# Numerical Computation of the Sign of the Determinant with Additive and Multiplicative Preconditioning

by

**Islam A.T.F. Taj-Eddin**

A dissertation
submitted to the Graduate Faculty in Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
The City University of New York

2007

This manuscript
has been read and accepted for the Graduate Faculty in Computer Science
in satisfaction of the dissertation requirement for the degree of
Doctor of Philosophy.

| | |
|---|---|
| Date | Dr. Victor Y. Pan, Chair of Examining Committee |
| Date | Dr. Theodore Brown, Executive Officer |

Dr. Victor Y. Pan, Adviser and Chair
The Graduate School and University Center, Lehman College

Dr. Michael Anshel
The Graduate School and University Center, The City College

Dr. Stathis Zachos
The Graduate School and University Center, Brooklyn College

Dr. Mohammad Tabanjeh
Virginia State University

Supervisory Committee

**THE CITY UNIVERSITY OF NEW YORK**

## Abstract


**NUMERICAL COMPUTATION OF THE SIGN OF THE DETERMINANT WITH ADDITIVE AND MULTIPLICATIVE PRECONDITIONING**

by

**Islam A.T.F. Taj-Eddin**

**Advisor: Distinguished Professor Dr.Victor Y. Pan**

Accurate computation of the sign and the value of a matrix determinant attracts a great deal of attention. Various algebraic and geometric computations boil down to it. This includes the computation of a convex hull and a Voronoi diagram as well as the evaluation and expansion of scalar, univariate and multivariate resultants.

In the present day computing environment, it is most effective to compute determinants numerically with IEEE standard double precision floating-point numbers provided rounding errors are controlled. That control is difficult where the input matrix is ill conditioned but easy where the matrix is well conditioned. This motivates the application of preconditioning methods.

In this thesis, recent techniques of additive preconditioning are applied, the technicalities of this application are elaborated, and the power of the approach is demonstrated with numerical experiments.

# Acknowledgements

In the name of ALLAH the most Knowledgeable.

I would like to thank the following individuals :-

My adviser, Dr. Victor Y. Pan.

Dr. Michael Anshel, Dr. Stathis Zachos and Dr. Mohammad Tabanjeh, members of my Dissertation Examining Committee.

Dr. Theodore Brown, the Executive Officer of the Doctoral Program in Computer Science.

Dr. Stanley Habib, who was the Executive Officer of the Doctoral Program in Computer Science when I was admitted to the program.

Dr. Robert Feinerman, Chairman of the Department of Mathematics and Computer Science, Lehman College, The City University of New York.

Dr. Robert H. Lewis, Dr. Cris Poor, and Dr. William Hastings, Chairmen of the Department of Mathematics, Fordham College at Rose Hill, Fordham University.

Dr. Irene S. Leung, Department of Environmental, Geographic and Geological Sciences, Lehman College, The City University of New York.

The faculty of the Doctoral Program in Computer Science at The Graduate School and University Center of The City University of New York, and the faculty of the Department of Mathematics and Computer Science at Lehman College of The City University of New York.

The faculty of the Computer Science Department, College of Computer and Information Sciences, King Saud University, Riyadh, Kingdom of Saudi Arabia.

My parents.

# Contents

# List of Tables

# 1 Definitions and preliminary results

## 1.1 Matrices

Consider the square matrix $A$, $A = (\mathbf{a}_i)_{i=1}^n = (a_{i,j})_{i,j=1}^n$ is an $n \times n$ matrix of *dimension* $n$, with elements $a_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq n$,

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{3,1} & a_{3,2} & a_{3,3} & \dots & a_{3,n} \\ a_{n,1} & a_{n,2} & a_{n,3} & \dots & a_{n,n} \end{pmatrix}$$

$\mathbf{a}_i = (a_{i,j})_{j=1}^n$ is its $i$th column vector. $A_{i,j} = a_{i,j}$ is its $(i,j)$th entry.

The *determinant* of $A$, written as det(A) is a function of this matrix,

$$\det(A) = \sum_{i=1}^{n!} (p[\pi_i(n)]. \prod_{j=1}^{n} a_{j,\pi_{i,j}}) \tag{1.1}$$

where $\pi_i(n)$ is the $i$th distinct *permutation* on the indices $< 1, 2, \dots, n >$ and $\pi_{i,j}$ is the $j$th member of the $i$th permutation. $p[\pi_i(n)]$ is the *parity* of the $i$th permutation, which is $+1$ or $-1$ according to whether the number of inversions in the permutation is even or odd. An inversion occurs whenever an index in the permutation list is greater than one which follows. For instance, suppose $\pi_j(6) =< 6\ 1\ 4\ 2\ 3\ 5 >$, then the number of inversions in this list is 7 and $p[\pi_i(n)] = -1$. As an example of the foregoing, consider the $3 \times 3$ matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & -1 & 2 \\ 3 & 3 & 1 \end{pmatrix} \tag{1.2}$$

with index permutation data displayed in table 1.1. Then the computation of this determinant goes as follows: $\det(A) = 1 \times (-1) \times 1 - 1 \times 2 \times 3 + 2 \times 2 \times 3 - 2 \times 2 \times 1 + 3 \times 2 \times 3 - 3 \times (-1) \times 3 = -1 - 6 + 12 - 4 + 18 + 9 = 28$. The cost of computing the determinant by using equation (1.1) is $n!$ multiplications and $(n-1)!$ additions. The complexity is not satisfactory for the practical evaluation of large matrices.

| j | $\pi_j(3)$ | inversions | $p[\pi_i(3)]$ |
|---|---|---|---|
| 1 | 1 2 3 | 0 | +1 |
| 2 | 1 3 2 | 1 | -1 |
| 3 | 2 3 1 | 2 | +1 |
| 4 | 2 1 3 | 1 | -1 |
| 5 | 3 1 2 | 2 | +1 |
| 6 | 3 2 1 | 3 | -1 |

Table 1.1: Permutation Data

Hereafter computations are assumed in the field of real numbers. "Ops" stands for "arithmetic operations".

$|A|$ is the matrix $(|a_{i,j}|)_{i,j=1}^n$. Write $|A| \leq |W| \iff W = (w_{i,j})_{i,j=1}^n$ and $(|a_{i,j}|)_{i,j=1}^n \leq (|w_{i,j}|)_{i,j=1}^n$ for all $i$ and $j$.

$A^T = (a_{j,i})_{i,j}$ is the transpose of $A = (a_{i,j})_{i,j}$.

$I_k$ is the $k \times k$ identity matrix.

$0_k$ is the $k \times k$ null matrix.

rank$A$ is its rank.

An $m \times n$ matrix $A$ is *orthogonal* if $A^T A = I_n$, is *lower triangular* if $a_{i,j} = 0$ for $i > j$, is *upper triangular* if $a_{i,j} = 0$ for $i < j$, and is *diagonal* if $a_{i,j} = 0$ for $i \neq j$.

$A = \mathrm{diag}(a_{i,i})_{i=1}^n$ is an $n \times n$ diagonal matrix. $A = \mathrm{diag}(B, C)$ is a $2 \times 2$ block diagonal matrix with diagonal blocks $B$ and $C$.

For an $m \times n$ matrix $A$ of rank $\rho$, its left nullity $\mathrm{lnul}\, A = n - \rho$ and its right nullity $\mathrm{rnul}\, A = m - \rho$ are the dimensions of its left and right null spaces $N(A)$ and $LN(A)$, respectively. Its *nullity* $\mathrm{nul}\, A = \min(m - \rho, n - \rho)$ is the minimum of $\mathrm{lnul}\, A$ and $\mathrm{rnul}\, A$.

$\nu$-nullity is the total number of singular values $\sigma$ of the $SVD(A) \leq \nu$.

$\epsilon$ denotes the *unit roundoff*, also called *machine epsilon*.

The following properties of determinants are well known and readily verified (see, e.g.[H64], [CD80]).

**Theorem 1.1.** *Consider an $n \times n$ matrix $A$. The following properties are true:*

- *If any columns or rows of $A$ is a linear combination of any set of other rows or columns, respectively, then* $\det(A) = 0$.
- *If* $\det(A) \neq 0$ *then $A$ has full rank (i.e.* $\operatorname{rank}(A) = n$*).*
- *Interchanging any two columns or rows of $A$ changes the sign of* $\det(A)$.
- $\det(A) = \det(A^T)$ *for all square matrices $A$.*
- $|\det(Q)| = 1$ *if $Q$ is an orthogonal matrix.*
- $\det(A) = \det(B) \times \det(C)$ *if $A = B \times C$ as well as if* $A = \begin{pmatrix} B & X \\ 0 & C \end{pmatrix}$.
- $\det(A) = \prod_{i=1}^{n} a_{i,i}$ *if $A$ is a triangular matrix.*

*The matrix $A^{-1}$ is called the* inverse *to $A$ if $A \times A^{-1} = A^{-1} \times A = I$. A matrix whose determinant is zero is said to be a* singular *matrix, and a matrix has an inverse if and only if it is non-singular.*

## 1.2 Matrix factorizations

**Definition 1.1.** *$A = PLUP_1$ is a $PLUP_1$ factorization of a (generally rectangular) matrix $A$ if $L$ and $U^T$ are lower triangular matrices and $P$ and $P_1$ are permutation matrices. This includes $LUP_1$, $PLU$, and $LU$ factorizations as special cases where $P = I$ and/or $P_1 = I$.*

One can compute $PLUP_1$, $PLU$, $LUP$ and $LU$ factorizations by applying Gaussian elimination with complete pivoting, partial pivoting (with row or column permutations), and without pivoting, respectively. For an $n \times n$ matrix these computations use $(\frac{2}{3})n^3$ ops; also $(\frac{2}{3})n^3$ and $O(n^2)$ comparisons are required for complete and partial pivoting, respectively [GV96], [H02], [S98a]. Here and hereafter drop the lower order terms in the ops and comparisons bounds.

**Definition 1.2.** *$A = QR$ is a $QR$ factorization of a (generally rectangular) matrix $A$ if $R$ is an upper triangular matrix with nonnegative diagonal entries and $Q$ is an orthogonal matrix.*

The factorization is unique if the matrix $A$ has full rank. One can compute numerically the $R$-factor in the $QR$ factorization of an $n \times n$ matrix in $(\frac{4}{3})n^3$ ops by applying Householder reflections or in $2n^3$ ops by applying either Givens rotations or alternatively the modified Gram-Shmidt algorithm [GV96, Section 5.2].

Besides ops, computation of a $QR$ factorization of an $n \times n$ matrix requires computation of $O(n)$ square roots. This is a minor part of the overall computational cost, but the implied rounding errors can be unpleasant. A rational version of the modified Gram-Schmidt algorithm in [C92] avoids computing square roots and contains only rational ops.

$QRP$ factorization of a (generally rectangular) matrix is its QR factorization with column pivoting [GV96, Section 5.4.1] , [S98a, Algorithms 4.1.2 and 5.2.1] (see also [GV96, Section 5.4.2]). It is computed in $(\frac{4}{3})n^3$ ops (due to Businger and Golub 1965), see [GV96, Algorithm 5.4.1] and [S98a, Algorithm 5.2.1]. Pivoting enables the bounds $|r_{k,k}|^2 \geq \sum_{i=1}^{k-1}$, $k = 2, \ldots, n$. A *random unitary* matrix is the Q-factor in QRP factorization of a random matrix, which is the Q-factor in its QR factorization if this is a matrix of full rank.

## 1.3 Matrix norms, Condition number

**Definition 1.3.** *$||A|| = max_{\mathbf{v}:||\mathbf{v}||=1} \frac{||a\mathbf{v}||}{\mathbf{v}}$ is an operator norm of a matrix $A$ consistent with a vector norm $||\mathbf{v}||$. $||.|| = ||.||_h$ denotes a fixed operator matrix norm, in particular the 2-norm $||.||_2$ will be used, the row norm $||A||_\infty = max_i \sum_j |a_{i,j}|$, and the column norm $||A||_1 = max_j \sum_i |a_{i,j}|$ for a matrix $A = (a_{i,j})_{i,j=1}^n$ (cf. [GV96] or [H02]).*

**Theorem 1.2.** *For a nonsingular matrix $A$ and any operator matrix norm $||.||$ the following is true $min_B||B-A|| = \frac{1}{||A^{-1}||}$ where the minimum is over all singular matrices $B$.*

*Proof.* See [EY39], [K66]. □

$||A||_2 = \sigma_1$ is the 2-norm of a matrix $A$, consistent with the Euclidean vector norm $||\mathbf{v}|| = (\sum_j |v_j|^2)^{\frac{1}{2}}$.

$$\frac{1}{||A^{-1}||_2} = \sigma_n \tag{1.3}$$

for a nonsingular $n \times n$ matrix $A$.

**Definition 1.4.** *The ratio $cond_2 A = \frac{\sigma_1}{\sigma_r}$ is called the condition number of a matrix $A$ of a rank $r$. $cond_2 A = ||A||_2||A^{-1}||_2$ for a nonsingular matrix $A$. $cond_2 A$ for a matrix $A$ of a rank $r$ is the ratio $\frac{\sigma_1(A)}{\sigma_{r(A)}}$ of its largest and smallest singular values. $Cond_2 A = ||A||_2||A^{-1}||_2$ for a nonsingular matrix $A$. For two nonnegative integers $g$ and $h$, $g+h<r$, define the $(g,h)$-condition number of the matrix $A$ as the value $cond_{2,(g,h)}A = cond_2(A_{(g,h)}) = \frac{\sigma_{g+1}}{\sigma_{r-h}}$.*

## 1.4 Schur complements and the Sherman–Morrison–Woodbury formula

For a $2 \times 2$ block matrix

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},$$

the matrix $S_{22} = B_{22} - B_{21}B_{11}^{-1}B_{12}$ (respectively, $S_{11} = B_{11} - B_{12}B_{22}^{-1}B_{21}$) is the *Schur complement* of the northwestern block $B_{11}$ (respectively, the southeastern block $B_{22}$) in the matrix $B$ provided $B_{11}^{-1}B_{11} = I$ and/or $B_{11}B_{11}^{-1} = I$ (respectively, $B_{22}^{-1}B_{22} = I$ and/or $B_{22}B_{22}^{-1} = I$) [GV96, page 103], [S98a, page 155]. The following lemma will be verified.

**Lemma 1.1.** *Let the above block matrix $B$ be nonsingular and let*

$$B^{-1} = \begin{pmatrix} W & X \\ Y & Z \end{pmatrix}$$

*for some matrices $W$, $X$, $Y$ and $Z$. Then $W = S_{11}^{-1}$ (resp. $Z = S_{22}^{-1}$) if the block $B_{11}$ (resp. $B_{22}$) is nonsingular.*

**Theorem 1.3.** *For $n \times r$ matrices $U$ and $V$ and an $n \times n$ matrices $A$, let the matrix $C = A + UV^T$ be nonsingular. Then the matrices $A$ and $S = I_r - V^T C^{-1} U$ are the respective Schur complements of the blocks $I_r$ and $C$ in the matrix*

$$W = \begin{pmatrix} C & U \\ V^T & I_r \end{pmatrix}$$

*such that*

$$\det(W) = \det(A) = (\det(C)) \det(S). \tag{1.4}$$

*Furthermore [GV96, page 50], [S98a, Corollary 4.3.2], if the matrix $A$ is nonsingular, then so is the matrix $S$, and the Sherman–Morrison–Woodbury formula $(C - UV^T)^{-1} = C^{-1} + C^{-1} U S^{-1} V^T C^{-1}$ holds.*

## 1.5 Random matrices

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

A *random unitary* matrix is the Q-factor in a QRP factorization of a random matrix or the Q-factor in its QR factorization if the matrix has full rank.

**Lemma 1.2.** *[DL78] (cf. also [Z79], [S80]). For a finite set $\Delta$ of cardinality $|\Delta|$, let a polynomial in $m$ variables have total degree $d$ and not vanish identically on the set $\Delta^m$ and let the values of its variables be randomly sampled from the set $\Delta$. Then the polynomial vanishes with a probability of at most $d/|\Delta|$.*

## 1.6 The SVDs

The *compact Singular Value Decomposition* of an $m \times n$ matrix $A$ of a rank $\rho$ (also called the *compact SVD* of this matrix) is the decomposition

$$A = S^{(\rho)}\Sigma^{(\rho)}T^{(\rho)T} = \sum_{j=1}^{\rho} \sigma_j \mathbf{s}_j \mathbf{t}_j^T$$

where $S^{(\rho)} = (\mathbf{s}_j)_{j=1}^{\rho}$ and $T^{(\rho)} = (\mathbf{t}_j)_{j=1}^{\rho}$ are unitary matrices, $S^{(\rho)T}S^{(\rho)} = I^{(\rho)}$, $T^{(\rho)T}T^{(\rho)} = I^{(\rho)}$, $\Sigma^{(\rho)} = \mathrm{diag}(\sigma_j)_{j=1}^{\rho}$ is a diagonal matrix, $\mathbf{s}_j$ and $\mathbf{t}_j$ are $m$- and $n$-dimensional vectors, respectively, and $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_\rho > 0$. $||A||_2 = \sigma_1$ and $||A||_F^2 = \sum_{j=1}^{\rho} \sigma_j^2$.

Write $l = \mathrm{lnul}\, A = m - \rho$, $r = \mathrm{rnul}\, A = n - \rho$ and, for a pair $S^{(\mathrm{nul})} = (\mathbf{s}_j)_{j=\rho+1}^{m}$ and $T^{(\mathrm{nul})} = (\mathbf{t}_j)_{j=\rho+1}^{n}$ of left and right unitary null matrix bases for the matrix $A$, define the square unitary matrices $S = (S^{(\rho)}, S^{(\mathrm{nul})}) = (\mathbf{s}_j)_{j=1}^{m}$ and $T = (T^{(\rho)}, T^{(\mathrm{nul})}) = (\mathbf{t}_j)_{j=1}^{n}$ and the $m \times n$ matrix $\Sigma = \mathrm{diag}(\Sigma^{(\rho)}, 0_{l,r})$. The equation $A = S\Sigma T^T$ is the *Singular Value Decomposition* of the matrix $A$, also called its *SVD* and *full SVD*.

Hereafter write $\sigma_j = 0$ for $j > \rho$ and $\sigma_j = +\infty$ for $j < 1$. The scalars $\sigma_j$ for $j \ge 1$ are the *singular values* of the matrix $A$. The vectors $\mathbf{s}_j$ for $j = 1, \ldots, m$ and $\mathbf{t}_j$ for $j = 1, \ldots, n$ are the associated left and right *singular vectors*, respectively, so that the null vectors are the singular vectors associated with the singular value zero.

$A\mathbf{t}_j = \sigma_j \mathbf{s}_j$ and $\mathbf{s}_j^T A = \sigma_j \mathbf{t}_j^T$ for all $j$, $A^T = T\Sigma^T S^T$, $A^T A = T\Sigma^T \Sigma T^T$, and $AA^T = S\Sigma\Sigma^T S^T$.

## 1.7 Generalized inverses and conditioning of a matrix

$\mathrm{cond}(A) = \sigma_1(A)/\sigma_\rho(A) = ||A||_2||A^{-1}||_2$ is the condition number of a matrix $A$ of a rank $\rho$ (under the 2-norm). Effective norm and condition estimators can be found in [GV96, Section 3.5.4] and [S98a, Section 5.3].

Write $n >> d$ where the ratio $n/d$ is large.

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

**Fact 1.1.** *The matrices $A^T$ and $A^{-1}$ share their singular spaces, whereas*

$$\sigma_j(A^T) = \sigma_j(A) = 1/\sigma_{\rho+1-j}(A^{-1}) \quad for \quad j = 1, \dots, \rho, \quad \rho = \mathrm{rank}\, A,$$

$$\sigma_j(A^T) = \sigma_j(A) = \sigma_j(A^{-1}) = 0 \quad for \quad j > \mathrm{rank}\, A.$$

## 1.8 The $g$-head, $h$-tails, extended $h$-tails, and $(g, h)$-matrices

The *g-head*, *$(g,h)$-residue*, *h-tail*, and *extended h-tail* of the SVD are the triples $(S^{(g)}, \Sigma^{(g)}, T^{(g)})$, $(S_{g,h}, \Sigma_{g,h}, T_{g,h})$, $(S_h, \Sigma_h, T_h)$, and $(S_h^{(\mathrm{ext})}, \Sigma_h^{(\mathrm{ext})}, T_h^{(\mathrm{ext})})$, respectively, where $g$ and $h$ are two nonnegative integers, $g + h \le \rho$,

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

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

$$\Sigma_{g,h} = \mathrm{diag}(\sigma_j)_{j=g+1}^{\rho-h}, \quad \Sigma_h = \mathrm{diag}(\sigma_j)_{j=\rho-h+1}^{\rho}, \quad \Sigma_h^{(\mathrm{ext})} = \mathrm{diag}(\Sigma_h, 0_{l,r}),$$

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

$$T_h = (\mathbf{t}_j)_{j=\rho-h+1}^{\rho}, \quad T_h^{(\mathrm{ext})} = (T_h, T^{(nul)}) = (\mathbf{t}_j)_{j=\rho-h+1}^{n},$$

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

The spaces generated by the columns of the matrices $S^{(g)}$, $T^{(g)}$, $S_h^{(\mathrm{ext})}$, and $T_h^{(\mathrm{ext})}$ are the *left* and *right g-leading* and *h-trailing singular spaces* of the matrix $A$, respectively.

For a matrix $A$ of a rank $\rho$ and two nonnegative integers $g$ and $h \le \rho - g$, define the condition numbers for its $g$-head, $(g,h)$-residue, and $h$-tail as the ratios $\sigma_1/\sigma_g$, $\sigma_{g+1}/\sigma_{\rho-h}$, and $\sigma_{\rho-h+1}/\sigma_\rho$, respectively. The $g$-head, $(g,h)$-residue, and $h$-tail are *ill conditioned* if the respective ratios are large and are *well conditioned* otherwise. A matrix $A$ of a rank $\rho$ is a $(g,h)$ *matrix* if its $(g,h)$-residue is well conditioned, whereas its $(g+1)$-head and $(h+1)$-tail are ill conditioned. A $(g,h)$ *matrix* is a *strictly* $(g,h)$ *matrix* if its $g$-head and $h$-tail are well conditioned, whereas $\sigma_g >> \sigma_{g+1}$ and $\sigma_{\rho-h} >> \sigma_{\rho-h+1}$. In this paper the $(g,h)$-matrices are usually $(g,0)$ or $(0,h)$ matrices.

## 1.9 $(g,h)$-SVDs

Write

$$\begin{aligned} A^{(g)} &= \sum_{j=1}^{g} \sigma_j \mathbf{s}_j \mathbf{t}_j^T &&= S^{(g)} \Sigma^{(g)} T^{(g)T}, \\ A_{g,h} &= \sum_{j=g+1}^{\rho-h} \sigma_j \mathbf{s}_j \mathbf{t}_j^T &&= S_{g,h} \Sigma_{g,h} T_{g,h}^T, \quad \text{and} \\ A_h &= \sum_{j=\rho-h+1}^{\rho} \sigma_j \mathbf{s}_j \mathbf{t}_j^T &&= S_h \Sigma_h T_h^T. \end{aligned}$$

Observe that

$$\begin{aligned} A &= A^{(g)} + A_{g,h} + A_h, \\ S &= (S^{(g)}, S_{g,h}, S_h, S^{(\mathrm{nul})}), \\ \Sigma &= \mathrm{diag}(\Sigma^{(g)}, \Sigma_{g,h}, \Sigma_h, 0_{l,r}), \quad \text{and} \\ T &= (T^{(g)}, T_{g,h}, T_h, T^{(\mathrm{nul})}). \end{aligned}$$

Now, for a pair of unitary matrices $S_{g,h}^{(ORT)}$ of size $m \times (\rho - g - h)$ and $T_{g,h}^{(ORT)}$ of size $n \times (\rho - g - h)$ such that

$$\begin{aligned} S_{g,h}^{(ORT)T}(S^{(g)}, S_h, S^{(\mathrm{nul})}) &= 0, \\ T_{g,h}^{(ORT)T}(T^{(g)}, T_h, T^{(\mathrm{nul})}) &= 0, \end{aligned} \tag{1.5}$$

write $l = \mathrm{lnul}\, A$ and $r = \mathrm{rnul}\, A$, define the pair of unitary matrices

$$\begin{aligned} \tilde{S}_{g,h} &= (S^{(g)}, S^{(ORT)}_{g,h}, S_h, S^{(\mathrm{nul})}), \\ \tilde{T}_{g,h} &= (T_g, T^{(ORT)}_{g,h}, T_h, T^{(\mathrm{nul})}), \end{aligned} \tag{1.6}$$

and obtain the factorizations

$$A = \tilde{S}_{g,h} \begin{pmatrix} \Sigma^{(g)} T^{(g)T} \\ S^{(ORT)T}_{g,h} A \\ \Sigma_h T_h^T \\ 0_{l,n} \end{pmatrix}, \tag{1.7}$$

$$A = (S^{(g)}\Sigma^{(g)}, AT^{(ORT)}_{g,h}, S_h\Sigma_h, 0_{m,r})\tilde{T}^T_{g,h}, \tag{1.8}$$

$$A = \tilde{S}_{g,h}\, \mathrm{diag}(\Sigma^{(g)}, S^{(ORT)T}_{g,h} AT^{(ORT)}_{g,h}, \Sigma_h, 0_{l,r})\tilde{T}^T_{g,h}, \tag{1.9}$$

which had been called the *left* $(g,h)$-*SVD*, the *right* $(g,h)$-SVD, and the $(g,h)$-SVD or the *two-sided* $(g,h)$-SVD, respectively.

## 1.10 MPPs and APPs (definitions)

**Definition 1.5. Multiplicative preprocessors.** *Any pair of nonsingular matrices $M$ of size $m \times m$ and $N$ of size $n \times n$ is an* MPP *for a matrix $A$ of size $m \times n$, whereas the transition $A \leftarrow MAN$ is its* M-preprocessing. *Such an MPP is an MPC and the* M-preprocessing *is* M-preconditioning *if* $\mathrm{cond}(A) << \mathrm{cond}(MAN)$. *Such an MPP is an* MC *if the matrix $A$ is rank deficient, whereas the matrix $MAN$ turns into a full rank matrix after the deletion of its zero rows and columns. If one of the matrices in the pair of $M$ and $N$ is the identity matrix, then the other matrix is also called an MPP, MPC or MC, respectively.*

**Definition 1.6. Additive preprocessors.** *For a pair of matrices $U$ of size $m \times r$ and $V$ of size $n \times r$, both having full rank $r > 0$, the matrix $UV^T$ (of rank $r$) is an* APP (of rank $r$) *for any $m \times n$ matrix $A$, the matrix $C = A + UV^T$ is its* A-modification, *the matrices $U$ and $V$ are* generators *of the APP, and the transition $A \leftarrow C$ is* A-preprocessing of rank $r$ *for the matrix $A$. An APP $UV^T$ for a matrix $A$*

*is an* APC *and A-preprocessing is* A-preconditioning *if* $\operatorname{cond}(A) >> \operatorname{cond}(C)$. *An APP is an* AC *and A-preprocessing is* A-complementation *if the matrix A is rank deficient, whereas the A-modification C has full rank. An APP* $UV^T$ *is* unitary *if the matrices U and V are unitary. An AC of rank r is* balanced *if* $r = \operatorname{nul} A$.

## 1.11 Some abbreviations

"CG" stands for "Conjugate Gradient",

"SMW" for "Sherman–Morrison–Woodbury",

"PPs" for "preprocessors",

"PCs" for "preconditioners",

"A" for "additive",

"ACs" for "additive complements",

"M" for "multiplicative",

"MCs" for "multiplicative compressors", and

"ASAs" for "advanced algorithms for floating-point summation".

Also a combination of abbreviations is used, as in the previous two subsections, so that

"MPPs" stands for "multiplicative preprocessors",

"APPs" for "additive preprocessors",

"MPCs" for "multiplicative preconditioners", and

"APCs" for "additive preconditioners".

# 2 Introduction

The classical problem of computing det(A), the determinant of an $n \times n$ matrix $A$, and its sign, has a long history and turns out to be important in geometric and algebraic computations (see, e.g. [G72], [M60], [D61], [M55], [F64], [R52], [E67], [B68], [P88], [BP94]).

It turns out that some of the most fundamental problems of algebraic computations and computational geometry (such as the computation of convex hulls, Voronoi diagrams and the arrangement of lines and line segments) are reduced to the computation of $\det(A)$ or, more precisely, its sign, that is, testing whether $\det(A) = 0$, $\det(A) > 0$, or $\det(A) < 0$ for an associated $n \times n$ matrix $A$, whereas many multivariate polynomial computations involve the evaluation and expansion of scalar, univariate and multivariate resultants, which are the determinants of some structured matrices (see [EP05, Sections 5-7] and the bibliography therein, [BKMNSU95], [BEPP99], [ABDPY97], [EP05], [EC95], [FV93], [Y97], [YD95], [AF92], [MA93], [FR94], [FV93], [E98], [BEPP99]).

In these cases the input data can be small. In many areas of computational geometry, lower dimensional problems must be solved, and then $n$ ranges between 2 and 10, usually staying below 5. In this class of applications, the matrix $A$ is filled with "long" numbers, representing the real data with a high precision (and thus allowing to treat the important case of a nearly singular input). In another major class of applications [DL94], [DL97], [ES87], [AF92], [FR94], [MA93], $n$ is large (say, in the range from 100 to 500), whereas the matrix $A$ is filled with relatively short integers (say, represented with 5 to 10 bits). Such applications include the computation of the orientation of a polyhedron or an algebraic variety in a high-dimensional space (for instance, such computations are required in the area of convex optimization in statistical physics and chemistry).

Let us comment on the affect of the input and rounding errors on the validity of the computed value of sign(det(A)). If the absolute value of the error is denoted by $e(A)$ and the correct value of the determinant by $D$, then the computed result is $D - e(A) < \det(A) < D + e(A)$. If $\det(A)$ is anywhere near $e(A)$ in magnitude, then the sign of $\det(A)$ is in question. Note also that the test whether $(\det(A)) = 0$ test is essentially meaningless in the presence of any error whatever. Happily, $e(A)$ grows at a lower order than $\det(A)$ in the dimension of a matrix and the input data size. So, in general, $e(A) < |\det(A)|$ when $|\det(A)|$ is large. And, also fortuitous, $|\det(A)|$ is large *most* of the time. Real trouble comes in the near singular cases, when $|\det(A)|$ is small. Sadly, the latter case is the quite interesting, e.g. in applications to convex hull computations.

Now consider the linear systems $Ax = f$ (where $A$ and $f$ are the input and $x$ is the output of an error-free solver) and $(A+E)x' = f$ (where $A$ and $f$ are the input to a numerical solver, $x'$ is the output, and the matrix $E$ represents the affect of rounding errors). Then (see, for instance [GV96, Sec. 2.5, pp. 24-28], [ORT, Sec. 2.1.1, pp. 32-35)]:

$$\frac{||x - x'||}{||x||} \leq \text{cond}_2(A) \times \frac{||E||}{||A||}$$

up to smaller order errors.

Here the vertical bars denote the pair of compatible vector and matrix norms. The essence of these bounds is that the relative error of the output is equal roughly to the condition number of the problem *multiplied by* the relative computational errors. The effect is that for poorly conditioned problems, that is, for the ones where $\text{cond}(A)$ is very large, moderate computational errors may easily be inflated to yield useless output.

Under these conditions, the behavior of the algorithm will be unpredictable; and it exhibits *instability*.

There are several known cures for these ills. One is to increase the operand size in software. This is the *multi-precision* approach. The idea has been widely used in the exact arithmetic (symbolic) algorithms [K98], [FV93], [YD95], [Y97]. This approach is expensive in the computational cost sense. A significant problem with this "BigNum" approach is of how much multi-precision one needs. It is difficult to tailor an algorithm so that the expensive multi-precision arithmetic would be applied only to computations that need it and to the extent they need it.

The cited applications and problems have motivated extensive algorithmic work on computing the value and the sign of the determinant (see [P87], [P88], [C92], [BKMNSU95], [FV96], [ABDPY97], [BEPP99], [ABM99], [EGV00], [PY01], [KV04a], [P04], [S04], and the bibliography therein). The cited papers, except for [PY01] and partly [C92], rely on algebraic methods, but in [PY01] numerical methods were applied. They use less computer time and memory space. In particular Gaussian elimination (with pivoting) is the customary approach to approximating the triangular factors $L$ and $U$ of the input matrix (up to row/column permutations). As by-product, this produces the determinant. It is critical, however, to certify that the rounding errors do not corrupt the output, in particular do not change the sign of $\det(A)$.

The sign and the values can be immediately obtained from $LU$ factorization of the matrix $A$ (with or without pivoting) as well as from its $QR$ factorization or $SVD$ [C92], [PY01]. Indeed, $\det(AB) = (\det(A))\det(B)$ and the determinants of triangular and orthogonal matrices are readily computable.

In numerical implementation, the sign is certified if $\sigma_n$, the smallest singular value of the matrix $A$, exceeds an upper estimate $e(A)$ for its factorization error [PY01] having (i.e. $\sigma_n(A) > e(A)$). From [H02], [S98a] then $e(A) \le c||A||\epsilon$ where $||A||$ is the 2-norm or the Frobenius norm of the matrix $A$, $\epsilon$ is the output relative

error for a fixed machine precision and is called the *unit roundoff* and also the *machine epsilon*, and $c$ is a small positive constant. Since the matrix $A$ could be normalized, so that $||A|| \approx 1$, this suggests decreasing the value $cond_2 A = \frac{\sigma_1}{\sigma_n}$ while keeping the sign of the determinant invariant or controlled.

The straightforward certification is by comparison of the magnitude of $\det(A)$ with the error bound $e(A)$ of its numerical computation. This works poorly where relying on computing the $SVD$ of the input matrix because the known error bounds on the output value of the determinant are proportional to the Hadamard's worst case bound on $|\det(A)|$ (Hadamard's bound is

$$|\det(A)| \leq \prod_{k=1}^{n} ||A_k||_2$$

where $A_k$ denotes the $k$-th column vector of $A$ (cf. [H02, p.287]). Thus this bound is overly pessimistic in the important case where the actual value of $|\det(A)|$ is not large. The improved certification recipe in [PY01] employs the factorization error norm bound $e_{LU} = ||LU(A) - A||_2$, which is typically much smaller than $e(A)$.

The sign is certified if $e_{LU}$ is exceeded by the smallest singular value $\sigma_n$ of the input matrix $A$.

To approximate the required extremal singular values (or all singular values) $\sigma_j$ and the associated singular vectors, any $SVD$ algorithm can be applied as long as it fits the assumed framework and cost bounds.

# 3 Previous work

In this section some previous works on algebraic and numerical computations of the sign of the determinant have been covered.

## 3.1 Symbolic approach to the sign and the value of the determinant

### 3.1.1 Introduction

The approach of *Arithmetic filtering* combines both symbolic and numerical methods to perform the computations both faster and correctly. One begins with numerical algorithms, which rapidly compute the certified correct output. If the algorithm fail to produce the reliable sign of $\det(A)$, then one relays the task to symbolic methods, which are slower but reliable (cf., e.g., [PYS97]). The main goal is to use faster numerical methods most of the time, and only in the rare cases where they fail, to resort to the slower symbolic algorithms.

### 3.1.2 LU Decomposition Algorithm

Many algorithms can be implemented both numerically and in modular arithmetic. Here, for the sake of comparison, is an example given by numerical and modular $LU$ matrix decomposition in the C language style.

**Algorithm 3.1. Numerical LU factorization.**

INPUT: *Matrix* $A[n][n]$*; int* $i$*,* $j$*,* $k$*;*

OUTPUT: $L$ *and* $U$*.*

COMPUTATIONS:

*for* $(i = 1; i < n; i++)$

*for* $(j = i + 1; j < n; j++)$ $\{$

$A[j][i] = (A[j][i]/A[i][i]);$

*for* $(k = i + 1; k < n; k++)$

$A[j][k] = A[j][k] + A[i][k] * A[j][i];$

}

**Algorithm 3.2. Modular LU factorization, Mod_Mtx LU (Mod_Mtx x).**

INPUT: *Matrix x; int i, j, k, dim = Dim(x), $Mod_Mtx$ y(x); Pivoting preprocessing not implemented.*

OUTPUT: *L and U.*

COMPUTATIONS:

*for* $(k = 0; k < dim; k++)$

*for* $(i = k + 1; i < dim; i++)$ {

$y(i, k) = -(y(i, k)/y(k, k));$

*for* $(j = k + 1; j < dim; j++)$

$y(i, j) = y(i, j) + y(i, k) * y(k, j);$

}

*return y;*

The latter algorithm is almost identical to the numerical $LU$ algorithm. The only change is the use of the modulus parameters (the $Mod_Mtx$ declarations) as operands in the implementations and, of course, the modulus vector arithmetic operations.

Divisions by zero stop the computations. Divisions by very small values may cause overflow and large errors. A possible consequence is the instability of the floating point output.

In the modular version, instability is a *crisp* phenomenon. The algorithm either generates undefined components, or it does not. If an undefined component is generated in a pivot element, then all results in the right hand square below this pivot will contain the same undefined components.

Let the modular $LU$ algorithm operate on integers in the range $[-2^m, 2^m]$. Then the overall bit complexity of the resulting algorithm in this domain is $\tilde{O}(mn^3)$, that is, $(mn^3)^{1+o(1)}$ for computing the factors $LU$.

Finally, an example is offered of computing the determinant based on $LU$ decomposition in modular arithmetic.

$$\begin{pmatrix} -26085 & -114752 & -24 & 60672 & 5080 \\ 117533 & 145857 & 884 & -171619 & -146386 \\ -75942 & -216371 & -1288 & 177628 & 211880 \\ 235 & 672 & 4 & -551 & -658 \\ 178309 & -258918 & -1532 & 99849 & 249854 \end{pmatrix}$$

The determinant computed from this matrix based on floating point $LU$ factoring and the subsequent multiplication of the diagonal elements of $U$ is

$$75649.28268318021.$$

If the determinant is re-evaluated using $LU$ and diagonal multiplication in the modular vector ring $\mathbf{M}^2 = \{\mathbf{F}_{1009}, \mathbf{F}_{1013}\}$, then the recomputed determinant, recovered from the modular result, is

$$1279.999999999944.$$

The correct value, known in this case, is $\pm 1280$. The determinant is certified to be positive.

## 3.2 Numerical approach to the sign of the determinant

Numerical approach had been studied because it is faster than algebraic/symbolic. The price is extra care of the affect of rounding errors on the output. The care must be taken to insure that the rounding errors do not contaminate the output.

### 3.2.1 Introduction

In [PY01] the authors propose some simple but novel techniques for the certification of the sign of det(A) computed numerically. The presented algorithm either certifies that the sign of det(A) has been computed correctly or shows which increase of the precision of computing should yield the certified output.

If the computed approximation, $d^*$, to det(A) and an available upper estimate, $e_d^+$, for the error satisfy $|d^*| > e_d^+$, then it is certified that $\text{sign}(d^*) = sign(\det(A))$, that is, $\text{sign}(\det(A))$ is computed correctly. Otherwise, the result of the computations is not certified, and one must repeat numerical computations with a higher precision or shift to exact integer or rational computations.

The major technical novelty of the paper is the estimation of the distance to a closest singular matrix, instead of estimating the round-off error of computing $\det(A)$. An estimate for the minimum distance from the matrix $\tilde{L}\tilde{U}$ to a singular matrix, $\tilde{L}$ and $\tilde{U}$ being the computed approximations to the factors $L$ and $U$ of the triangular $(PLUP_1)$ factorization of $A$.

By a well-known theorem [EY39], [K66], this distance is equal to $1/N$, for $N = ||\tilde{U}^{-1}\tilde{L}^{-1}||$.

The idea is that $\det(A+E)$ does not change its sign when $E$ ranges in the ball of radius $N$ centered in the origin.

On the other hand, since the matrices $\tilde{L}$ and $\tilde{U}$ are the two available triangular matrices, it is not hard to obtain a quite tight upper bound on $N$ at the cost of performing $O(n^3)$ arithmetic operations and comparisons.

### 3.2.2 Roundoff errors of matrix factorization by Gaussian elimination

As had been said earlier, $\det(A)$ and its sign for a given matrix $A$ can be immediately obtained from the triangular $(PLUP_1)$ factorization of $A$, but the problem is to analyze the effect of the roundoff errors when the factorization is computed

numerically.

In this analysis some known error estimates had been applied, which will be recalled in this section.

**Theorem 3.1.** (Cf. [H02, Theorem 9.3, page 175].) Let

$$\begin{aligned} A &= PA'P_1, \\ \tilde{A} &= A' + E = \tilde{L}\tilde{U}, \end{aligned} \tag{3.1}$$

where $A$, $A'$, $\tilde{A}$, $E$, $P$, $P_1$, $\tilde{L}$ and $\tilde{U}$ are $n \times n$ matrices, $P$ and $P_1$ are permutation matrices, $\tilde{L} = (\tilde{l}_{i,j})$ is a unit lower triangular matrix (so that $\det(\tilde{L} = 1)$, and $\tilde{U} = (\tilde{u}_{i,j})$ is an upper triangular matrix, $\tilde{L}$ and $\tilde{U}$ are computed numerically, by means of Gaussian elimination (with complete pivoting) applied to the matrix $A$ with unit roundoff $\epsilon$ (machine epsilon). Then $|E| \leq \gamma_n |\tilde{L}| \cdot |\tilde{U}|$, $\gamma_n = n_\epsilon/(1 - n_\epsilon)$, $n_\epsilon < 1$, and close in magnitude to $\epsilon$.

Two special cases of this result cover Gaussian elimination with partial pivoting (for $P_1 = I$) and with no pivoting (for $P_1 = P = I$).

By (3.1)

$$\det(A) = (\det(P))(\det(A'))(\det(P_1)), \tag{3.2}$$

$$\det(\tilde{A}) = \det(\tilde{U}) = \tilde{u}_{1,1}\tilde{u}_{2,2}\ldots\tilde{u}_{n,n}. \tag{3.3}$$

since $\det(P)$ and $\det(P_1)$ are readily available (they equal 1 or $-1$), the equations (3.2) and (3.3) define sign(det(A)) provided the diagonal entries of $\tilde{U}$ are available and $\epsilon$ is sufficiently small to guarantee that

$$sign(\det(A')) = sign(\det(\tilde{A})). \tag{3.4}$$

In the next section it will be shown how to verify (3.4) by using the following corollary of theorem 3.1.

**Corollary 3.1.** *Under the assumptions of theorem 3.1,*

$$||E|| \le e = ||(|\tilde{L}| \cdot |\tilde{U}|)||\gamma_n \tag{3.5}$$

*for any fixed operator matrix norm.*

The computation of the row and column norms of $|\tilde{L}| \cdot |\tilde{U}|$ involves only $O(n^2)$ arithmetic operations because $||(|\tilde{L}| \cdot |\tilde{U}|)||_\infty = ||(|\tilde{L}|(u_i)||_\infty$, $||(|\tilde{L}| \cdot |\tilde{U}|)||_1 = ||({l_j}^T|\tilde{U}|)||_1$, where $(l_j)$ and $(u_i)$ are two vectors with the components $l_j = \Sigma_i|\tilde{l}_{i,j}|$ and $u_i = \Sigma_j|\tilde{u}_{i,j}|$.

In virtue of corollary 3.1 the rounding error of the computation of the $PLUP_1$ factorization of $A$ is bounded in terms of $\gamma_n$ and the norm of the matrix $|L| \cdot |U|$. It is known that in the case of using complete pivoting, the $(k,j)$th entries $\tilde{a}_{k,j}$, $\tilde{l}_{k,j}$ and $\tilde{u}_{k,j}$ of the matrices $\tilde{A}$, $\tilde{L}$ and $\tilde{U}$ of (3.1), respectively, satisfy the bounds

$$|\tilde{l}_{k,j}| \le 1,\ |\tilde{u}_{k,j}| \le \rho_k \max_{(g,h)} |\tilde{a}_{g,h}|,$$

for all $k$ and $j$, where $\rho_k \le k^{1/2}(2 \cdot 3^{1/2} \dots k^{1/(k-1)})^{1/2} \le 1.8k^{(lnk)/4}$ (cf. [GV96, p.119]). The same bounds have been observed in practice in the case of partial pivoting, but formally proved only for $\rho_k \le 2^{k-1}$ (cf. [GV96, p.116]). Some improvement of the worst case error bound (in both cases, of partial and complete pivoting) can be obtained by means of symmetrization.

### 3.2.3 Certification of the sign in terms of the smallest distance to a singular matrix

The following sufficient condition could be verified for (3.4),

$$|\det(\tilde{U})| = |\det(\tilde{A})| > |(\det(A)) - \det(P\tilde{A}P_1)| = e_d,$$

by applying the straightforward crude estimate: $e_d \le n^2 e_+ D_+$, where $e_+$ denotes the maximum absolute value of the entries of $E$, and $D_+ = \prod_{k=1}^{n}(||A_k||_2 + ne_+)$.

This estimate is based on Hadamard bound,

$$|\det(A)| \le \prod_{k=1}^{n} ||A_k||_2 \tag{3.6}$$

(cf. [H02, p.287]). It is not easy to compute $e_+$, but $e_+$ may be replaced by its upper bound $e^* = \gamma_n \max_{i,j}(|\tilde{L}| \cdot |\tilde{U}|)_{i,j}$ implied by theorem 3.1. Here $(|\tilde{L}| \cdot |\tilde{U}|)_{i,j}$ denotes the $(i,j)$th entry of the matrix $(|\tilde{L}| \cdot |\tilde{U}|)$. Then the following estimate will be obtained:

$$e_d \le e_d^+ = D^+ n^2 e^*, \ D^+ = \prod_{k=1}^{n} (||A_k||_2 + ne^*). \tag{3.7}$$

More refined techniques for the verification of equation (3.4) rely on the next two results.

**Proposition 3.1.** *For two given matrices $W$ and $W + \triangle$, for a fixed matrix norm $||\cdot||$ and for all singular matrices $S$, let*

$$\max\{||W - S||, ||W + \triangle - S||\} > ||\triangle||. \tag{3.8}$$

*Then*

$$\operatorname{sign}(\det(W)) = \operatorname{sign}(\det(W + \triangle)). \tag{3.9}$$

*Proof.* Unless (3.9) holds, $S = W + t\triangle$ is a singular matrix for some real $t$, $0 \le t \le 1$. Clearly,

$$\begin{aligned} ||W - S|| = t||\triangle|| &\le ||\triangle||, \\ ||W + \triangle - S|| = (1-t)||\triangle|| &\le ||\triangle||, \end{aligned}$$

and (3.8) is violated. □

The next theorem is due to Eckart and Young, 1939, [EY39], in the case of the norm $||\cdot|| = ||\cdot||_2$ and to Gastinel, 1966, [K66], for the general norm $||\cdot||$.

**Theorem 3.2.** *For any fixed nonsingular matrix $W$ and any fixed operator matrix norm $||\cdot||$, the following is true*

$$\frac{1}{\min ||W - S||} = ||W^{-1}||,$$

*where the minimum is over all singular matrices S.*

Combining proposition 3.1 and theorem 3.2 implies the next result.

**Corollary 3.2.** *Under the assumptions of theorem 3.2,*

$$if \quad \frac{1}{\min\{||W^{-1}||, ||(W+\triangle)^{-1}||\}} > ||\triangle||, \quad then \quad (3.9) \quad holds.$$

Apply corollary 3.2 to $W = A'$ and $\triangle = \tilde{A} - A'$ to obtain the next result.

**Corollary 3.3.** *The equation (3.4) holds if* $||E|| < 1/\min\{||(A')^{-1}||, ||\tilde{A}^{-1}||\}$.

Now an algorithm will be described for computing sign(det(A)). Its stage 2 relies on the simple bound (3.7), which suffices for a large class of inputs, and its stage 3 relies on corollary 3.3.

**Algorithm 3.3.**

INPUT: *an $n \times n$ matrix A, a fixed matrix norm ($||\cdot||_\infty$ or $||\cdot||_2$), and a unit roundoff $\epsilon$.*

OUTPUT: *either the certified value of sign(*$\det$*(A)) or FAILURE.*

COMPUTATIONS:

1. *Apply Gaussian elimination (with complete, partial, or no pivoting) using the unit roundoff $\epsilon$, to compute the matrices $\tilde{L}$ and $\tilde{U}$ of (3.1).*

2. *Compute an upper bound on $e_d$ (in particular, the simple crude bound $e_d^+$ of (3.7)), may be used, and check if $|\det(\tilde{U})|$ exceeds this bound. If so, compute and output sign(*$\det$*(A)) based on (3.2)-(3.4).*

3. *Otherwise, estimate the norm $N = ||\tilde{A}^{-1}|| = ||\tilde{U}^{-1}\tilde{L}^{-1}||$ from above and/or below to decide whether*

$$eN < 1. \tag{3.10}$$

*If the latter inequality is verified, compute and output sign(*$\det(A)$*) based on (3.2)-(3.4). Otherwise output FAILURE.*

The complexity of the computations by the algorithm is dominated by the complexity of its stages 1 and 3. The reader could be referred to [GV96] and [BP94] on the complexity of stage 1 and to section 5 on the complexity of stage 3. In both cases $O(n^3)$ arithmetic operations is needed. At stage 1, $O(n^3)$ or $O(n^2)$ comparisons for complete or partial pivoting, respectively may also be needed.

### 3.2.4 Some recipes in the case of FAILURE

Suppose that algorithm 3.3 has output FAILURE and that an upper bound $N^+$ on $N$ and the value $f = eN^+$ are available. Then several options could arise:

1. Repeat the computation but with the unit roundoff $\epsilon_{new} = c\epsilon_{old}/f$ for some heuristic choice of $c < 1$. The value $c$ can be adapted dynamically, depending on the resulting change of the value $eN^+$. If the latter value changes proportionally to $\epsilon_{new}$ (as can be expected unless $A$ is a very ill-conditioned matrix), then (3.10) holds for $\epsilon = \epsilon_{new} = c\epsilon_{old}/f$ and for any $c < 1$. Recomputation of $\tilde{L}$ and $\tilde{U}$ for $\epsilon_{new}$ can be simplified because several leading digits in the representation of the computed values stay invariant, and only the remaining trailing digits must be recomputed (compare [EPY98]).

2. Unless it is known already that $N^{-1} \geq 1/e$, try to improve the upper estimate $N^+$ for $N = ||\tilde{U}^{-1}\tilde{L}^{-1}||$ at stage 3.

3. If the value $f$ is too large so that numerical computations with a unit roundoff $\epsilon_{new} < \epsilon_{old}/f$ become too expensive, shift to the symbolic algorithms of [BEPP99]. In this case, one needs an a priori upper bound on $|\det(A)|$. Hadamard's inequality, $|\det(A)| \leq \prod_{k=1}^{n} ||A_k||_2$, or the bound

$|\det(\mathrm{A})| \leq e_d^+ + |\det(\tilde{U})|$ can be used, but it may pay to refine these bounds by performing some additional computations.

# 4 Further work

## 4.1 Preliminaries

First recall the notion of *certifying* the results of floating point computation [PY01]. The meaning of this is that the error, generated in the calculation, is estimated. This estimate becomes a part of the algorithm. In most cases the data needed to perform the computation is already available in the standard numerical algorithms. The error estimate becomes just an additional output.

For example, suppose $\det(A)$ is computed using an efficient floating point algorithm. Then if $e_d$ is also computed and output, this is a position to *certify the correctness* of the sign of the determinant by comparing $|\det(A)|$ with $e_d$. Then, as seen above, if $e_d < |\det(A)|$, then the *sign* of the computed $\det(A)$ is certified. Otherwise the computation is not certified to be correct, and exact arithmetic method will be used just for this case.

To extend the approach of [PY01], modify the input matrix $A$ keeping the sign of its determinant invariant but decreasing its condition number by means of its *multiplicative & additive preconditioning* proposed in [PIMRTYa].

*Multiplicative preconditioning* is a popular technique for solving linear systems of equations $A\mathbf{x} = \mathbf{b}$. The original idea was to shift to equivalent but better conditioned linear systems $BA\mathbf{x} = B\mathbf{b}$, $AC\mathbf{y} = \mathbf{b}$, or more generally $BAC\mathbf{y} = B\mathbf{b}$ for $\mathbf{x} = C\mathbf{y}$, so that a more accurate numerical solution $\mathbf{x}$ can be computed faster (see [C05] and the bibliography therein). The present study stems from this idea, although the compression of the singular spectrum of the matrix $A$ has become an even more popular means towards the same goal. The latter direction is not relevant to the task of determinant computation.

*Additive preconditioning* is adding a matrix of a bounded rank to the input matrix $A$ to decrease its condition number. It is a complement or counterpart to

multiplicative preconditioning.

Hereafter the abbreviations of *M-preconditioning* and *A-preconditioning* will be used for multiplicative and additive preconditioning and *MPs* and *APs* for multiplicative and additive preconditioners, respectively.

By combining *A-preconditioning* and *M-preconditioning* with computing the $SVD$ of the matrix $A$, one can decrease the condition number to its minimum value one, but according to some extensive tests a less radical decrease, achieved with *A-preconditioning* at a lower computational cost, is usually sufficient even for the harder inputs, where the determinant nearly vanishes.

It is planned to elaborate upon the details of this approach and to test it numerically.

In the next section some definitions and preliminary results for the study of *A-preconditioning* and *M-preconditioning* will be introduced.

## 4.2 Additive preconditioning: why and how?

Multiplicative preconditioners are closely linked to the Singular Value Decomposition (SVD) of the input matrix. This is somewhat restrictive because the computation of the smallest singular values and the associated singular vectors of an ill conditioned matrix is costly.

As an alternative or complementary tool, in [P04] *additive preprocessing* or *additive modification* $A \leftarrow C = A+UV^T$ was proposed, i.e., a matrix $UV^T$ (having a smaller rank and/or structured) is added to the input matrix $A$ to decrease its condition number. Here and hereafter $M^T$ denotes the Hermitian (that is, complex conjugate) transpose of a matrix $M$, which is just its transpose $M^T$ where $M$ is a real matrix. Hereafter $I_k$ will express the $k \times k$ identity matrix, $M^{-H}$ for $(M^T)^{-1} = (M^{-1})^T$, $Q(M)$ for the Q-factor in the QR factorization of a matrix $M$ of full rank, $\sigma_j(A)$ for the $j$th largest singular value of a matrix $A$, $\rho = \operatorname{rank} A$ for its

rank, and $\text{cond}_2 A = \sigma_1(A)/\sigma_\rho(A)$ for its condition number, and the abbreviations of *MPPs*, *APPs*, *MPCs*, *APCs*, *A-modification*, and *M-* and *A-preconditioning* will be used for multiplicative and additive preprocessors and preconditioners, additive modification, and multiplicative and additive preconditioning, respectively.

We rely on *additive preconditioning* $A \leftarrow C = A + UV^T$, i.e., add a matrix $UV^T$ (which typically has a smaller rank) to the input matrix $A$ to decrease its condition number, and use the abbreviations of *APPs*, *APCs*, *A-modification*, and *A-preconditioning* for additive preprocessors and preconditioners, additive modification, and additive preconditioning, respectively. (An APC is an APP that decreases the condition number.)

Given a nonsingular $n \times n$ matrix $A$, a positive integer $r < n$, and the $r$-tail (resp. $r$-head) of its SVD, that is the $r$ smallest (resp. largest) singular values of the matrix $A$ together with the associated singular spaces, one can immediately define an APC $UV^T$ of a rank $r$ and the A-modification $C = A + UV^T$ such that $\text{cond}\, C = \sigma_1(A)/\sigma_{n-r}(A)$ (resp. $\text{cond}\, C = \sigma_{r+1}(A)/\sigma_n(A)$). If both $r$-head and $r$-tail are known and if $2r < n$, one can readily obtain the optimal APCs $UV^T$ of a rank $r < n/2$, such that $\text{cond}\, C = \sigma_{r+1}(A)/\sigma_{n-r}(A)$ [W]. This can help even if just $r$-tail and/or $r$-head had been approximated because an APC tends to remain an APC in its small-norm perturbation.

One can obtain the $r$-head and $r$-tail of the SVD by applying the Lanczos algorithm [GV96, Chapter 9], [S98b, Chapter 5], but according to [PIMRTYb] it is possible to rely even on the less costly choice of an APP $UV^T$ of a rank $r$ which is

a) random (general, sparse, or structured [PIMRTYa]),

b) well conditioned, and

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

small.

Then according to the analysis and extensive experiments in [PIMRTYb], it is likely to arrive at an A-modification $C = A + UV^T$ with $\operatorname{cond} C$ of the order of $\sigma_1(A)/\sigma_{n-r}(A)$.

For computing APPs with properties b) and c) as well as at the other stages of computing and employing A-preconditioning, it is possible to apply the effective norm and condition estimators in [GV96, Section 3.5.4] and [S98a, Section 5.3]. With some additional SVD-free techniques in [PIMRTYa], [PIMRTYb] it is possible to

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

# 5 SVD-based preconditioning

## 5.1 SVD-based MPPs

To solve a matrix equation $AY = B$, one can employ factorizations $F = MAN$ or equivalently $MA = FN^{-1}$ for two nonsingular matrices $M$ and $N$. One should seek the matrices $M$, $N$ and/or $N^{-1}$ for which the solution is simplified. In particular one can seek factorization $F = MAN$ such that the matrix $F$

a) becomes a full rank matrix after the deletion of its zero rows and/or columns and/or

b) is better conditioned than the matrix $A$.

The MPP $\{M, N\}$ is an MC in case a) and/or MPC in case b).

Given the SVD $A = S\Sigma T^T$, it is possible to immediately obtain the MCs $M = S^T$, $N = T$, and $(M, N) = (S^T, T)$ in the case of a rank deficient matrix $A$ and the MPCs $M = \Sigma^{-1}S^T$, $N = T\Sigma^{-1}$, $(M, N) = (\Sigma^{-1}S^T, T)$, and $(M, N) = (S^T, T\Sigma^{-1})$ in the case of an ill conditioned matrix $A$. The next two theorems, implied by equations (1.7)–(1.9), show MCs and MPCs where the $g$-head and/or $h$-tail of the SVD are only known.

**Theorem 5.1.** *[PIMRTYb]. Let $A = S\Sigma T^T$ be the SVD of an $m \times n$ matrix $A$ of a rank $\rho$ and let $\tilde{S}_{0,0}$ and $\tilde{T}_{0,0}$ be two matrices in equations (1.6) for $g = h = 0$. Then the deletion of the zero rows and columns of the matrices $\tilde{S}_{0,0}^T A$, $A\tilde{T}_{0,0}$, and $\tilde{S}_{0,0}^T A\tilde{T}_{0,0}$ turns them into full rank matrices of sizes $\rho \times n$, $m \times \rho$, and $\rho \times \rho$, respectively, so that the matrices $M = \tilde{S}_{0,0}^T$ and $N = \tilde{T}_{0,0}$ as well as the matrix pair $(M, N) = (\tilde{S}_{0,0}^T, \tilde{T}_{0,0})$ are MCs for a rank deficient matrix $A$.*

**Theorem 5.2.** *[PIMRTYb]. Keep the assumptions of Theorem 5.1 but use the matrices $\tilde{S}_{g,h}$ and $\tilde{T}_{g,h}$ in equations (1.6) for any pair of nonnegative integers $g$*

*and* $h \le \rho - g$. *Let* $\sigma$ *lie in the closed interval* $[\sigma_{\rho-h}, \sigma_{g+1}]$. *Write* $d_j = \sigma/\sigma_j$ *for* $j = 1, \ldots, g, \rho - h + 1, \ldots, \rho$,

$$\begin{aligned} D^{(g)} &= \mathrm{diag}(d_j)_{j=1}^{g}, \\ D_h &= \mathrm{diag}(d_j)_{j=\rho-h+1}^{\rho}, \\ D_{g,h,k}(\sigma) &= \mathrm{diag}(D^{(g)}, I_{\rho-g-h}, D_h, I_k) \end{aligned}$$

*for* $k = l$ *and* $k = r$,

$$\begin{aligned} F_l &= D_{g,h,l}(\sigma)\tilde{S}_{g,h}^T A, \\ F_r' &= A\tilde{T}_{g,h} D_{g,h,r}(\sigma), \\ F_{l,r} &= D_{g,h,l}(\sigma)\tilde{S}_{g,h}^T A\tilde{T}_{g,h}, \\ F_{l,r}' &= \tilde{S}_{g,h}^T A\tilde{T}_{g,h} D_{g,h,r}(\sigma). \end{aligned}$$

*Then* $\mathrm{cond}_2 F_l = \mathrm{cond}_2 F_r' = \mathrm{cond}_2 F_{l,r} = \mathrm{cond}_2 F_{l,r}' = \sigma_{g+1}/\sigma_{\rho-h}$, *so that for a* $(g, h)$ *matrix* $A$, *the matrix pairs*

$$\begin{gathered} (D_{g,h,l}(\sigma)\tilde{S}_{g,h}^T,\ I_n), \\ (I_m,\ \tilde{T}_{g,h} D_{g,h,r}(\sigma)), \\ (D_{g,h,l}(\sigma)\tilde{S}_{g,h}^T,\ \tilde{T}_{g,h}),\quad \textit{and} \\ (\tilde{S}_{g,h}^T,\ \tilde{T}_{g,h} D_{g,h,r}(\sigma)) \end{gathered}$$

*are its MPCs.*

One can extend factorizations (1.7)–(1.9) and Theorem 5.2 by replacing the unitary matrices above with their well conditioned approximations.

## 5.2 Null-based ACs and SVD-based APCs

It is possible to turn a singular (resp. rank deficient) matrix into a nonsingular (resp. full rank) matrix by adding matrix $UV^T$ where $U$ and $V$ are null matrix bases.

**Algorithm 5.1. Computing a null-based AC.**

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

OUTPUT: *two matrices $U$ and $V$ and the matrix*

$$C = A + UV^T = \sum_{j=1}^{\rho} \sigma_j \mathbf{s}_j \mathbf{t}_j^T + \sigma \sum_{j=\rho+1}^{u} \mathbf{s}_j \mathbf{t}_j^T \tag{5.1}$$

*of full rank $u = \min\{m, n\}$.*

COMPUTATIONS: *Fix a positive $\sigma$ and compute and output a pair of matrices*

$$U = (\sigma \mathbf{s}_j)_j, \ V = (\mathbf{t}_j)_j \ \text{ for } \ j = \rho + 1, \dots, u. \tag{5.2}$$

If $\sigma_\rho \le \sigma \le \sigma_1$ in equation (5.1), then it is possible to have $\mathrm{cond}_2 C = \sigma_1/\sigma_\rho$. Unless $\sigma_1 >> \sigma_\rho$, the matrix $C$ is well conditioned, and then so is the matrix $C + E$ if $||C||_2 >> ||E||_2$. It is possible to extend Algorithm 5.1 to SVD-based computation of APCs for $(0, h)$ matrices because they lie near singular matrices.

**Algorithm 5.2.** *[PMQRT].* **Computing an SVD-based APC.**

INPUT: *an $m \times n$ matrix $A$ of a rank $\rho$, two nonnegative integers $g$ and $h$ such that $g + h \le \rho$, the $g$-head and the $h$-tail of the SVD $A = S\Sigma T^T$, and a positive $\sigma$ in the range $[\sigma_{g+1}, \sigma_{\rho-h}]$.*

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

$$\mathrm{cond}_2(A + UV^T) = \max\{\sigma, \sigma_{g+1}\} / \min\{\sigma, \sigma_{\rho-h}\}. \tag{5.3}$$

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

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

$$\begin{aligned} UV^T &= \sum_{j=1}^{g}(\sigma - \sigma_j)\mathbf{s}_j\mathbf{t}_j^T + \sum_{j=\rho-h+1}^{\rho}(\sigma - \sigma_j)\mathbf{s}_j\mathbf{t}_j^T, \\ U &= ((\sigma - \sigma_j)\mathbf{s}_j)_j, \\ V &= (\mathbf{t}_j)_j \end{aligned} \tag{5.4}$$

*for $j = 1, \ldots, g;\ \rho - h + 1, \ldots, \rho$.*

To verify correctness of the algorithm, observe that

$$C = A + UV^T = \sum_{j=g+1}^{\rho-h}\sigma_j\mathbf{s}_j\mathbf{t}_j^T + \sigma\widehat{\sum}_j\mathbf{s}_j\mathbf{t}_j^T \tag{5.5}$$

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

**Remark 5.1.** *Given the $g$-head and the extended $h$-tail of a $(g, h)$ matrix $A$, it is possible to combine Algorithms 5.1 and 5.2 to arrive at an APP $UV^T$ such that the matrix $A + UV^T$ has full rank and satisfies equation* (5.3).

# 6 SVD-free preconditioning

This section relies on the results in [PIMRTYb]. They support the use of A-preconditioning for computing determinants.

## 6.1 SVD-free ACs and APCs

### 6.1.1 SVD-free APPs versus SVD-based APPs and MPPs

Two problems with the SVD-based preconditioning could happen.

a) Realistically, approximations to the $g$-heads of the SVDs of an input matrix $A$ for smaller positive $g$ are readily available, but the task is more costly for the $h$-tails of the SVDs for positive $h$ [GV96, Sections 9.1 and 9.2], [S98b, pages 366 and 367], [BDDRV00], [LZ05].

b) The SVD does not preserve matrix structure.

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

### 6.1.2 Error-free A-preconditioning

Given a $(g, 0)$ matrix $A$ and its SVD-based or SVD-free APC $UV^T$, the rounding errors of computing the A-modification $C = A + UV^T$ are of the order of $\epsilon||A||_2 = \epsilon\sigma_1(A)$ where $\epsilon$ is the unit round-off. They are large relatively to the output norm $\tilde{\sigma} = ||A + UV^T||_2 = \max\{\sigma, \sigma_{g+1}\}$ if $\sigma_1 >> \tilde{\sigma}$, and so they can ruin positive effect of preconditioning.

We, however, avoid the problem by filling the generators $U$ and $V$ with shorter numbers to yield an error-free A-modification $C$ in double precision computations [DH03] and [PS91]. Small-norm perturbations of the generators caused by the truncation of their entries keep the matrix $C$ well conditioned.

### 6.1.3 Randomized ACs

We do not use the results of this section explicitly, but they have been included to provide a more complete insight into A-preconditioning.

The next theorem [Pa] links the ranks of a random APP $UV^T$ and of the A-modification $C = A + UV^T$ of an $m \times n$ matrix $A$. First we sketch the main claims of this theorem by writing $u = \min\{m, n\}$ and using "$\Longrightarrow$" for "implies" and then state and prove it formally.

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

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

**Theorem 6.1.** *For a finite set $\Delta$ of cardinality $|\Delta|$ in a ring $\mathbb{R}$, $u = \min\{m, n\}$, and four matrices $A \in \mathbb{R}^{m \times n}$ of a rank $\rho$, $U \in \Delta^{m \times r}$, $V^T \in \Delta^{r \times n}$, and $C = A + UV^T$, the following is true*

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

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

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

*Proof.* Part a) is verified immediately. Now let $r + \rho \geq u$, let a $u \times u$ submatrix $A_u$ of the matrix $A$ have rank $\rho$, and let $C_u = A_u + (UV^T)_u$ denote the respective $u \times u$ submatrix of the matrix $C$. Then clearly, $\operatorname{rank} C = \operatorname{rank} C_u = u$ if $U = V$ and if the entries of the matrix $U$ are indeterminate. Since $\det(C_u)$ is a polynomial

of a total degree of at most $2(u-\rho) \le 2r$ in these entries, part b) follows from Lemma 1.2. Part c) is proved similarly to part b). □

The following algorithm recursively generates random APPs $UV^T$ of ranks $r = 0, 1, \ldots$ and stops where either it arrives at an AC or a fixed upper bound $r(+)$ on $r$ is exceeded. Foreseeing an extension to numerical computation of APCs, it is possible to choose random unitary matrices $U$ and $V$.

**Algorithm 6.1.** *[PIMRTYb].* **Randomized computation of a unitary AC.**

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

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

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

COMPUTATIONS:

1. *If $r$ exceeds $r(+)$, stop and output FAILURE.*
2. *Otherwise apply Subroutine FULL·RANK to test if the $A$-modification $C = A + UV^T$ has full rank. If so, output the integer $r$ and the matrices $U$ and $V$ and stop.*
3. *Otherwise sample two normalized random vectors $\mathbf{u}$ and $\mathbf{v}$ of dimension $n$ such that $U^T\mathbf{u} = \mathbf{0}$ and $V^T\mathbf{v} = \mathbf{0}$ (unless $r = 0$), set $r \leftarrow r + 1$, $U \leftarrow (U, \mathbf{u})$, and $V \leftarrow (V, \mathbf{v})$, and go to Stage 1.*

In virtue of Theorem 6.1, Algorithm 6.1 is correct and is likely to output $r = u - \rho = \text{nul}\, A$.

The most costly stage of the algorithm is the expected $u - \rho$ applications of the Subroutine FULL·RANK to the matrices $C$, but $2\lceil \log_2(u-\rho) \rceil$ of them, are needed at most, if the binary search for the nullity is incorporated.

### 6.1.4 Extension to computing APCs and their consistent scaling

To implement Algorithm 6.1 numerically, just replace Subroutine FULL·RANK with estimating whether $\text{cond}_2 C$ exceeds a fixed tolerance bound. Applied to $(0, h)$ matrix $A$, the resulting algorithm either fails (in some pathological cases) or outputs a well conditioned A-modification $C = A + UV^T$.

Properly scaled and well conditioned, rather than unitary, random APPs can be more readily consistent with a desired matrix structure and are still likely to be APCs according to the analysis and the test results in Sections 6.2 and [P04, section 6].

A specify for the proper scaling. An APP $UV^T$ is *scaled consistently* with a $(0, h)$ matrix $A$, and the pair of $A$ and $UV^T$ is scaled consistently, if both the ratio $||A||_2/||UV^T||_2$ and its reciprocal are bounded by a moderate constant, that is, if this ratio is neither large nor small. It is possible to scale APPs by the powers of two to avoid rounding errors.

## 6.2 APPs and conditioning

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

First consider a normalized singular well conditioned matrix $A$ with nullity $r$ and a random unitary APP $UV^T$ of rank $r$ and show that the A-modification $C = A + UV^T$ is expected to be nonsingular and well conditioned. Then extend

this result to a consistently scaled pair of a $(0, r)$ matrix $A$ and a random and well conditioned APP $UV^T$ of rank $r$.

### 6.2.1 ACs and conditioning

First factorize the A-modification $C$.

**Theorem 6.2.** *[PIMRTYb]. Let $A$ be an $n \times n$ matrix of rank $\rho = n - r$, so that $r = \operatorname{nul} A$. Let $U$ and $V$ be $n \times r$ matrices such that the $n \times n$ matrix $C = A + UV^T$ is nonsingular. Let $A = S\Sigma T^T$ be the SVD of the matrix $A$, where the matrices $S^T$ and $T^T$ are unitary, $\Sigma = \operatorname{diag}(\Sigma_A, 0_r)$, and $\Sigma_A = \operatorname{diag}(\sigma_j)_{j=1}^{\rho}$ is the diagonal matrix of the singular values of the matrix $A$. Write*

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

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

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

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

*Proof.* Write $\tilde{C} = \Sigma + S^T U V^T T$. Observe that

$$C = A + UV^T = S\Sigma T^T + SS^T UV^T TT^T = S\tilde{C}T^T,$$

$$R_U \Sigma R_V^T = \Sigma,$$

$$S^T U = R_U \begin{pmatrix} 0 \\ I_r \end{pmatrix}, \quad \text{and}$$

$$T^T V = R_V \begin{pmatrix} 0 \\ I_r \end{pmatrix}.$$

Deduce that $\tilde{C} = R_U \sum R_V^T + R_U \operatorname{diag}(0, I_r) R_V^T = R_U \operatorname{diag}(\sum_A, I_r) R_V^T$. Substitute this expression into the equation $C = S\tilde{C}T^T$ to arrive at part a). Part b) follows because the matrix $C$ is nonsingular. □

**Corollary 6.1.** *[PIMRTYb]. Under the assumptions of Theorem 6.2 the following is true*

$$\frac{||\operatorname{diag}(\sum_A, I_r)||_2}{||R_U^{-1}||_2||R_V^{-1}||_2} \le ||C||_2 \le ||\operatorname{diag}(\sum_A, I_r)||_2||R_U||_2||R_V||_2,$$

$$\frac{||\operatorname{diag}(\sum_A^{-1}, I_r)||_2}{||R_U||_2||R_V||_2} \le ||C^{-1}||_2 \le ||\operatorname{diag}(\sum_A^{-1}, I_r)||_2||R_U^{-1}||_2||R_V^{-1}||_2,$$

*so that*

$$\frac{\operatorname{cond}_2 \operatorname{diag}(\sum_A, I_r)}{(\operatorname{cond}_2 R_U)\operatorname{cond}_2 R_V} \le \operatorname{cond}_2 C \le (\operatorname{cond}_2 R_U)(\operatorname{cond}_2 R_V)\operatorname{cond}_2 \operatorname{diag}(\sum_A, I_r).$$

*Proof.* The corollary follows from Theorem 6.2 because $||S||_2 = ||S^{-1}||_2 = ||T||_2 = ||T^{-1}||_2 = 1$, $\operatorname{cond}_2 M = ||M||_2||M^-||_2$ and $||M^T||_2 = ||M||_2$ for any matrix $M$. □

A specify for the estimate for $\operatorname{cond}_2 C$ provided the matrices $U$ and $V$ are unitary and the matrix $A$ is scaled properly.

**Theorem 6.3.** *[PIMRTYb]. If the matrices $U$ and $V$ are unitary, then the following is true*

$$||R_U||_2 \le \sqrt{2}, \qquad ||R_V||_2 \le \sqrt{2},$$

$$||R_U^{-1}||_2 \le 1 + \sqrt{2}||U_r^{-1}||_2, \qquad ||R_V^{-H}||_2 = ||R_V^{-1}||_2 \le 1 + \sqrt{2}||V_r^{-1}||_2.$$

*Proof.* The first two bounds of the theorem follow because $R_U = (I_{n,\rho}, S^T U)$, $R_V = (I_{n,\rho}, T^T V)$ where $I_{n,\rho} = \begin{pmatrix} I_\rho \\ 0 \end{pmatrix}$ and because $||(X,Y)||_2 = ||(X,Y)^T||_2 \le \sqrt{2}$ for a pair of matrices $X$ and $Y$ with 2-norms of at most one.

To deduce the two other bounds, observe that

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

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

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

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

the 2-norms of the matrices $S$, $T$, $U$, and $V$ are equal to one, $||U_\rho||_2 \le ||U||_2 = 1$ and $||V_\rho||_2 \le ||V||_2 = 1$. □

**Theorem 6.4.** *[PIMRTYb]. Under the assumptions of Theorem 6.2, suppose that*

$$\sigma_{n-r} \le 1 \le \sigma_1. \tag{6.1}$$

*Then* $||\operatorname{diag}(\Sigma_A, I_r)||_2 = ||A||_2$ *and* $||(\operatorname{diag}(\Sigma_A, I_r))^{-1}||_2 = \sigma_{n-r}$.

*Proof.* Immediate by inspection. □

**Corollary 6.2.** *[PIMRTYb]. Write*

$$p_r \;=\; ||R_U^{-1}||_2||R_V^{-1}||_2 \;\le\; (1+\sqrt{2}||U_r^{-1}||_2)(1+\sqrt{2}||V_r^{-1}||_2).$$

*Under bounds (6.1) and the assumptions of Theorem 6.2, suppose the matrices $U$ and $V$ are unitary. Then*

*a)* $||A||_2/p_r \le ||C||_2 \le 1 + ||A||_2 \le 2||A||_2$,

*b)* $1/(2\sigma_{n-r}) \le ||C^{-1}||_2 \le p_r/\sigma_{n-r}$, *and*

*c)* $(\operatorname{cond}_2 A)/(2p_r) \le \operatorname{cond}_2 C \le 2p_r \operatorname{cond}_2 A$.

*Proof.* Part a) follows immediately from part a) of Theorem 6.2 because $||C||_2 \le ||A||_2 + ||UV^T||_2$ and $||UV^T||_2 = 1 \le \sigma_1 = ||A||_2$.

To prove part b), combine Corollary 6.1 with Theorems 6.3 and 6.4.

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

### 6.2.2 Small-norm perturbation of an AC and conditioning

Any ill conditioned matrix $\tilde{A}$ is a small-norm perturbations of a singular matrix $A = \tilde{A} - E$. To extend Corollary 6.2 to ill conditioned matrices $\tilde{A}$, it remains to bound the estimates in the corollary in a small-norm perturbation $A \to \tilde{A}$.

**Theorem 6.5.** *[PIMRTYb]. Under the assumptions of Corollary 6.2, let the matrix $\tilde{C} = C + E$ be nonsingular. Write $\delta = ||E||_2$, and $\delta_C = \delta||C^{-1}||_2$. Then the following is true:*

*a)* $||\tilde{C}||_2 \le \delta + ||C||_2$,

*b) if* $\delta_C < 1$, *then* $||\tilde{C}^{-1}||_2 \le ||C^{-1}||_2(1 + (1 - \delta_C)^{-1}\delta_C)$, *so that*

$$\operatorname{cond}_2 \tilde{C} \le (1 + (1 - \delta_C)^{-1}\delta_C)(1 + \delta/||C||_2) \operatorname{cond}_2 C, \quad \textit{and}$$

*c) if the matrices $C$ and $E$ are Hermitian and nonnegative definite, then*

$$||\tilde{C}^{-1}||_2 \le ||C^{-1}||_2, \ \textit{so that} \ \operatorname{cond}_2 \tilde{C} \le (1 + \delta/||C||_2) \operatorname{cond}_2 C.$$

*Proof.* Part a) is proved immediately, similarly to part a) of Corollary 6.2.

To prove part b), apply the SMW formula in Theorem 1.3 to the matrices $C + UV^T = \tilde{C}$, $U = -E$, and $V = I_n$ and obtain that $\tilde{C}^{-1} = C^{-1}(I_n + E(I_n - C^{-1}E)^{-1}C^{-1})$. Part b) follows from this equation and the assumption that $\delta_C = \delta||C^{-1}||_2 < 1$.

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

### 6.2.3 The impact of scaling APPs

Under the assumptions of Corollary 6.2, the ratio $\frac{\operatorname{cond}_2 C}{\operatorname{cond}_2 A}$ is not large unless the product $p_r = ||U_r^{-1}||_2||V_r^{-1}||_2$ is large. Moreover, by relaxing the assumption (6.1), assume that $\sigma_1 = 1/\theta \ge \sigma_{n-r}$ or $\sigma_1 \ge \sigma_{n-r} = \theta$ for $\theta > 1$, and ignore the resulting dynamics in the factors of $||U_r^{-1}||_2$ and $||V_r^{-1}||$, then the upper estimate $2p_r$ on this ratio in Corollary 6.2c would increase by the factor of $\theta$. In some extensive tests, the value of $\theta$ tended to be nicely bounded for random, well conditioned, and consistently scaled APPs.

For random unitary $n \times r$ matrices $U$ and $V$, it is also possible to view the $n \times r$ unitary matrices $S^T U$ and $T^T V$ in Theorem 6.2 as random, so that the norms of the inverses of their $r \times r$ southern-most submatrices $U_r$ and $V_r$ are likely to be reasonably bounded for smaller $r$. (If these norms are large, then $\mathrm{cond}_2 C$ is large, and if this had been detected, it is possible to resample the random matrices $U$ and $V$.)

To define consistent scaling the 2-norms of the matrix $A$ and the APP $UV^T$ must be estimated. The effective algorithms in [S98a, Section 5.3.3] compute quite tight bounds on both norms $||A||_2$ and $||A^{-1}||_2$. Here are some cruder low cost bounds in [GV96, Section 2.3.2 and Corollary 2.3.2],

$$1 \le \frac{||A||_2}{\max_{i,j} |a_{i,j}|} \le \sqrt{mn},$$

$$1 \le \frac{\sqrt{||A||_1 ||A||_\infty}}{||A||_2} \le (mn)^{1/4}.$$

# 7 Solving linear systems of equations with APPs

The results of this section are from [Pa]. A similar simplified approach had been applied to computing determinants.

## 7.1 APCs and aggregation (a brief outline)

Assume a singular or ill conditioned nonsingular $n \times n$ input matrix $A$, an APP $UV^T$ of a rank $r < n$, and a well conditioned nonsingular A-modification $C = A+UV^T$. Then next goal is to reduce the original computational problem with the matrix $A$ (such as the solution of linear systems of equations) to similar problems with some matrices of smaller sizes (it had been called *aggregates*) $I_r - V^T C^{-1} U$. These *aggregation methods* are natural descendants of the ones in [MP80], which in the eighties evolved into the *Algebraic Multigrid*. Unlike [MP80], the aggregation and disaggregation methods in [PIMRTYa] preserve matrix structure and, besides decreasing the size of the input matrix, also improve its conditioning. If needed, apply aggregation recursively until no ill conditioned matrices are involved in the computations. The original numerical problems do not disappear, but it is possible to confine them to the stages of computing certain sums and products and then solve by applying the known effective ASAs.

To solve a linear system $A\mathbf{y} = \mathbf{b}$ it is possible to apply the Sherman–Morrison–Woodbury formula from Theorem 1.3 (also see equation (7.1) in the next subsection). The formula reduces the problem to computing the $r \times r$ *Schur aggregate* $G = I_r - V^T C^{-1} U$ and the matrix $UG^{-1}V^T$ where $r$ is the rank of the APC $UV^T$ and $C = A+UV^T$ is an A-modification. This reduction had been called the *Schur Aggregation*.

The singular values of the Schur aggregate $G$ can be estimated via the values $\sigma_1(C)$, $\sigma_n(C)$, and the $r$ smallest singular values of the matrix $A$. If the

A-modification $C$ is well conditioned, then $\mathrm{cond}_2 G$ has the order of the ratio $\sigma_{n-r+1}(A)/\sigma_n(A)$. For some ill conditioned matrices $A$ this ratio is not large. Then numerical problems are confined to computing the matrix $G$, and counter them by applying iterative refinement and the ASAs.

## 7.2 APCs and aggregation (some technical details)

**The Schur Aggregation** is defined by the *SMW formula*

$$A^{-1} = (C - UV^T)^{-1} = C^{-1} + C^{-1}U(I_r - V^TC^{-1}U)^{-1}V^TC^{-1} \tag{7.1}$$

of Sherman, Morrison, and Woodbury from Theorem 1.3. The formula reduces a linear system $A\mathbf{y} = \mathbf{b}$ to linear systems with the coefficient matrices $C = A + UV^T$ and $G = I_r - V^TC^{-1}U$, the latter matrix being the Schur complement (Gauss transform) of the block $C$ in the block matrix

$$\begin{pmatrix} C & U \\ V^T & I_r \end{pmatrix}.$$

If the A-modification $C$ is well conditioned, then the numerical problems in the inversion of the matrix $A$ are confined to the computations of and with the Schur aggregate $G$.

Theorem 7.3 relates the singular values of the matrices $A$, $C$, and $G$ to each other. Suppose $A$ is an $n \times n$ nonsingular matrix, $U$ and $V$ are $n \times r$ matrices, $n > r$, and all singular values of the A-modification $C = A + UV^T$ lie in a line interval $[c_-, c_+]$ where $c_+ \geq c_- > 0$. Then by Theorem 7.3 it is possible to have $c_-^2\sigma_j(A^{-1}) - c_- \leq \sigma_j(G^{-1}) \leq c_+^2\sigma_j(A^{-1}) + c_+$ for $j = 1, \ldots, r$. It follows that $\mathrm{cond}_2 G = \sigma_1(G)/\sigma_r(G)$ has the order of the ratio $\sigma_{n-r+1}(A)/\sigma_n(A)$ unless the ratio $c_+/c_-$ is large, that is, unless the A-modification $C$ is ill conditioned.

If both ratios $c_+/c_-$ and $\sigma_{n-r+1}(A)/\sigma_n(A)$ are not large, then the matrix $G = I_r - V^TC^{-1}U$ is well conditioned, and so the only remaining numerical problem

is the computation of this matrix. In this case if the matrix $A$ is ill conditioned, then the above bounds imply that the matrix $G$ has a small norm. Consequently, the most significant bits of some diagonal entries are cancelled when subtract the matrix $V^TC^{-1}U$ from the matrix $I_r$. However, it is possible to control the computational errors by applying iterative refinement and the ASAs.

By post-multiplying the SMW formula (7.1) by a vector $\mathbf{y}$ it is possible to reduce a linear system of equations $A\mathbf{y} = \mathbf{b}$ to some systems with the matrices $C$ (the A-modification) and $G = I_r - V^TC^{-1}U$ (the Schur aggregate). We call this reduction the *Schur Aggregation*.

For smaller values $r$ one can readily solve linear systems with the matrix $G$ by applying the algorithms of the CG/GMRES type, even if the matrix is ill conditioned [GV96, Sections 10.2–10.4], [BBCDDDEPRV93]–[V03], but the conditioning of this matrix can become the central issue where $r$ is large.

## 7.3 The norms and conditioning of the Schur complements

Next estimate the $j$th singular values of the matrix $G^{-1}$, $j = 1, \dots, r$, in terms of the singular values $\sigma_j(A^{-1})$, $\sigma_1(C)$, and $\sigma_1(C^{-1})$.

**Theorem 7.1.** *[Pb]. Let $W$ denote an $m \times n$ matrix of full rank $\rho = \min\{m, n\}$. Write $\sigma_+(W) = \sigma_1(W)$, $\sigma_-(W) = \sigma_\rho(W)$. Then the following is true*

$$\sigma_j(M)\sigma_-(W) \le \sigma_j(MW) \le \sigma_j(M)\sigma_+(W) \quad \textit{and}$$

$$\sigma_j(N)\sigma_-(W) \le \sigma_j(WN) \le \sigma_j(N)\sigma_+(W)$$

*for $j = 1, \dots, \rho$ and $\rho \times \rho$ matrices $M$ and $N$.*

*Proof.* Since singular values are invariant in multiplication by a unitary matrix, it is sufficient to consider the case of a positive diagonal matrix $W$. In this case the claimed bounds readily follow from the Courant–Fischer Minimax Characterization [GV96, Theorem 8.1.2], [S98b, Theorem 3.3.2]. □

The next theorem is a special case of [S98b, Theorem 3.3.3] where $E = I_n$.

**Theorem 7.2.** *[Pb]. The following is true*

$$\sigma_j(W) - 1 \le \sigma_j(W + I_n) \le \sigma_j(W) + 1$$

*for an $n \times n$ matrix $W$ and for $j = 1, 2, \dots, n$.*

**Theorem 7.3.** *[Pb]. For positive integers $n$ and $r$, a normalized $n \times n$ matrix $A$, and a pair of unitary matrices $U$ and $V$ of size $n \times r$, write $C = A + UV^T$ and $G = I_r - V^T C^- U$. Suppose the matrices $A$ and $C = A + UV^T$ have full rank $\rho \ge r$. Then the matrix $G$ is nonsingular, and the following is true:*

$$\sigma_j(A^{-1})\sigma_-^2(C) - \sigma_-(C) \quad \le \quad \sigma_j(G^{-1}) \quad \le \quad \sigma_j(A^{-1})\sigma_+^2(C) + \sigma_+(C)$$

*for $\sigma_-(C) = \sigma_\rho(C)$, $\sigma_+(C) = \sigma_1(C) \le 2$, $\sigma_j(A^{-1}) = 1/\sigma_{\rho-j+1}(A)$, $j = 1, \dots, r$.*

**Corollary 7.1.** *[Pb]. Under the assumption of Theorem 7.1 the following is true*

$$\operatorname{cond}_2 G = \operatorname{cond}_2(G^{-1}) \le (\operatorname{cond}_2 C)\frac{\sigma_1(A^{-1})\sigma_+(C) + 1}{\sigma_r(A^{-1})\sigma_-(C) - 1}.$$

Suppose $A$ is an $n \times n$ nonsingular normalized strictly $(0, r)$ matrix with $r = \operatorname{nnul} A$ and $UV^T$ is a random or pseudo random unitary APP of a rank $r$. Then the values $\sigma_{n-j+1}(A)$ are small for $j \le r$ and not small for $j > r$, whereas the value $\sigma_n(C)$ is expected to be of the order of $\sigma_{n-r}(A) >> \sigma_{n-r+1}(A)$. Therefore, all singular values $\sigma_{r-j+1}(G) = 1/\sigma_j(G^{-1})$ for $j = 1, \dots, r$ are expected to be at most of the small order of $\sigma_{n-j+1}(A)$ and, furthermore (cf. Corollary 7.1), the condition number $\operatorname{cond}_2 G$ is expected to be of the order of at most $(\operatorname{cond}_2 C)^2\sigma_{n-r+1}(A)/\sigma_n(A)$.

Thus *the matrix $G$ is expected to have a small norm and to be well conditioned if $A$ is a strictly $(0, r)$ matrix.*

## 7.4 Solving linear systems with APCs and iterative refinement

Assume that $A$ and $C$ are nonsingular $n \times n$ matrices and $A$ is a normalized strictly $(0, r)$ matrix and choose an APPs $UV^T$ of rank $r$ according to the mentioned recipes. Equation (7.1) reduces a linear system $A\mathbf{y} = \mathbf{b}$ to linear systems with the matrices $C$ and $G = I_r - V^T C^{-1} U$. Suppose that these two matrices are well conditioned and the norm $||G||_2$ is small, as expected due to Sections 6.2, [P04, section 6] and Theorem 7.3.

Since the norm $||G||_2$ is small, it may seem that it is necessary to use a high precision when compute the matrices $W = C^{-1}U$ (or $V^T C^{-1}$) and $V^T C^{-1} U \approx I_r$. We, however, apply the iterative refinement in [GV96, Section 3.5.3], [S98a, Sections 3.3.4 and 3.4.5], and [H02, Chapter 11] to compute the matrix $C^{-1}U$, choosing its less customary variant with dynamically decreasing residuals. Furthermore, all numerical problems had been reduced to the summation and apply the known effective advance algorithms for this operation, [DH03] and [PS91]. Hereafter, refer to them as ASAs.

The customary iterative refinement algorithm would output approximation $W^*$ to the matrix $W = C^{-1}U$ with less than double precision, but add to the output values as many correct bits as possible, even though keep performing the computations with double precision.

Fix a sufficiently large integer $k$ and approximate the matrices $W = \sum_{i=0}^{k} W_i$ and $G = I_r - V^T W = I_r + \sum_{i=1}^{k} F_i$ by writing $U_0 = U$ and $G_0 = I_r$ and successively computing the matrices $W_i \leftarrow C^{-1}U_i$, $U_{i+1} \leftarrow U_i - CW_i$, $F_i \leftarrow -V^T W_i$, and $G_{i+1} \leftarrow G_i + F_i$ for $i = 0, 1, \ldots, k$.

First examine the rounding errors in the subiteration

$$U_0 = U, \quad W_i \leftarrow \mathrm{fl}(C^{-1}U_i) = C^{-1}U_i - E_i, \quad U_{i+1} \leftarrow \mathrm{fl}(U_i - CW_i) = U_i - CW_i + Z_i$$

for $i = 0, 1, \dots, k$.

**Theorem 7.4.** *[Pb]. For the above subiteration, the following is true*

$$C(W_0 + \cdots + W_k) = U + Z_0 + \cdots + Z_{k-1} - CE_k.$$

*Proof.* Due to the assumed equations, the following is true

$$CW_i = U_i + Z_i - U_{i+1}, \quad i = 0,\ 1,\ \dots,\ k-1.$$

Sum the latter equations to obtain that

$$C(W_0 + \cdots + W_{k-1}) = U_0 - U_k + Z_0 + \cdots + Z_{k-1}.$$

Substitute the equations $U_0 = U$ and $U_k = CW_k + CE_k$ and obtain the theorem. □

The theorem implies that the sum $W_0 + \cdots + W_k$ approximates the matrix $W = C^{-1}U$ with the error matrix $C^{-1}(Z_0 + \cdots + Z_{k-1}) - C^{-1}U_k$. Annihilate the error term $C^{-1}(Z_0 + \cdots + Z_{k-1})$ by applying the ASAs to compute all residuals $U_{i+1} = U_i - CW_i$ with no rounding errors. (Facilitate this task by filling the matrices $U$ and $V$ with shorter binary numbers (cf. Section 6.1.2).) Likewise avoid rounding errors at the stages of computing the products $F_i = -V^T W_i$ and the sums $G_{i+1} = G_i + F_i$ for $i = 0, 1, \dots, k$.

Actually only the matrix $G$ is needed, and Theorem 7.3 defines a small upper bound $\eta$ on its norm if $A$ is a strictly $(0, r)$ matrix. Therefore, dealing with $(0, r)$ matrices $A$, expect to have $G_i = 0$ for $i = 0, 1, \dots, k$ and a reasonably large integer $k$. Surely, it is not needed to store such matrices $G_i$. Furthermore, at the $i$th step of iterative refinement for $i \le k$ it is possible to overwrite the matrices $W_{i-1}$, $U_i$, and $F_{i-1}$ with their updates $W_i$, $U_{i+1}$, and $F_i$, to reduce the memory space.

It remains to show that the error term $C^{-1}U_{k+1}$ converges to zero as $k \to \infty$. Write $e_i = ||E_i||_2$ and $u_i = ||U_i||_2$ for all $i$.

**Theorem 7.5.** *[Pb]. Let $Z_i = 0$ for all $i$. Then the following is true*

$$U_{i+1} = CE_i \quad \textit{and consequently} \quad u_{i+1} \le e_i||C||_2 \quad \textit{for all} \quad i.$$

*Proof.* Pre-multiply the matrix equation $E_i = C^{-1}U_i - W_i$ by $C$ and combine the resulting equation with the equation $U_{i+1} - U_i + CW_i = Z_i = 0$. □

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

$$\begin{aligned}||(C+E)^{-1} - C^{-1}||_2 &\le ||(C+E)^{-1} - C^{-1}||_F \\ &\le 2||E||_F \max\{||C^{-1}||_2^2,\ ||(C+E)^{-1}||_2^2\}.\end{aligned}$$

*Proof.* See [GV96, Section 5.5.5]. □

**Corollary 7.2.** *[Pb]. Assume that $W_i = (C - \tilde{E}_i)^{-1}U_i = C^{-1}U_i - E_i$. Then*

$$e_i \le \delta u_i \quad \textit{where} \quad \delta = \delta(C, \tilde{E}_i) = 2||\tilde{E}_i||_F \max\{||C^{-1}||_2^2, ||(C - \tilde{E}_i)^{-1}||_2^2\}.$$

Combine Theorem 7.5 and Corollary 7.2, and obtain that $u_{i+1} \le \theta u_i$ for $\theta = \delta||C||_2$ for all $i$. Recall that $U = U_0$ and summarize the estimates in the following corollary.

**Corollary 7.3.** *[Pb]. Under the above assumptions the following is true*

$$e_i \le \delta u_i \le \delta\theta^i||U||_2 \quad \textit{for} \quad i = 1,\ 2,\ \ldots,\ k.$$

The corollary shows linear convergence of the error norms $e_i$ to zero as $i \to \infty$ provided $\theta < 1$. This implies linear convergence of the matrices $W_0 + \cdots + W_i$ to $W$ and, consequently, $U_0 + \cdots + U_i$ to $U$, $F_0 + \cdots + F_i$ to $F$, and $G_{i+1}$ to $G$.

Finally estimate the overall number of ops in the computations for a normalized strictly $(0, r)$ input matrix $A$ provided the A-modification $C = A + UV^T$ is well conditioned. In this case, the matrix $G$ has 2-norm in $O(1/\operatorname{cond}_2 A)$, and yielding this matrix $G$ within the error norm $\epsilon$ by stopping in $O((\log \operatorname{cond}_2 A)/\log(1/\epsilon))$

steps of iterative refinement. $O(M_{A,r})$ double precision ops per step is needed and therefore $O((M_{A,r} \log \mathrm{cond}_2 A)/\log(1/\epsilon))$ double precision ops overall provided it is possible to multiply the matrix $A$ by an $n \times r$ matrix in $M_{A,r}$ ops and have a crude approximation to the inverse matrix $C^{-1}$. The computational cost is low for smaller integers $r$.

Due to the equation $((C_-)^{-1})^{-1} = A^{-1} + VU^T$, it is possible to express the solution $\mathbf{y}$ to the linear system $A\mathbf{y} = \mathbf{b}$ as follows (cf. [Pb]),

$$\mathbf{y} = \mathbf{z} - VU^T\mathbf{b}, \quad (C_-)^{-1}\mathbf{z} = \mathbf{b}. \tag{7.2}$$

If the $q$-head of the SVD $A = S\Sigma T^T$, is known then it is possible to choose

$$U_q = \left(\left(\frac{1}{\sigma} - \frac{1}{\sigma_j}\right)\mathbf{s}_j\right)_j, \quad V_q = (\mathbf{t}_j)_j \quad \text{for} \quad j = 1, \ldots, q,$$

and obtain that

$$(C_-)^{-1} = \sum_{j=q+1}^{\rho} \sigma_j \mathbf{s}_j \mathbf{t}_j^T + \sigma \sum_{j=1}^{q} \mathbf{s}_j \mathbf{t}_j^T$$

and

$$I_q + U^T AV = \sigma^{-1}\,\mathrm{diag}(\sigma_j)_{j=1}^{q}.$$

# 8 Computation of determinants

## 8.1 Introduction comments

Next it is shown how application of the techniques of M- and A-preconditioning in the previous section helps us to simplify the computation of determinants.

## 8.2 SVD-based MPPs for the signs and the values of determinants

Rounding errors can easily ruin the results of the SVD-based computation of the determinants, but briefly this subject will be recalled for the sake of completeness. Suppose the determinant of a $(g, h)$ matrix $A$ available with the $g$-head and/or $h$-tail of its SVD is needed. ($g = 0$ or $h = 0$ are the cases usually dealt with.) Recall that $\det(\mathrm{A}) = (\det(B))\det(C)$ if $A = BC$ as well as if $A = \mathrm{diag}(B, C)$, apply Theorem 5.2, and obtain that

$$\det(A) = (\det(F_0))\left(\widehat{\prod}_j\left(\frac{\sigma_j}{\sigma}\right)\right)/\det(\tilde{S}_{g,h}),$$
$$\det(A) = (\det(F'_0))\left(\widehat{\prod}_j\left(\frac{\sigma_j}{\sigma}\right)\right)/\det(\tilde{T}_{g,h}),$$
$$\det(A) = (\det(F_{0,0}))\left(\widehat{\prod}_j\left(\frac{\sigma_j}{\sigma}\right)\right)/((\det(\tilde{S}_{g,h}))(\det(\tilde{T}_{g,h}))).$$

Here the symbol $\widehat{\prod}_j$ stands for the product in $j$ ranging from 1 to $g$ and from $n - h + 1$ to $n$, and assume the definitions of Theorem 5.2 for $r = l = 0$ and $m = n = \rho$, so that the matrices $\tilde{S}_{g,h}$ and $\tilde{T}_{g,h}$ from equations (1.6) are unitary, $\sigma_{n-h} \le \sigma \le \sigma_g$, and the matrices $F_0$, $F'_0$, and $F_{0,0}$, from the theorem 5.2, are expected to be better conditioned than a $(g, h)$ matrix $A$. Recall that $|\det(U)| = 1$ for a unitary matrix $U$.

It could be possible to further simplify the computation of the sign of the determinant. Namely, by recalling that $\sigma_j > 0$ for $0 < j \le \rho$, $\det(\Sigma^{(g)}) > 0$, and

$\det(\Sigma_h) > 0$, it is possible to deduce from equations (1.7)–(1.9) for $r = l = 0$ that

$$\begin{aligned}
\operatorname{sign}\det(A) &= \left(\operatorname{sign}\det\begin{pmatrix} T^{(g)T} \\ S_{g,h}^{(ORT)T} A \\ T_h^T \end{pmatrix}\right) \operatorname{sign}\det(\tilde{S}_{g,h}),\\
\operatorname{sign}\det(A) &= \left(\operatorname{sign}\det(S^{(g)}, AT_{g,h}^{(ORT)}, S_h)\right) \operatorname{sign}\det(\tilde{T}_{g,h}),\\
\operatorname{sign}\det(A) &= \left(\operatorname{sign}\det(S_{g,h}^{(ORT)T} AT_{g,h}^{(ORT)})\right) (\operatorname{sign}\det(\tilde{S}_{g,h})) \operatorname{sign}\det(\tilde{T}_{g,h})
\end{aligned}$$

where the unitary matrices $S_{g,h}^{(ORT)T}$, $T_{g,h}^{(ORT)T}$, $\tilde{S}_{g,h}$, and $\tilde{T}_{g,h}$ are defined in Section 1.9 (see equations (1.5) and (1.6)).

The three equations above also hold in numerical computation of the sign unless the rounding errors of computing the matrices

$$\tilde{S}_{g,h}, \quad \tilde{T}_{g,h}, \quad \begin{pmatrix} T^{(g)T} \\ S_{g,h}^{(ORT)T} A \\ T_h^T \end{pmatrix}, \quad (S^{(g)}, AT_{g,h}^{(ORT)}, S_h), \quad \text{and} \quad S_{g,h}^{(ORT)T} AT_{g,h}^{(ORT)}$$

exceed their smallest singular values. These values equal one for the unitary matrices $\tilde{S}_{g,h}$ and $\tilde{T}_{g,h}$ and equal $\sigma_{n-h}(A)$ for the three other matrices, whose norms equal $\sigma_g(A)$, provided $\sigma_{n-h}(A) \le 1 \le \sigma_g(A)$.

## 8.3 APC-based factorizations for the determinants

The same APCs could be used for determinants as in Section 6.1. Apply the primal and dual Schur Aggregation and the recursive dual Schur Aggregation in Section 7 but substantially simplify them by using the factorization

$$\det(A) = (\det(C)) \det(I_r - V^T C^{-1} U)$$

in (1.4) for $C = A + UV^T$. Seeking the sign of the determinant it could be safely to stop wherever the rounding errors in computing the auxiliary matrices are smaller than their smallest singular values.

By that way, if $C$ is a well conditioned matrix, this study task is reduced to computing

- the $r \times r$ matrix $G = I_r - V^T C^{-1} U$ (known as the Gauss transform of the matrix $W$ and as the Schur complement of its block $C$ [GV96, pages 95,103] (for $r < n$ it has been called an *A-aggregate* of the matrix $A$ (cf. [MP80])), and

- $\det G$.

# 9 Advanced summation and multiplication

In this section some effective semi-numerical summation/multiplication algorithms from [PMQRa] will be covered, hereafter referred to as ASAs, that is, the advanced algorithms for floating-point summation and multiplication, which yield a high precision or error-free output for a sequence of double precision additions, subtractions, and multiplications, even where many leading significant bits of the output are cancelled. The algorithms employ and extend the known algorithms in [H02], [LDB02], [DH03], [ORO05], [ROO06], and the extensive bibliography therein.

We employ these algorithms for high precision computation of the Schur complements $G = I_r - V^T C^{-1} U$. We use Matlab-like notation [M04] to reproduce (for completeness) three old basic ASAs used as building blocks in the advanced algorithms in [ORO05], [ROO06]. Assume the IEEE standard representation of floating-point numbers as $s = \sigma 2^e f$ where $\sigma$ is equal to $-1$ or one, $e$ is an integer in a fixed range $[1-r, r]$ for a fixed natural $r$, and $f$ is either zero or a binary number in the range $[1, 2)$ represented with $p+1$ bits, including the leftmost bit one, which is the leading and most significant bit. In particular $r = 127$ and $p = 23$ for the single precision IEEE standard floating-point numbers and $r = 1023$ and $p = 52$ for the double precision IEEE standard floating-point numbers.

## 9.1 Splitting of a floating-point number into two parts

The floating-point multiplication algorithm by Veltkamp employs splitting of a floating-point number into two parts by Dekker. Both algorithms appeared in [D71]. Assuming that $g$ is an integer, $0 < g \leq p$.

**Algorithm 9.1. Splitting of a floating-point number into two parts**.

$$\begin{aligned} \text{function}[x, y] &= \text{Split}(a) \\ c &= \text{fl}(\text{factor} \cdot a) \quad \% \quad \text{factor} = 2^g + 1 \\ x &= \text{fl}(c - (c - a)) \\ y &= \text{fl}(a - x) \end{aligned}$$

## 9.2 Transformation of the product of two floating-point numbers

The shorter precision numbers $x$ and $y$ satisfy the equation $a = x2^g + y$. Under the common assumption that $0 \leq \lceil p/2 \rceil - g \leq 1$, these are the half-precision numbers. For any integer $g$, the output value $y = a \mod 2^g = \sum_{i<g} a_i$ is the *residue modulo* $2^g$ of a binary number $a = \sum_{i \leq e} a_i$, $a_e = 1$, obtained by zeroing all its bits that represent the powers $2^i$ for $i \geq g$. Algorithm 9.1 also computes the remaining leading part $x = (a - y)/2^g$ in the binary floating-point representation of the number $a$. $a = 2^g x$ for $g < e - p - 1$ where $a = \sum_{i=e-p-1}^{e} a_i$ and $a = y$ for $g > e$.

**Algorithm 9.2. Transformation of the product of two floating-point numbers**.

$$\begin{aligned} \text{function}[x, y] &= \text{TwoProduct}(a, b) \\ x &= \text{fl}(a \cdot b) \\ [a_1, a_2] &= \text{Split}(a) \\ [b_1, b_2] &= \text{Split}(b) \\ y &= \text{fl}(a_2 \cdot b_2 - (((x - a_1 \cdot b_1) - a_2 \cdot b_1) - a_1 \cdot b_2)) \end{aligned}$$

The output floating-point numbers $x$ and $y$ satisfy the equation $a \cdot b = x + y$ and $x = \text{fl}(a \cdot b)$, so that the two latter algorithms reduce multiplication to addition.

## 9.3 Error-free transformation of the sum of two floating-point numbers

Similarly, the Knuth's algorithm of 1969 [K98] below transforms two input floating-point numbers $a$ and $b$ into two output floating-point numbers $x$ and $y$ such that

$$a + b = x + y \quad \text{and} \quad x = \text{fl}(a + b).$$

**Algorithm 9.3. Error-free transformation of the sum of two floating-point numbers**

$$\begin{aligned} \text{function}[x, y] &= \text{TwoSum}(a, b) \\ x &= \text{fl}(a + b) \\ z &= \text{fl}(x - a) \\ y &= \text{fl}((a - (x - z)) + (b - z)) \end{aligned}$$

The Kahan–Babušhka's and Dekker's algorithm [B69], [D71] outputs the same solution for the same problem provided $|a| \geq |b|$. It uses fewer ops but includes branches, which slow down the code optimization.

By combining the summation and multiplication algorithms, one can handle the sequences of ring operations, in particular the dot product computation. The computations boil down to summation of many numbers. The reader could be refereed to [ORO05], [ROO06] on the detailed analysis of this approach and to other cited bibliography on other successful techniques.

One can extend either of the two cited summation algorithms to floating-point summation of $h$ numbers for any $h$ by applying the Kahan–Babuška cascaded summation [H02]. Its variant in [ORO05] uses the Knuth's basic algorithm and approximates the sum $\sum_{i=1}^{T} s_i$ of $h$ numbers with an error of at most $(h/(2^p - h)) \sum_{i=1}^{h} |s_i|$ (cf. [ORO05, Lemma 4.2]).

Even the above algorithms, however, are not sufficient for the task, where having $|s| = |s_1 + \cdots + s_h| << |s_1| + \cdots + |s_h|$, and then applying some more involved algorithms from [PMQRT], which approximate the sums with arbitrarily high precision.

# 10 Numerical tests for signs of determinants with APPs

The following tasks need to be answered:

- Check if the approach of section 4.2 is also valid or is corrupted by rounding errors in the hard cases.
- Compare the results with Matlab results and theoretically with symbolic methods.
- How much of symbolic computation, within *arithmetic filtering*, is needed?
- The choice of highly accurate numerical floating-point algorithms for sums and products (see sections 7.4, 9, [ROO06], [LDB02], [H02], [PIMRT07] and [PIMRTYa]).

The signs of the determinants of nonsingular matrices $A = PML$ have been computed, where $P$ denoted permutation matrices, each swapping $k$ random pairs of the rows of the matrix $A$, and where $L$ and $M^T$ denoted random $n \times n$ lower triangular matrices with unit diagonal entries and with integer subdiagonal entries randomly sampled from the line intervals $[-\gamma, \gamma]$ for a fixed positive $\gamma$. It followed that $\det(A) = (-1)^k$. Such matrices for $k = 2n$ and $k = 2n - 1$ and for $\gamma \geq 5,000$ have been generated.

Both numerical subroutines in Matlab and the algorithm based on factorization (1.4) were applied.

To generate random APPs,

- first a positive integer $r$ was fixed,
- then two random $n \times r$ unitary matrices were generated, their entries truncated to represent them with 20 bits precision, thus obtaining two matrices

$\tilde{U}$ and $\tilde{V}$.

- then computing and scaling the matrix $\tilde{U}\tilde{V}^T$ of rank $r$ to yield an APP

$$UV^T = 2^d \tilde{U}\tilde{V}^T$$

for an integer $d$ such that $1/2 < \theta = ||UV^T||_2/||A||_2 \leq 2$ (cf. Section 6.2.3).

First assume that an APC $UV^T$ of rank one transforms an ill conditioned matrix $A$ into a well conditioned A-modification $C = A + UV^T$. For this large and important class of input matrices $A$ and APCs $UV^T$, $\det G = G = 1 - V^T C^{-1} U$ is a scalar. Its numerical computation, however, is a challenge because the value $|G|$ is very small under the above assumptions, as follows from the Theorem 7.3 for $r = 1$. Consequently, the value $V^T C^{-1} U$ is very close to one and must be computed with a high precision.

The problem and its solution are quite similar in the more general case where the ratios $\sigma_1(A)/\sigma_{n-r}(A)$ and $\sigma_{n-r+1}(A)/\sigma_n(A)$ are not large, but the matrix $A$ is ill conditioned because $\sigma_{n-r} >> \sigma_{n-r+1}$. In this case it could be possible to obtain a well conditioned A-modification $C = A + UV^T$ for an appropriate APC $UV^T$ of rank $r > 1$ (see [PIMRTYa]).

The numerical subroutines in Matlab performed poorly for matrices of the selected class. They lost the competition in accuracy not only to the slower symbolic subroutines in MAPLE but also to numerical tests using the algorithms given above. Already for $n = 4$ and $\gamma = 5,000$, Matlab's numerical outputs had the wrong sign in over 45% out of 100,000 runs and were off from the true value of $\det A$ by a factor of two or more in over 99% of the runs, whereas the algorithms given above produced correct output in over 99% of the runs for substantially larger $n$ and $\gamma$.

# References

[ABDPY97] F. Avnaim, J.-D. Boissonnat, 0. Devillers, F. P. Preparata, M. Yvinec, *Evaluating Signs of Determinants Using Single-Precision Arithmetic*, Algorithmica, 17, 111–132, 1997.

[ABM99] J. Abbott, M. Bronstein, T. Mulders, *Fast Deterministic Computations of the Determinants of Dense Matrices*, Proc. of International Symposium on Symbolic and Algebraic Computation (ISSAC'99), 197–204, ACM Press, New York, 1999.

[AF92] D. Avis, K. Fukuda, *A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra*, Discrete Comput. Geometry, 8, 295-313, 1992.

[B69] I. Babuška, *Numerical Stability in Mathematical Analysis*, Information Processing, 68 (Proc. of IFIP Congress), 11–23, North-Holland, Amsterdam, 1969.

[B68] E. H. Bareiss, *Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination*, Math. of Come., 22, 565-578, 1968.

[BBCDDDEPRV93] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van Der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1993.

[BDDRV00] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[BEPP99] H. Brönnimann, I. Z. Emiris, V. Y. Pan, S. Pion, *Sign Determination in Residue Number Systems*, Theoretical Computer Science, 210, 1, 173-197, 1999. Proceedings Version in Proc. 13th Ann. ACM Symp. on Computational Geometry, 174-182, ACM Press, New York, 1997.

[BKMNSU95] C. Burnikel, J. Könnemann, K. Melhom, S. Näher, S. Schirra, C. Uhrig, *Exact Geometric Computation in LEDA*, Proc. 11th Ann. ACM Symp. on Computational Geometry, C18619, 1995. Package available at http://www. mpi- sb.mpg.de/LEDA/leda.html

[BMP00] D. Bondyfalat, B. Mourrain, V. Y. Pan, *Computation of a Specified Root of a Polynomial System of Equations Using Eigenvectors*, Linear Algebra and Its Applications, 319, 193–209, 2000.

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

[C05] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, Cambridge, England, 2005.

[C92] K. L. Clarkson, *Safe and Effective Determinant Evaluation*, Proc. 33rd Ann. IEEE Symp. on Foundations of Computer Science, 387–395, IEEE Computer Society Press, Los Alamitos, California, 1992.

[CD80] S. Conte, C. deBoor, *Elementary Numerical Analysis: An Algorithmic Approach*, McGraw Hill, New York, 1980.

[D71] T. J. Dekker, *A Floating-point Technique for Extending the Available Precision*, Numerische Math., 18, 224–242, 1971.

[D61] E. Durand, *Solutions Numeriques des Équations Algébriques. Vol. II: Systémes de Plusieurs Équations. Valenrs Propres des Matrices*, Masson et Cie, Paris, 1961.

[DH03] J. Demmel, Y. Hida, *Accurate and Efficient Floating Point Summation*, SIAM J. on Scientific Computing, 25, 1214–1248, 2003.

[DL97] M. M. Deza, M. Laurent, *Geometry of Cuts and Metrices*, Springer, Berlin, 1997.

[DL94] M. Deza, M. Laurent, *Applications of Cut Polyhedra*, J. of Computational and Applied Math., 55, 1, 191-216 and 55, 2, 217-247, 1994.

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

[E98] I. Z. Emiris, *A Complete Implementation for Computing General Dimensional Convex Hulls*, Intern. J. Computational Geom. & Applications, 8, 2, 223-253, 1998.

[E67] J. Edmonds, *Systems of Distinct Representatives and Linear Algebra*, J. Res. Nat. Bur. Standards, Sect. B, 71, 4, 241-245, 1967.

[EC95] I. Z. Emiris, J.F. Canny, *A General Approach to Removing Degeneracies*, SIAM J. Computing, 24, 3, 650-664, 1995.

[EGV00] W. Eberly, M. Giesbrecht, and G. Villard, *On Computing the Determinant and Smith Form of an Integer Matrix*, Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS'2000), 675–685, IEEE Computer Society Press, Los Alamitos, California, 2000.

[EP05] I. Z. Emiris, V. Y. Pan, *Improved Algorithms for Computing Determinants and Resultants*, J. of Complexity, 21, 1, 43–71, 2005. Proceedings version in Proc. of the 6th International Workshop on Computer Algebra in Scientific Computing (CASC '03), edited by E. W. Mayr, V. G. Ganzha, and E. V. Vorozhtzov, 81–94, Technische Univ. München, Germany, 2003.

[EPY98] I. Z. Emiris, V. Y. Pan, Y. Yu, *Modular Arithmetic for Linear Algebra Computations in the Real Field*, J. of Symbolic Computation, 21, 1–17, 1998.

[ES87] R. M. Erdahl, V. H. Smith (editors), *Density Matrices and Density Functionals*, Proc. of the A. John Coleman Symp., Reidel, Dordrecht, 1987.

[EY39] C. Eckart, G. Young, *A Principle Axis Transformation for Non-Hermitian Matrices*, Bulletin of American Society (New Series), 45, 118-121, 1939.

[F64] L. Fox, *An Introduction to Numerical Linear Algebra*, Clarendon, 1964.

[FR94] K. Fukuda, V. Rosta, *Combinatorial Face Enumeration in Convex Polytopes*, Computational Geometry, Theory and Applications, 4, 191-198, 1994.

[FV96] S. Fortune, C. J. Van Wyk, *Static Analysis Yields Efficient Exact Integer Arithmetic for Computational Geometry*, ACM Trans. Graph, 15, 3, 223-248, 1996.

[FV93] S. Fortune, C. J. Van Wyk, *Efficient Exact Arithmetic for Computational Geometry*, Proc. 9th Ann. ACM Symp. on Computational Geometry, 163-172, 1993.

[G72] H. Goldstine, *The Computer from Pascal to von Neumann*, Princeton Univ. Press, Princeton NJ, 1972.

[GV96] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 3rd edition, The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[H02] N. J. Higham, *Accuracy and Stability in Numerical Analysis*, SIAM, Philadelphia, 1996. *Accuracy and Stability in Numerical Analysis*, SIAM, Philadelphia, 2002 (second edition).

[H64] F. Hohn, *Elementary Matrix Algebra*, Macmillan, New York, 1964.

[K98] D. E. Knuth, *The Art of Computer Programming: Volume 2, Seminumerical Algorithms*, Addison-Wesley, Massachusetts, 1969 (first edition), 1981 (second edition), 1998 (third edition).

[K66] W. Kahan, *Numerical Linear Algebra*, Canadian Math. Bulletin, 9, 751-807, 1966.

[KV04a] E. Kaltofen, G. Villard, *Computing the Sign or the Value of the Determinant of an Integer Matrix*, a Complexity Survey, J. Computational Applied Math., 162, 1, 133–146, 2004.

[LDB02] X. Li, J. Demmel, D. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Kang, A. Kapur, M. Martin, B. Thompson, T. Tung, D. Yoo, *Design, Implementation and Testing of Extended and Mixed Precision BLAS*, ACM Transactions on Math. Software, 28, 152–205, 2002. http //crd.lbl.gov/ xiaoye/XBLAS/.

[LZ05] T. W. Li, Z. Zeng, *A Rank Revealing Method with Updating, Downdating and Applications*, SIAM J. on Matrix Analysis and Applications, 26, 918–946, 2005.

[M04] The MathWorks Inc., *Matlab User's Guide*, Version 7, 2004.

[M60] T. Muir, *The Theory of Determinants in Historical Order of Development*, four volumes bound as two (I, 1693-1841; II, 1841-1860; III, 1861-1880; IV, 1881-lQOO), Dover, New York, 1960; *Contributions to the History of Determinants*, 1900-1920, Blackie and Son, London, 1930 and 1950.

[M55] R H. Macmillan, *A New Method for the Numerical Evaluation of Determinants*, J. Roy. Aeronaut. Soc., 59, 772, 1955.

[MA93] R. E. M. Moore, I. O. Angell, *Voronoi Polygons and Polyhedra*, J. of Computational Physics, 105, 301-305, 1993.

[MP80] W. L. Miranker, V. Y. Pan, *Methods of Aggregations*, Linear Algebra and Its Applications, 29, 231–257, 1980.

[ORO05] T. Ogita, S. M. Rump, S. Oishi, *Accurate Sum and Dot Product*, SIAM Journal on Scientific Computing, 26, 6, 1955–1988, 2005.

[Pa] V. Y. Pan, *Null Aggregation*, preprint.

[Pb] V. Y. Pan,, *Additive Preconditioning and the Schur Aggregation*, preprint.

[P04] V. Y. Pan, *On Theoretical and Practical Acceleration of Randomized Computation of the Determinant of an Integer Matrix*, Zapiski Nauchnykh Seminarov POMI (in English), 316, 163–187, St. Petersburg, Russia, 2004.

[P88] V. Y. Pan, *Computing the Determinant and the Characteristic Polynomials of a Matrix via Solving Linear Systems of Equations*, Information Processing Letters, 28, 71–75, 1988.

[P87] V. Y. Pan, *Complexity of Parallel Matrix Computations*, Theoretical Computer Science, 54, 65–85, 1987.

[PIMRTYa] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, I. Taj-Eddin, Y. Tang, X. Yan, *Additive Preconditioning and Aggregation in Matrix Computations*, Computers and Mathematics with Applications, 2006, in press.

[PIMRTYb] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, *Additive Preconditioning for Matrix Computations*, Preprint.

[PIMRT07] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Wang, X. Yan, *Root-finding with Eigen-solving*, pages 219–245 in Symbolic-Numerical Computation, (Dongming Wang and Lihong Zhi editors), Birkhäuser, Basel/Boston, 2007.

[PMQRa] V. Y. Pan, B. Murphy, G. Qian, R. E. Rosholt, *High Precission Summation*, preprint.

[PMQRT] V. Y. Pan, B. Murphy, G. Qian, R. E. Rosholt, I. Taj-Eddin, *Numerical Computation of Determinants with Additive Preconditioning*, Preprint.

[PS91] V. Y. Pan, R. Schreiber, *An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications*, SIAM Journal on Scientific and Statistical Computing, 12, 5, 1109–1131, 1991.

[PY01] V. Y. Pan, Y. Yu, *Certification of Numerical Computation of the Sign of the Determinant of a Matrix*, Algorithmica, 30, 708–724, 2001.

[PYS97] V. Y. Pan, Y. Yu, C. Stewart, *Algebraic and Numerical Techniques for the Computation of Matrix Determinants*, Computers & Math. (with Applications), 34, 1, 43-70, 1997.

[R52] J. B. Rosser, *A Method of Computing Exact Inverses of Matrices with Integer Coefficients*, J. R.es. Na. Bur. Standards, Sect. B, 49, 349-358, 1952.

[ROO06] S. M. Rump, T. Ogita, S. Oishi, *Accurate Floating-Point Summation*, Tech. Report 05.12, Faculty for Information and Communication Sciences, Hamburg University of Technology, November 2005, 41 pages, submitted for publication, 2006.

[S04] A. Storjohann, *The Shifted Number System for Fast Linear Algebra on Integer Matrices*, Technical Report TR CS-2004-18, School of Computer Science, University of Waterloo, Canada, April 2004.

[S98a] G. W. Stewart, *Matrix Algorithms, Vol I: Basic Decompositions*, SIAM, Philadelphia, 1998.

[S98b] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, 1998.

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

[V03] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge, England, 2003.

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

[Y97] C. Yap, *Towards Exact Geometric Computation*, Computational Geometry, Theory and Applications, 7, 3-23, 1997.

[YD95] C. K. Yap, T. Dubhe, *The exact Computation Paradigm*, D. Du and F. Hwang, editors, Computing in Euclidean Geometry. World Scientific Press, 1995.

[Z79] R. E. Zippel, *Probabilistic Algorithms for Sparse Polynomials*, Proceedings of EUROSAM'79, Lecture Notes in Computer Science, 72, 216–226, Springer, Berlin, 1979.