direct Methods, Proceedings of VECPAR 2014, Lecture Notes in Computer Science, 8969, 135–144, Springer, 2015.

[BP94] D. Bini, V. Y. Pan, Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms, Birkhäuser, Boston, 1994.

[BV88] W. Bruns, U. Vetter, Determinantal Rings, Lecture Notes in Math., 1327, Springer, Heidelberg, 1988.

[CD05] Z. Chen, J. J. Dongarra, Condition Numbers of Gaussian Random Matrices, SIAM. J. on Matrix Analysis and Applications, 27, 603–620, 2005.

[CPW74] R. E. Cline, R. J. Plemmons, G. Worm, Generalized Inverses of Certain Toeplitz Matrices, Linear Algebra and Its Applications, 8, 25–33, 1974.

[D88] J. Demmel, The Probability That a Numerical Analysis Problem Is Difficult, Math. of Computation, 50, 449–480, 1988.

[DDF14] S. Donfack, J. Dongarra, M. Faverge, M. Gates, J. Kurzak, P. Luszczek, I. Yamazaki, A Survey of Recent Developments in Parallel Implementations of Gaussian Elimination, CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE, Concurrency Comput.: Pract. Exper. (2014) Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3306

[DE] Dillon Engineering, http://www.dilloneng.com/fft_ip/parallel-fft.

[DKM06] P. Drineas, R. Kannan, M.W. Mahoney, Fast Monte Carlo Algorithms for Matrices I–III, SIAM J. on Computing, 36, 1, 132–206, 2006.

[DL78] R. A. Demillo, R. J. Lipton, A Probabilistic Remark on Algebraic Program Testing, Information Processing Letters, 7, 4, 193–195, 1978.

[DS01] K. R. Davidson, S. J. Szarek, Local Operator Theory, Random Matrices, and Banach Spaces, in Handbook on the Geometry of Banach Spaces (W. B. Johnson and J. Lindenstrauss editors), pages 317–368, North Holland, Amsterdam, 2001.

[E88] A. Edelman, Eigenvalues and Condition Numbers of Random Matrices, SIAM J. on Matrix Analysis and Applications, 9, 4, 543–560, 1988.

[EGH13] Y. Eidelman, I. Gohberg, I. Haimovici, Separable Type Representations of Matrices and Fast Algorithms. Volume 1. Basics. Completion Problems. Multiplication and Inversion Algorithms. Volume 2. Eigenvalue method, Birkhauser, 2013.
[ES05] A. Edelman, B. D. Sutton, Tails of Condition Number Distributions, *SIAM J. on Matrix Analysis and Applications*, **27**, 2, 547–560, 2005.

[FKV98] A. Frieze, R. Kannan, S. Vempala, Fast Monte-Carlo Algorithms for Finding Low-rank Approximations, *J. of ACM*, **51**, 1025–1041, 2004. Proc. version in *39th FOCS*, pages 370–378, IEEE Computer Society Press, 1998.

[G11] J.F. Grcar, Mathematicians of Gaussian Elimination, *Notices of the Amer. Math. Society*, **58**, 6, 782–792, June/July 2011.

[GL13] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 2013 (fourth addition).

[GS72] I. Golberg, A. Semenstul, On the Inversion of Finite Toeplitz Matrices and Their Continuous Analogs, *Matematicheskiie Issledovaniia* (in Russian), **7**, 2, 187–224, 1972.

[GTZ97] S. A. Goreinov, E. E. Tyrtyshnikov, N. L. Zamarashkin, A Theory of Pseudo-skeleton Approximations, *Linear Algebra and Its Applications*, **261**, 1–21, 1997.

[GZT97] S. A. Goreinov, N. L. Zamarashkin, E. E. Tyrtyshnikov, Pseudo-skeleton Approximations by Matrices of Maximal Volume, *Mathematical Notes*, **62**, 4, 515–519, 1997.

[H02] N. J. Higham, *Accuracy and Stability in Numerical Analysis*, SIAM, Philadelphia, 2002 (second edition).

[HMT11] N. Halko, P. G. Martinsson, J. A. Tropp, Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions, *SIAM Review*, **53**, 2, 217–288, 2011.

[KP91] E. Kaltofen, V. Y. Pan, Processor Efficient Parallel Solution of Linear Systems over an Abstract Field, *Proc. 3rd Ann. ACM Symp. on Parallel Algorithms and Architectures (SPAA’91)*, 180–191, ACM Press, New York, 1991.

[KS91] E. Kaltofen, B. D. Saunders, On Wiedemann’s Method for Solving Sparse Linear Systems, *Proceedings of AAECC–5, Lecture Notes in Computer Science*, **536**, 29–38, Springer, Berlin, 1991.

[M74] M. Morf, Fast Algorithms for Multivariable Systems, Ph.D. Thesis, *Department of Electrical Engineering, Stanford University*, Stanford, CA, 1974.

[M80] M. Morf, Doubling Algorithms for Toeplitz and Related Equations, *Proc. IEEE International Conference on ASSP*, 954–959, IEEE Press, Piscataway, New Jersey, 1980.

[M11] M. W. Mahoney, Randomized Algorithms for Matrices and Data, *Foundations and Trends in Machine Learning*, NOW Publishers, **3**, 2, 2011. (Abridged version in: *Advances in Machine Learning and Data Mining for Astronomy*, edited by M. J. Way, et al., pp. 647-672, 2012.)

[P90] V. Y. Pan, On Computations with Dense Structured Matrices, *Math. of Computation*, **55**, **191**, 179–190, 1990. Proceedings version in *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC’89)*, 34–42, ACM Press, New York, 1989.

[P01] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

[P15] V. Y. Pan, Transformations of Matrix Structures Work Again, *Linear Algebra and Its Applications*, 465, 1–32, 2015.

[Pa] V. Y. Pan, How Bad Are Vandermonde Matrices? Available at arxiv: 1504.02118, submitted on 8 April 2015, revised on 10 July 2015.
[PGMQ] V. Y. Pan, D. Grady, B. Murphy, G. Qian, R. E. Rosholt, A. Ruslanov, Schur Aggregation for Linear Systems and Determinants, *Theoretical Computer Science, Special Issue on Symbolic–Numerical Algorithms* (D. A. Bini, V. Y. Pan, and J. Verschelde editors), **409**, 2, 255–268, 2008.

[PIMR10] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, *Linear Algebra and Its Applications*, **432**, 1070–1089, 2010.

[PP95] D. S. Parker, B. Pierce, The Randomizing FFT: An Alternative to Pivoting in Gaussian Elimination, Tech. Report CSD 950037, *Computer Science Dept., Univ. California at Los Angeles*, 1995.

[PQ10] V. Y. Pan, G. Qian, Solving Homogeneous Linear Systems with Randomized Preprocessing, *Linear Algebra Appl.*, **432**, 3272–3318, 2010.

[PQ12] V. Y. Pan, G. Qian, Solving Linear Systems of Equations with Randomization, Augmentation and Aggregation, *Linear Algebra and Its Applications*, **437**, 2851–1876, 2012.

[PQY14] V. Y. Pan, G. Qian, X. Yan, Random Multipliers Numerically Stabilize Gaussian and Block Gaussian Elimination: Proofs and an Extension to Low-rank Approximation, arxiv 1406.5802 math.NA (28 pages), June 23, 2014.

[PQY15] V. Y. Pan, G. Qian, X. Yan, Random Multipliers Numerically Stabilize Gaussian and Block Gaussian Elimination: Proofs and an Extension to Low-rank Approximation, *Linear Algebra and Its Applications*, **481**, 202–234, 2015.

[PQZ13] V. Y. Pan, G. Qian, A. Zheng, Randomized Preprocessing versus Pivoting, *Linear Algebra and Its Applications*, **438**, 4, 1883–1899, 2013.

[PQZC] V. Y. Pan, G. Qian, A. Zheng, Z. Chen, Matrix Computations and Polynomial Rootfinding with Preprocessing, *Linear Algebra and Its Applications*, **434**, 854–879, 2011.

[PQZa] V. Y. Pan, G. Qian, L. Zhao, New Studies of Randomized Augmentation and Additive Preprocessing, arxiv 1412.5864 math.NA (38 pages), submitted on December 18, 2014, revised on July 10, 2015.

[PSZ15] V. Y. Pan, J. Svadlenka, L. Zhao, Estimating the Norms of Circulant and Toeplitz Random Matrices and Their Inverses, *Linear Algebra and Its Applications*, **468**, 197–210, 2015.

[PW08] V. Y. Pan, X. Wang, Degeneration of Integer Matrices Modulo an Integer, *Linear Algebra and Its Applications*, **429**, 2113–2130, 2008.

[PZ15] V. Y. Pan, L. Zhao, Randomized Circulant and Gaussian Preprocessing, in *Proceedings of the 17th International Workshop on Computer Algebra in Scientific Computing (CASC’2015)*, (V. P. Gerdt, V. Koepf, and E. V. Vorozhtsov, editors), *Lecture Notes in Computer Science*, **9301**, 359–373, Springer International Publishing Switzerland, 2015.

[S80] J. T. Schwartz, Fast Probabilistic Algorithms for Verification of Polynomial Identities, *Journal of ACM*, **27**, 4, 701–717, 1980.

[SST06] A. Sankar, D. Spielman, S.-H. Teng, Smoothed Analysis of the Condition Numbers and Growth Factors of Matrices, *SIAM J. on Matrix Analysis and Applications.*, **28**, 2, 446–476, 2006.

[T00] E. E. Tyrtyshnikov, Incomplete Cross Approximation in the Mosaic-skeleton Method, *Computing*, **64**, 4, 367–380, 2000.
Appendix

A  General matrices

1. $W = S_W \Sigma_W T_W^T$ is its full Singular Value Decomposition (or full SVD), where $S_W$ and $T_W$ are unitary (or real orthogonal) square matrices and $\Sigma_W = \text{diag}(\text{diag}(\sigma_j(W)))_{j=1}^\rho$, for $\rho = \text{rank}(W)$, is the $m \times n$ diagonal matrix of the singular values of the matrix $W$,

$$\sigma_1(W) \geq \sigma_2(W) \geq \cdots \geq \sigma_\rho(W) > 0, \quad \sigma_j(W) = 0 \text{ for } j > \rho.$$

2. $W = S_{W,\rho} \Sigma_{W,\rho} T_{W,\rho}^T$ is its compact SVD, where $\Sigma_{W,\rho} = \text{diag}(\sigma_j(W))_{j=1}^\rho$ is the $\rho \times \rho$ leading submatrix of $\Sigma_W$ and the matrices $S_{W,\rho}$ and $T_{W,\rho}$ are formed by the first $\rho$ columns of the matrices $S_W$ and $T_W$, respectively.

3. $W^+ = T_{W,\rho} \Sigma_{W,\rho}^{-1} S_{W,\rho}^T$ is its Moore–Penrose pseudo inverse.

$$(W^+)^+ = W, \quad WW^+ = I_n \text{ if } \text{rank}(W) = n, \quad W^+W = I_n \text{ if } \text{rank}(W) = n, \quad W^+ = W^{-1} \text{ for a nonsingular matrix } W.$$

4. $||W|| = \sigma_1(W)$ and $||W||_F = (\sum_{j=1}^\rho \sigma_j^2(W))^{1/2} = (\text{Trace } (W^HW))^{1/2}$ denote its spectral and Frobenius norms, respectively.

$$||VW|| \leq ||V|| \cdot ||W|| \text{ and } ||VW||_F \leq ||V||_F \cdot ||W||_F, \text{ for any matrix product } VW.$$

$||U|| = ||U^+|| = 1, \quad ||UW|| = ||W|| \text{ and } ||WU|| = ||W|| \text{ if the matrix } U \text{ is unitary (or real orthogonal)}.$

5. $\sigma_\rho(W) = 1/||W^+||. \quad \kappa(W) = ||W|| \cdot ||W^+|| = \sigma_1(W)/\sigma_\rho(W) \geq 1$ denotes the condition number of a matrix $W$.

6. The $\xi$-rank of a matrix, for a positive $\xi$, is the minimum rank of its approximations within the norm bound $\xi$. The numerical rank of a matrix is its $\xi$-rank for $\xi$ small in context.

7. A matrix $W$ is ill-conditioned if its condition number is large in context or equivalently if its rank exceeds its numerical rank. The matrix is well-conditioned if its condition number is reasonably bounded. The ratio of the output and input error norms of Gaussian elimination is roughly the condition number of an input matrix (cf. [GL13]).
B Random matrices

B.1 Definitions and expected strong nonsingularity

We use the acronym “i.i.d.” for “independent identically distributed”, keep referring to standard Gaussian random variables just as Gaussian, and call random variables uniform over a fixed finite set if their values are sampled from this set under the uniform probability distribution on it.

Defining a random matrix $H$, we assume that its entries are linear combinations of finitely many i.i.d. random variables, under the Gaussian or uniform probability distribution. The matrix is Gaussian if all its entries are i.i.d. Gaussian variables.

**Theorem B.1.** Assume a nonsingular $n \times n$ matrix $A$ and an $n \times n$ matrix $H$ whose entries are linear combinations of finitely many i.i.d. random variables.

Let $\det((AH)_{l,l})$ vanish identically in them for neither of the integers $l, l = 1, \ldots, n$.

(i) If the variables are uniform over a set $S$ of cardinality $|S|$, then the matrix $A^H$ is singular with a probability at most $l/|S|$, for any $l$, and the matrix $AH$ is strongly nonsingular with a probability at least $1 - 0.5(n - 1)n/|S|$.

(ii) If these i.i.d. variables are Gaussian, then the matrix $AH$ is strongly nonsingular with probability 1.

**Proof.** Part (i) of the theorem follows from a celebrated lemma of [DL78], also known from [Z79] and [S80]. Derivation is specified, e.g., in [PW08]. Part (ii) follows because the equation $\det((AH)_{l,l})$ for any integer $l$ in the range from 1 to $n$ defines an algebraic variety of a lower dimension in the linear space of the input variables (cf. [BV88, Proposition 1]).

B.2 Rotational invariance and the condition number of a Gaussian matrix

**Lemma B.1.** (Rotational invariance of a Gaussian matrix.) Suppose that $k$, $m$, and $n$ are three positive integers, $G$ is an $m \times n$ Gaussian matrix, and $S$ and $T$ are $k \times m$ and $n \times k$ orthogonal unitary matrices, respectively. Then $SG$ and $GT$ are Gaussian matrices.

Next we recall some estimates for the norm and the condition number of a Gaussian matrix. For simplicity we assume that we deal with real matrices, but similar estimates in the case of complex matrices can be found in [D88], [E88], [CD05], and [ES05].

Hereafter we write $\nu^*_m,n = ||G||$ for a Gaussian $m \times n$ matrix $G$, and write $\mathbb{E}(v)$ for the expected value of a random variable $v$.

**Theorem B.2.** (Cf. [DS01, Theorem II.7].) Suppose that $m$ and $n$ are positive integers, $h = \max\{m, n\}$, $t \geq 0$, and $z \geq 2\sqrt{h}$. Then (i) Probability $\{\nu^*_m,n > t + \sqrt{m} + \sqrt{n}\} \leq \exp(-t^2/2)$ and (ii) $\mathbb{E}(\nu^*_m,n) < 1 + \sqrt{m} + \sqrt{n}$.

**Theorem B.3.** Let $\Gamma(x) = \int_0^\infty \exp(-t)x^{-1}dt$ denote the Gamma function and let $x > 0$. Then

(i) Probability $\{\nu^*_m,n \geq m/x^2\} \leq \frac{\Gamma(m/n + 1)}{(m/n)^{m/n}}$ for $m \geq n \geq 2$,

(ii) Probability $\{\nu^*_m,n \geq x\} \leq 2.35\sqrt{n}/x$ for $n \geq 2$,

(iii) $\mathbb{E}(\nu^*_m,n,m,n) = m/|m - n|$, provided that $|m - n| > 1$, and

(iv) $\mathbb{E}(\nu^*_m,n,m,n) \leq e\sqrt{m}/|m - n|$, provided that $m \neq n$.

**Proof.** See [CD05, Proof of Lemma 4.1] for part (i), [SST06, Theorem 3.3] for part (ii), and [HMT11, Proposition 10.2] for parts (iii) and (iv).

Probabilistic upper bounds on $\nu^*_m,n$ of Theorem B.3 are reasonable already for square matrices, for which $m = n$, but become much stronger as the difference $|m - n|$ grows large.

Theorems B.2 and B.3 combined imply that an $m \times n$ Gaussian matrix is very well-conditioned if the integer $m - n$ is large or even moderately large, and still can be considered well-conditioned if the integer $|m - n|$ is small or even vanishes (possibly with some grain of salt in the later case). These properties are immediately extended to all submatrices because they are also Gaussian.
C  Matrices of discrete Fourier transform

Definition C.1. Write $\omega = \exp\left(\frac{2\pi i}{n}\right)$, $\Omega = \Omega_n = (\omega^i)_{i,j=0}^{n-1}$, $\Omega^H$ is unitary, $\Omega^{-1} = \frac{1}{n}\Omega^H$, $\omega$ denotes a primitive $n$-th root of unity, $\Omega$ and $\Omega^{-1}$ denote the matrices of the discrete Fourier transform at $n$ points and its inverse, to which we refer as $DFT(n)$ and $IDFT(n)$, respectively,

$$\Omega = \Omega_n = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^{n-1} & \omega^{2n-2} & \omega^{3n-3} & \cdots & \omega^{(n-1)^2} \end{pmatrix}.$$

Remark C.1. If $n = 2^k$ is a power of 2, we can apply the FFT algorithm and perform $DFT(n)$ and $IDFT(n)$ by using only $1.5n \log_2(n)$ and $1.5n \log_2(n) + n$ arithmetic operations, respectively. For an $n \times n$ input and any $n$, we can perform $DFT(n)$ and $IDFT(n)$ by using $cn \log(n)$ arithmetic operations, but for a larger constant $c$ (see [P01, Section 2.3]).

D  Circulant and f-circulant matrices

For a positive integer $n$ and a complex scalar $f$, define the $n \times n$ unit $f$-circulant matrix $Z_f = \begin{pmatrix} 0 & f & 0 & \cdots \\ I_{n-1} & 0 & f & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ 0 & \cdots & \cdots & 0 & f \end{pmatrix}$ and the $n \times n$ general $f$-circulant matrix $Z_f(v) = \sum_{i=0}^{n} v_i Z_f^i$,

$$Z_f = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 & f \\ 1 & \ddots & \ddots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \cdots & \cdots & \vdots \\ \vdots & \ddots & \ddots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 1 & 0 \end{pmatrix} \quad \text{and} \quad Z_f(v) = \begin{pmatrix} v_0 & f v_{n-1} & \cdots & f v_1 \\ v_1 & v_0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ v_{n-1} & \cdots & \cdots & v_1 \end{pmatrix}.$$

$Z_f(v)$ is a lower triangular Toeplitz matrix for $f = 0$, circulant for $f = 1$, and skew-circulant for $f = -1$. $Z_f^2 = f I_n$, and the matrix $Z_f(v)$ is defined by its first column $v = (v_i)_{i=0}^{n-1}$. $Z = Z_0$ is the unit down-shift matrix.

We call an $f$-circulant matrix a Gaussian $f$-circulant (or just Gaussian circulant if $f = 1$) if its first column is filled with independent Gaussian variables. For every fixed $f$, the $f$-circulant matrices form an algebra in the linear space of $n \times n$ Toeplitz matrices

$$T = (t_{i-j})_{i,j=0}^{n-1}. \quad \text{(D.1)}$$

Hereafter, for a vector $u = (u_i)_{i,j=0}^{n-1}$, write $D(u) = \text{diag}(u_0, \ldots, u_{n-1})$, that is, $D(u)$ is the diagonal matrix with the diagonal entries $u_0, \ldots, u_{n-1}$.

Theorem D.1. (Cf. [CPW74].) If $f \neq 0$, then $f^n$-circulant matrix $Z_f^\ast(v)$ of size $n \times n$ can be factored as follows, $Z_f^\ast(v) = U_{f}^{-1} D(U_{f} v) U_{f}$ for $U_{f} = \Omega D(f)$, $f = (f^i)_{i=0}^{n-1}$, and $f \neq 0$. In particular, for circulant matrices, $D(f) = I$, $U_{f} = \Omega$, and $Z_f^\ast(v) = \Omega^{-1} D(\Omega v) \Omega$.

Remark D.1. We cannot apply this theorem directly to a triangular Toeplitz (0-circulant) matrix, but we can represent such a matrix (as well as any square Toeplitz matrix) as the sum of a circulant matrix and a skew-circulant one and then multiply this sum by a vector at roughly the double computational cost, compared to the case of $f \neq 0$.

Remark D.2. For an $n \times n$ Toeplitz or Toeplitz-like matrix $A$ and an $n \times n$ circulant matrix $H$, one can compute a standard displacement representation of the product $AH$ by applying just $O(n \log(n))$ flops (see definitions and derivation in [P01]).
Corollary D.1. (i) Suppose that we are given a diagonal matrix $D(u) = \text{diag}(u_0, \ldots, u_{n-1})$ for $u = \Omega v$. Then we can recover the vector $v = \frac{1}{\sqrt{n}} \Omega R u$, which defines the entries of the circulant matrix $Z_1(v)$.

(ii) If the vector $v$ is Gaussian, then so is also the vector $u = (u_i)_{i=1}^n = \frac{1}{\sqrt{n}} \Omega v$ (by virtue of Lemma B.1) and vice versa. Each of the two vectors defines a Gaussian circulant matrix $Z_1(v)$.

(iii) By choosing $u_i = \exp(\frac{\phi_i \sqrt{-1}}{\sqrt{n}})$ and real Gaussian variable $\phi_i$ for all $i$, we arrive at a random real orthogonal or unitary $n \times n$ circulant matrix $Z_1(v)$ defined by $n$ real Gaussian parameters $\phi_i$, $i = 0, \ldots, n - 1$. Alternatively we can set $\phi_i = \pm 1$ for all $i$ and choose the signs $\pm$ at random, with i.i.d. probability $1/2$ for each sign.

(iv) By adding another Gaussian parameter $\phi$, we can define a random real orthogonal or unitary $f$-circulant matrix $Z_1(v)$ for $f = \exp(\frac{\phi \sqrt{-1}}{\sqrt{n}})$.

The following results of [PSZ15] imply that a Gaussian circulant matrix is well-conditioned with a probability close to 1.

Theorem D.2. Suppose that $Z_1(v) = \Omega R D \Omega$ is a nonsingular circulant $n \times n$ matrix, and let $D(g) = \text{diag}(g_i)^{n}_{i=1}$, for $g = (g_i)^{n}_{i=1}$. Then $||Z_1(v)|| = \max_{i=1}^{n} |g_i|$, $||Z_1(v)^{-1}|| = \min_{j=1}^{n} |g_j|$, and $\kappa(Z_1(v)) = \max_{i,j=1}^{n} |g_j/g_i|$, for $v = \Omega^{-1} g$.

Remark D.3. Suppose that a circulant matrix $Z_1(v)$ has been defined by its first column vector $v$ filled with the integers $\pm 1$ for a random choice of the i.i.d. signs $\pm$, each + and − chosen with probability $1/2$. Then, clearly, the entries $g_i$ of the vector $g = \Omega v = (g_i)^{n}_{i=1}$ satisfy $|g_i| \leq n$ for all $i$ in an $n$, and furthermore, with a probability close to 1, $\max_{i=1}^{n} \log(1/|g_i|) = O(\log(n))$ as $n \to \infty$.

Remark D.4. In the case of a Gaussian circulant matrix $Z_1(v)$, all the entries $g_i$ are i.i.d. Gaussian variables, and the condition number $\kappa(Z_1(v)) = \max_{i,j=1}^{n} |g_i/g_j|$ is not likely to be large.

Remark D.5. (Cf. Remark 2.1.) In the presence of rounding errors, the MBA algorithm of Remark 2.1 fails to solve Toeplitz linear systems of equations except for the inputs of small size because of severe numerical problems [B85], while pivoting is not an option for solving Toeplitz or Toeplitz-like linear systems of equations because it destroys the matrix structure. So preprocessing is badly needed in this case. Fortunately, preprocessing with appropriate randomized structured multipliers is likely to fix these problems. In particular, random circulant multiplication keeps Toeplitz structure intact (cf. [P01, Chapters 4 and 5]), and the computation of a standard displacement representation of the product of a Toeplitz or Toeplitz-like $n \times n$ matrix by a circulant matrix only requires $O(n \log(n))$ arithmetic operations (cf. [P01]) or just $O(n)$ multiplications if the circulant matrices involved are represented by using the factorization of Theorem D.1. One can similarly apply structured preprocessing in order to support the extension of the MBA algorithm to computations with other structured matrices (cf. [P01, Sections 5.6 and 5.7]).