

# Schur Aggregation for Linear Systems and Determinants<sup>★</sup>

V. Y. Pan<sup>a,\*</sup>, D. Grady<sup>c</sup>, B. Murphy<sup>a</sup>, G. Qian<sup>b</sup>,  
R. E. Rosholt<sup>a</sup>, A. D. Ruslanov<sup>c</sup>

<sup>a</sup>*Department of Mathematics and Computer Science, Lehman College, City  
University of New York, Bronx, NY 10468, USA*

<sup>b</sup>*Ph.D. Program in Computer Science, City University of New York, New York,  
NY 10036 USA*

<sup>c</sup>*Department of Computer and Information Sciences, State University of New  
York, Fredonia, NY 14063 USA*

---

## Abstract

We apply our recent preconditioning techniques to the solution of linear systems of equations and computing determinants. We combine these techniques with the Sherman–Morrison–Woodbury formula, its new variations, aggregation, iterative refinement, and advanced algorithms that rapidly compute sums and products either error-free or with the desired high accuracy. Our theoretical and experimental study shows the power of this approach.

*Key words:* Additive preconditioning, Linear systems of equations, Determinants, Sherman–Morrison–Woodbury formula, Iterative refinement  
*2000 MSC:* 65A12, 65F22, 65F05, 65F40

---

## 1 Introduction

### 1.1 Traditional preconditioning

The popular techniques of preconditioning facilitate the solution of an ill conditioned linear system of equations  $\mathbf{A}\mathbf{y} = \mathbf{b}$  by transforming it into an equivalent, but better conditioned linear system  $MAN\mathbf{x} = M\mathbf{b}$  for  $\mathbf{y} = N\mathbf{x}$  and appropriate nonsingular preconditioners  $M$  and  $N$ . The latter linear system can be solved faster and more accurately (see [1], [3], [6], [17], and the extensive bibliography therein). In particular preconditioning is vital for some effective iterative algorithms, such as the Conjugate Gradient algorithms and iterative refinement.

Traditional multiplicative preconditioning flourishes for large, but special classes of linear systems. Generally it is as costly as approximate factorization or inversion. Furthermore it may require additional care of preserving sparseness and structure of an input matrix.

### 1.2 Additive preconditioning

Our point of departure is the alternative techniques of *additive preconditioning* in [33]–[36], [39], [40], [42], [43], that is selecting an *additive preconditioner*  $P$  for an ill conditioned input matrix  $A$  and computing its better conditioned *additive modification*  $C = A + P$ . Hereafter we write “A-” for “additive”, “APC” for “additive preconditioner”,  $A^T$  for the transpose and  $A^H$  for the Hermitian (that is complex conjugate) transpose of a matrix  $A$ , so that  $A^H = A^T$  for a real matrix  $A$ . We observe the following attractive features of A-preconditioning.

---

\* Supported by PSC CUNY Awards 68291–0037, 69330–0038, and 69350–0038. Some results of this paper have been presented at the International Conferences on the Matrix Methods and Operator Equations in Moscow, Russia, in June of 2005 and July 2007, on the Foundations of Computational Mathematics (FoCM’2005) in Santander, Spain, in July 2005, and on Industrial and Applied Mathematics, in Zürich, Switzerland, in July 2007, as well as at the SIAM Annual Meeting, in Boston, in July 2006, and at the International Workshop on Symbolic-Numeric Computation (SNC’07) in London, Ontario, Canada, in July 2007.

\* Corresponding author.

*Email addresses:* [victor.pan@lehman.cuny.edu](mailto:victor.pan@lehman.cuny.edu) (V. Y. Pan), [grady@cs.fredonia.edu](mailto:grady@cs.fredonia.edu) (D. Grady), [brian.murphy@lehman.cuny.edu](mailto:brian.murphy@lehman.cuny.edu) (B. Murphy), [gqian@gc.cuny.edu](mailto:gqian@gc.cuny.edu) (G. Qian), [rhys.rosholt@lehman.cuny.edu](mailto:rhys.rosholt@lehman.cuny.edu) (R. E. Rosholt), [anatole@cs.fredonia.edu](mailto:anatole@cs.fredonia.edu) (A. D. Ruslanov).

*URL:* <http://comet.lehman.cuny.edu/vpan/> (V. Y. Pan).

- APCs are readily available for a large class of matrices
- We can extend to APCs the structure and sparseness of an input matrix
- A-preconditioning has a wide range of applications, which include rank deficient matrix computations and eigen-solving [40], [42], [43]

According to both theoretical and extensive experimental study in [35], [36], a matrix  $P = UV^H$  is likely to be an APC for a given ill conditioned matrix  $A$ , that is to define a well conditioned A-modification  $C = A + P$ , if the ratio  $\frac{\|A\|}{\|P\|}$  is neither large nor small,  $U$  and  $V$  are random matrices of full rank  $r$ , and the matrix  $A$  has at most  $r$  singular values that are small relatively to the norm  $\|A\|$ . Moreover, one can choose a *weakly random* APC  $P$ , endowed with the desired patterns of sparseness and structure. In Section 12 we sketch a further simplification of our approach, based on weakly random expansion of the input matrix by appending to it new rows and columns.

### 1.3 How to employ additive preconditioners

In this paper we apply such APCs to facilitate the solution of the original ill conditioned linear system of equations  $A\mathbf{y} = \mathbf{b}$  and the computation of the determinant  $\det A$ . We rely on the Sherman–Morrison–Woodbury formula for matrix inversion and its modifications (see some alternative techniques in [40], [42], [43]). Hereafter we use the abbreviation *SMW*. The SMW formulae reduce our tasks to computations with the A-modification  $C$  and a Schur complements  $G$  of a smaller size in the input matrix  $A$ , which we call a *Schur Aggregate*. For scaled weakly random APCs the matrices  $C$  and  $G$  are expected to have smaller condition numbers than the input matrix  $A$ .

This paper is devoted to further elaboration upon our approach and its analysis. We cover the choice of the ranks of APCs, specify the algorithms, estimate the rounding errors, and relate to each other the singular values (and thus also the norms and conditioning) of the matrices  $A$ ,  $C$  and  $G$ . In the case of an ill conditioned input we face numerical problems at the stage of computing the Schur aggregate  $G$ , but we overcome them by extending the classical iterative refinement. Our extension is in the style of Hensel’s lifting in [8] and [22], which is one of the basic techniques of symbolic computing.

### 1.4 Precision of computing

In our computations we proceed with double precision, but incorporate the advanced algorithms that rapidly compute sums and products error-free or with a desired high accuracy. This symbolic part of our computations is involved into the computation of the A-preconditioners  $C$  and is the basis for

our extension of the classical numerical iterative refinement, whose output values we represent as the sums of double precision pieces. Unlike the case of the classical iterative refinement and Hensel’s lifting, we bypass storing these pieces and only store the Schur aggregates with double precision. According to our analysis and test results, the leading bits in the representation of the entries of the Schur aggregates vanish from the beginning and deep into the refinement process, and we safely ignore these bits, to save the computer time and the memory space.

### 1.5 The bit-operation complexity estimates

Hereafter “ops” is our abbreviation for “arithmetic operations”.

Let us quantify the gain from combining preconditioning with the Schur aggregation where we seek the solution of an ill conditioned linear system  $\mathbf{A}\mathbf{y} = \mathbf{b}$ . Recall that the condition number of a matrix  $A$  is roughly the ratio  $\frac{\text{output error norm}}{\text{input error norm}}$ . So, no matter which algorithm we apply, we must process  $n^2$  input entries with the precision  $p_{comp}$  of at least  $p_{out} + \log_2 \text{cond } A$  bits (here and hereafter  $p_{out}$  denotes the required output precision). This, however, can still be less costly than the order of  $n^3$  ops with the precision of the order  $p_{comp}$  above, required in Gaussian elimination.

Our approach involves multiplication of  $n \times n$  by  $n \times r$  matrices (in  $O(rn^2)$  ops) and iterative refinement of the solution of some auxiliary well conditioned linear systems of equations. We first compute a crude approximation to LU or PLU factorization of the well conditioned matrix  $C$  or to its inverse, by using  $O(n^3)$  ops with the IEEE standard double precision  $p_{double}$ . Then in each loop of iterative refinement we use  $O(rn^2)$  ops with this precision to produce about  $p_{double} - \log_2 \text{cond } C$  new correct bits per an output value. For a well conditioned matrix  $C$  and a smaller rank  $r$  (say for a constant  $r$ ) the resulting overall time bound in  $O(\frac{rn^2 p_{out}}{p_{double} - \log_2 \text{cond } C})$  shows the order of magnitude acceleration versus the solution of an ill conditioned linear system  $\mathbf{A}\mathbf{y} = \mathbf{b}$  with Gaussian elimination.

### 1.6 Application to computing determinants

The classical problem of computing determinants is highly important to the fundamental geometric and algebraic–geometric computations (see [4], [11], [15], [44], [47], and the extensive bibliography therein). The computation of convex hulls and Voronoi diagrams essentially amounts to recursive computation of the signs of determinants, whereas the computation of the values of

determinants is a basic stage of the resultant methods for polynomial systems of equations [15].

The sign of  $\det A$ , computed numerically, with unit roundoff  $u$ , can be certified if  $\|A\| \operatorname{cond} A > cu$  for a certain constant  $c$  (cf. [44]). Generally, numerical computation of both sign and value of the determinant is harder where the input matrix is ill conditioned, which motivates using  $A$ -preconditioning. In our extensive tests our symbolic–numerical approach regularly yielded correct output where the customary numerical techniques fail. We refer the readers to [4], [13], [32], [50], and the bibliography therein on some effective symbolic algorithms for computing determinants.

### *1.7 Symbolic–numerical aspects*

Technically the present paper is more numerical than our earlier work on symbolic–numerical computations in [5], [14], [24], [26], [28]–[31], [34], and the bibliography therein. Symbolic techniques, however, are critical where we deal with ill conditioned inputs and must counter the inherent numerical problems. Based on combining our randomization, preconditioning, and aggregation techniques, we reduce the original ill conditioned task to extended iterative refinement for well conditioned linear systems of equations, and we apply fast symbolic error-free summation and multiplication to support both preconditioning and refinement. We also note that one of our main tasks is the computation of determinants, motivated by some fundamental problems of algebraic-geometric computations, which are commonly viewed as the subjects of symbolic computations.

### *1.8 Organization of the paper*

We organize our presentation as follows. In the next section we introduce basic definitions. In Section 3 we recall some results in [35], [36], and [51] that relate the power of APCs to their ranks. In Section 4 we cover the SMW formula and its new variations. In Section 5 we specify our algorithms. In Sections 6 and 7 we estimate the rounding errors of our computations and link the singular values of the input matrix, its  $A$ -modification, and the Schur aggregate. In Section 8 we outline the techniques for double precision computations that produce high accuracy output. This includes the extension of the classical iterative refinement, which we cover in some detail in Section 9, including the precision and error bounds. We comment on preserving and employing matrix structure in Section 10. In Section 11 we describe our numerical tests. The tests have been performed jointly by all authors. Otherwise the paper is due to the first author and should be cited as his work. In Section 12 we sketch

a further simplification of our approach by using expansion as well as application of weakly randomized multiplicative preconditioning as a substitution for pivoting. This paper is the proceedings version of the paper [39].

## 2 Basic definitions

$M^H$  denotes the Hermitian transpose of a matrix  $M$ . ( $M^H$  is the transpose  $M^T$  if  $M$  is a real matrix.) We assume the customary notation for matrix computations in [2], [9], [18], [19], [48], [49], e.g.,  $\mathbf{v}$  is a vector,  $I_k$  denotes the  $k \times k$  identity matrix,  $I$  is  $I_k$  for an unspecified  $k$ .

$\|M\|_l$  for  $l = 1, 2, \infty$  denotes the  $l$ -norm of a matrix  $M$ .  $\|\cdot\| = \|\cdot\|_2$  denotes the 2-norm of a matrix. A matrix  $A$  is *normalized* if  $\|A\| = 1$  and is unitary if  $A^H A = I$ . A matrix  $A$  of a rank  $\rho$  has the Frobenius norm  $\|A\|_F$  such that  $\|A\|_F^2 = \text{trace}(A^H A) = \sum_{j=1}^{\rho} \sigma_j^2(A)$  and  $\|A\| \leq \|A\|_F \leq \sqrt{\rho} \|A\|$ .

Hereafter we use the abbreviation “SVD” for “Singular Value Decomposition”. The *compact SVD* of an  $m \times n$  matrix  $A$  of a rank  $\rho$  is the decomposition

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

where  $S^{(\rho)} = (\mathbf{s}_j)_{j=1}^{\rho}$  and  $T^{(\rho)} = (\mathbf{t}_j)_{j=1}^{\rho}$  are unitary matrices,  $S^{(\rho)H} S^{(\rho)} = I_{\rho}$ ,  $T^{(\rho)H} T^{(\rho)} = I_{\rho}$ ,  $\Sigma^{(\rho)} = \text{diag}(\sigma_j)_{j=1}^{\rho}$  is a diagonal matrix,  $\mathbf{s}_j$  and  $\mathbf{t}_j$  are  $m$ - and  $n$ -dimensional vectors, respectively, and  $\sigma_j = \sigma_j(A)$  for  $j = 1, \dots, \rho$  are the singular values of the matrix  $A$ ,  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\rho} > 0$ .

The *Moore-Penrose generalized inverse* of an  $m \times n$  matrix  $A$  of a rank  $\rho$  (also called its *pseudo inverse*) is the matrix  $A^+ = \sum_{j=1}^{\rho} \sigma_j^{-1} \mathbf{t}_j \mathbf{s}_j^H$ , equal to  $A^{-1}$  where  $m = n = \rho$ .

$\text{cond } A = \frac{\sigma_1}{\sigma_{\rho}} = \|A\| \|A^+\|$  is the condition number of a matrix  $A$ . For any matrix product  $MN$  we have  $\text{cond}(MN) \leq (\text{cond } M) \text{cond } N$ . A matrix is ill (resp. well) conditioned if its condition number is large (resp. not large). The concepts “large”, “well conditioned” and “ill conditioned” can be quantified depending on the computational task and computer environment.

## 3 The ranks, the norms, and the preconditioning power of APCs

The following sharp lower bounds from [51] relate preconditioning power of APCs to their rank.

**Theorem 3.1.** For any  $n \times n$  matrix  $A \geq 0$ , we have

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

The minimum is reached where

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

**Theorem 3.2.** For any  $n \times n$  nonsingular matrix  $A$ ,

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

Knowing the SVD of an  $m \times n$  input matrix  $A$ , one could readily compute an APC supporting this theorem [51], but actually one should avoid the costly computation of the SVD. So we define APCs as the products  $P = UV^H$  of pairs of random rectangular matrices  $U$  and  $V$  of full ranks  $r < \min\{m, n\}$ .

According to the analysis and extensive tests in [35], [36], for an ill conditioned  $n \times n$  matrix  $A$ , an integer  $r < n$ , and random  $n \times r$  generators  $U$  and  $V$  scaled so that the ratio  $\frac{\|A\|}{\|UV^H\|}$  is neither large nor small, the A-modification  $C = A + UV^H$  tends to have the condition number of the order  $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$ , versus  $\text{cond } A = \frac{\sigma_1(A)}{\sigma_n(A)}$ . (In our tests the increase or decrease of the ratio  $\frac{\|A\|}{\|UV^H\|}$  by a larger factor  $\theta$  tended to cause the increase of  $\text{cond } C$  by roughly the same factor.) Moreover it is sufficient to generate weakly random matrices  $U$  and  $V$ , defined by fewer parameters. In the extensive tests in [35] we safely endowed the generator  $U$  with various patterns of structure and sparseness, filled it with lower precision integers (e.g., with  $-2, -1, 1$ , and  $2$  or just with  $-1, 1$ ), scaled it error-free by a proper power of two, and then set  $V = U$ . In our tests the power of A-preconditioning was regularly preserved under these restrictions.

To simplify the subsequent computations, one should generate APCs of the smallest ranks. E.g., one can generate scaled weakly random APCs whose rank  $r$  is increasing from a lower bound  $r_-$  until the matrix  $C$  passes a test for being well conditioned. Alternatively, one can first generate an APC  $UV^H$  of a larger rank  $r_+ \geq r$ , to yield a well conditioned A-modification  $C = A + UV^H$ , and then generate APCs of recursively decreasing ranks as long as the A-modifications remain well conditioned. One can apply the binary search for the rank or perform the test concurrently for a number of candidate values  $r$ . For a large class of input matrices  $A$  the minimum rank for APCs is known to lie in a fixed small range  $[r_-, r_+]$ . For a large class of inputs our search ends immediately with  $r = 1$ , and then  $G$  is a scalar and  $\text{cond } G = 1$ .

As the stopping criterion in our search we test whether the matrix  $C$  is well conditioned. We can by apply the effective condition estimators in [18, Section

3.5.4], [19, Chapter 15], and [48, Section 5.3]. Very crude estimates would suffice for us, but in our algorithms it could be more effective to approximate the inverses  $C^{-1}$  or generalized inverses  $C^+$  explicitly because we need them in our subsequent computations and because we can update them rapidly (see Remark 5.1 in Section 5). Approaching the rank  $r$  from above is also attractive because this enables us to avoid inversion of ill conditioned matrices. Furthermore we can combine compression of the APCs with their refinement according to the map  $U \leftarrow Q(C^+UT(U))$  and  $V^H \leftarrow Q(T^H(V)V^HC^+)$ . Here  $T(V)$  and  $T(U)$  are unitary matrices whose columns form bases for the left and right singular spaces, respectively, associated with the  $r$  smallest singular values of the matrices  $V^HC^{-1}A$  and  $AC^{-1}U$ , respectively, for a fixed  $r < r_+$ . According to the analysis in [40], [43], and [51] and the extensive tests in [43, Section 6], [51], this approach enhances the quality of computed APCs, at the cost of relatively small work. Without compression, we can still yield some refinement of an APC  $UV^H$  via the simplified maps  $U \leftarrow Q(C^+U)$  and  $V^H \leftarrow Q(V^HC^+)$ .

## 4 The Sherman–Morrison–Woodbury formula and its variations

### 4.1 The case of nonsingular matrices

For a  $2 \times 2$  block matrix

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

the matrix  $G_{22} = B_{22} - B_{21}B_{11}^+B_{12}$  (respectively,  $G_{11} = B_{11} - B_{12}B_{22}^+B_{21}$ ) is the *Schur complement* of its northwestern block  $B_{11}$  (respectively, southeastern block  $B_{22}$ ) provided  $B_{11}^+B_{11} = I$  or  $B_{11}B_{11}^+ = I$  (respectively,  $B_{22}^+B_{22} = I$  or  $B_{22}B_{22}^+ = I$ ) [18, pages 95, 103], [48, page 155]. We immediately verify the following lemma.

**Lemma 4.1.** Let the above block matrix  $B$  be nonsingular and let

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

be the respective block representation of the matrix  $B^{-1}$  for some matrices  $W$ ,  $X$ ,  $Y$  and  $Z$ . Then  $W = G_{11}^{-1}$  (resp.  $Z = G_{22}^{-1}$ ) if the block  $B_{22}$  (resp.  $B_{11}$ ) is nonsingular.

**Theorem 4.1.** For  $n \times r$  matrices  $U$  and  $V$  and an  $n \times n$  matrices  $A$ , let the matrix  $C = A + UV^H$  be nonsingular. Then the matrices  $A$  and  $G =$

$I_r - V^H C^{-1} U$  are the respective Schur complements of the blocks  $I_r$  and  $C$  in the matrix  $W = \begin{pmatrix} C & U \\ V^H & I_r \end{pmatrix}$  such that

$$\det W = \det A = (\det C) \det G. \quad (1)$$

Furthermore [18, page 50], [48, Corollary 4.3.2], if the matrix  $A$  is nonsingular, then so is the matrix  $G$ , and we have the Sherman–Morrison–Woodbury formula  $(C - UV^H)^{-1} = C^{-1} + C^{-1}UG^{-1}V^HC^{-1}$ . (Hereafter we refer to this equation as the SMW formula.)

*Proof.* These results are well known, but are basic for us, and we supply their short proof. Begin with the factorization

$$\begin{aligned} \begin{pmatrix} C & U \\ V^H & I_r \end{pmatrix} &= \begin{pmatrix} I_n & U \\ 0 & I_r \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & I_r \end{pmatrix} \begin{pmatrix} I_n & 0 \\ V^H & I_r \end{pmatrix} \\ &= \begin{pmatrix} I_n & 0 \\ V^H C^{-1} & I_r \end{pmatrix} \begin{pmatrix} C & 0 \\ 0 & G \end{pmatrix} \begin{pmatrix} I_n & C^{-1}U \\ 0 & I_r \end{pmatrix}, \end{aligned}$$

which implies equations (1). Invert this factorization to obtain that

$$\begin{aligned} \begin{pmatrix} A^{-1} & X \\ Y & Z \end{pmatrix} &= \begin{pmatrix} I_n & 0 \\ -V^H & I_r \end{pmatrix} \begin{pmatrix} A^{-1} & 0 \\ 0 & I_r \end{pmatrix} \begin{pmatrix} I_n & -U \\ 0 & I_r \end{pmatrix} \\ &= \begin{pmatrix} I_n & -C^{-1}U \\ 0 & I_r \end{pmatrix} \begin{pmatrix} C^{-1} & 0 \\ 0 & G^{-1} \end{pmatrix} \begin{pmatrix} I_n & 0 \\ -V^H C^{-1} & I_r \end{pmatrix} \\ &= \begin{pmatrix} C^{-1} + C^{-1}UG^{-1}V^HC^{-1} & X \\ Y & Z \end{pmatrix} \end{aligned}$$

for some matrices  $X$ ,  $Y$ , and  $Z$ . □

**Remark 4.1.** Equation (1) also follows from the two equations  $\det A = (\det C) \det(I_n - C^{-1}UV^H)$  (implied by the equation  $A = C(I_n - C^{-1}UV^H)$ ) and  $\det(I_r - XY) = \det(I_n - YX)$  [20, Exercise 1.14] for  $n \times r$  matrices  $X = V^H$  and  $Y = C^{-1}U$ . For  $r = 1$ ,  $U = \mathbf{u}$ , and  $V = \mathbf{v}$ , (1) turns into the equation  $\det A = (1 - \mathbf{v}^H C^{-1} \mathbf{u}) \det C$  (cf. [16] and [20]).

#### 4.2 The extension to the full rank matrices

Let us prove the following extension of the SMW formula

$$A^+ = C^+ + C^+U(I_r - V^HC^+U)^{-1}V^HC^+ \quad (2)$$

to the case of rectangular matrices  $A$ ,  $U$ ,  $V$ , and  $C = A + UV^H$  of sizes  $m \times n$ ,  $m \times r$ ,  $n \times r$ , and  $m \times n$ , respectively, having full ranks.

*Proof.* For  $m \geq n$  observe that the matrix  $I_n - C^+UV^H$  is nonsingular and  $A = C(I_n - C^+UV^H)$ ,  $A^+ = (I_n - C^+UV^H)^{-1}C^+$ , substitute  $C \leftarrow I_m$  and  $V^H \leftarrow V^HC^+$  into the SMW formula in Theorem 4.1 to obtain that

$$(I_n - C^+UV^H)^{-1} = I_n + C^+U(I_r - V^HC^+U)^{-1}V^H, \quad (3)$$

combine the two latter equations, and arrive at (2). Likewise if  $m \leq n$  observe that the matrix  $I_m - UV^HC^+$  is nonsingular and  $A = (I_m - UV^HC^+)C$ ,  $A^+ = C^+(I_m - UV^HC^+)^{-1}$  substitute  $C \leftarrow I_n$  and  $U \leftarrow C^+U$  into the SMW formula to obtain that

$$(I_m - UV^HC^+)^{-1} = I_m + U(I_r - V^HC^+U)^{-1}V^HC^+, \quad (4)$$

combine the two latter equations, and again arrive at (2).

#### 4.3 Schur Aggregation

Observe that  $G = I_r - V^HC^+U$  is the Schur complement of the block  $C$  in the matrix  $\begin{pmatrix} C & V^H \\ U & I_r \end{pmatrix}$ . In Section 7 we carry over numerical problems of computations with an ill conditioned matrix  $A$  to the computation and inversion of the Schur complement  $G$  of a smaller size provided the matrix  $C$  is well conditioned.

The SMW formula reduces the solution of an ill conditioned linear system  $A\mathbf{y} = \mathbf{b}$  to well conditioned computations, apart from computing and inverting the Schur aggregates  $G$  of smaller sizes, and we arrive at a new instance in the general class of *aggregation methods*. They successively a) aggregate an input  $\mathbb{I}$  into an input  $\mathbb{I}_1$  of a smaller size, b) compute the solution for a given task, but for the input  $\mathbb{I}_1$ , and c) disaggregate the solution  $Y_1$  producing the solution  $Y$  for the original input  $\mathbb{I}$ . In our case  $\mathbb{I} = A$ ,  $\mathbb{I}_1 = G$ ,  $Y_1 = G^{-1}$ , and  $Y = A^{-1}$ . This motivates our nomenclature of *Schur Aggregation* for our approach.

Aggregation methods for solving linear systems of equations are well known (see, e.g., the hierarchical aggregation in [23], which has served as the springboard for the *Algebraic Multigrid*), but our present novelty is the link to A-preconditioning.

#### 4.4 The SMW formulae for solving linear systems

To yield solution  $Y = A^+U$  to a matrix equation  $AY = B$ , we can successively compute the matrices  $W = C^+U$ ,  $V^HW$ ,  $G = I - V^HW$ ,  $G^{-1}$ ,  $Z = G^{-1}V^HW$ ,  $Y = W + WZ$ . We can simplify the computation a little by choosing an APC  $UV^H$  such that  $UF = B$  for a matrix  $F$ . Then the SMW formula (2) implies that

$$Y = C^+UG^{-1}F, \quad UF = B \quad (5)$$

where  $C = A + UV^H$  and  $G = I - V^HC^+U$ . Indeed, post-multiply equation (2) by  $B = UF$ , obtain that

$$Y = A^+B = A^+UF = C^+UF + C^+UG^{-1}V^HC^+UF = C^+U(I_r + G^{-1}V^HC^+U)F,$$

and substitute the equation  $I_r + G^{-1}V^HC^+U = G^{-1}$ . If  $B$ ,  $F$ , and  $Y$  are vectors, we denote them by  $\mathbf{b}$ ,  $\mathbf{f}$ , and  $\mathbf{y}$ , respectively, and obtain that

$$\mathbf{y} = C^+UG^{-1}\mathbf{f}, \quad U\mathbf{f} = \mathbf{b}. \quad (6)$$

If furthermore  $F = 1$  and  $U = B = \mathbf{u} = \mathbf{b}$  is a vector, then  $g$  is a scalar, and (5) turns into the following vector equation,

$$Y = \mathbf{y} = \frac{1}{g}C^+\mathbf{b}, \quad g = 1 - \mathbf{v}^HC^+\mathbf{u}. \quad (7)$$

#### 4.5 The dual SMW formula

Assume that the matrices  $A$ ,  $U$ ,  $V$ , and  $C_- = A^+ + VU^H$  have full rank, write  $q = \text{rank}(VU^H)$ , and deduce the *dual SMW formula* (cf. Remark 7.1)

$$(C_-)^+ = (A^+ + VU^H)^+ = A - AVH^{-1}U^HA, \quad H = I_q + U^HAV. \quad (8)$$

It expresses the matrix  $(C_-)^+$  via the inverse of the matrix  $H$ , which is the Schur complement of the block  $-A^+$  in the block matrix  $\begin{pmatrix} -A^+ & U^H \\ V & I_q \end{pmatrix}$ . Due

to the equation  $((C_-)^+)^+ = A^+ + VU^H$ , we can express the solution  $\mathbf{y}$  to the linear system  $A\mathbf{y} = \mathbf{b}$  as follows,

$$\mathbf{y} = \mathbf{z} - VU^H\mathbf{b}, \quad (C_-)^+\mathbf{z} = \mathbf{b}. \quad (9)$$

*Proof of formula (8).* For  $m \geq n$  observe that the matrix  $I_n + VU^H A$  is nonsingular and  $(C_-)^+ = A(I_n + VU^H A)^{-1}$ , apply the SMW formula to obtain that  $(I_n + VU^H A)^{-1} = I_n - V(I_q + U^H AV)^{-1}U^H A$ , combine the two latter equations, and obtain (8). Likewise for  $m \geq n$  observe that the matrix  $I_m + AVU^H$  is nonsingular and  $(C_-)^+ = (I_m + AVU^H)^{-1}A$ , apply the SMW formula to obtain that  $(I_m + AVU^H)^{-1} = I_m - AV(I_q + U^H AV)^{-1}U^H$ , combine the two latter equations, and again arrive at (8).

## 5 Matrix computations with Schur Aggregation

Let us specify some algorithms based on the SMW formulae in Sections 4.2 and 4.4 (see also Remark 5.1).

### Algorithm 1. Schur Aggregation for determinants and inverses.

INPUT: two integers  $n$  and  $r$ ,  $n > r > 0$ , and a nonsingular  $n \times n$  matrix  $A$ .

OUTPUT: FAILURE or the scalar  $\det A$  and the matrix  $A^{-1}$  if  $\det A \neq 0$ .

COMPUTATIONS:

1. Generate the pair of normalized  $n \times r$  matrices  $U$  and  $\tilde{V}$  such that  $\|U\| = \|\tilde{V}\| = 1$ .
2. Compute a crude estimate  $\nu$  for the norm  $\|A\|$ .
3. Compute the matrix  $V = \nu\tilde{V}$ .
4. Compute the  $n \times n$  matrix  $C = A + UV^H$  and its determinant  $\det C$ . Output FAILURE if  $\det C = 0$ . Otherwise compute the inverse  $C^{-1}$ .
5. Compute the  $r \times r$  matrix  $G = I_r - V^H C^{-1}U$  and its determinant  $\det G$ . Output FAILURE if  $\det G = 0$ . Otherwise compute the inverse  $G^{-1}$ .
6. Compute and output the  $n \times n$  matrix  $A^{-1} = C^{-1} + C^{-1}UG^{-1}V^H C^{-1}$  and the scalar  $\det A = (\det C) \det G$  and stop.

Correctness of the algorithm follows from Theorem 4.1. Failure of this algorithm (as well as the next one) is highly unlikely if the entries of the generators  $U$  and  $V$  are randomly sampled from a large set independently of each other (cf. [12], [46], [52]).

If we only seek the inverse  $A^{-1}$ , we can omit the computation of the determinants  $\det C$  and  $\det G$  at Stages 4 and 5 and test nonsingularity of the matrices  $C$  and  $G$  instead. If we only seek the determinant  $\det A$ , we can omit the computation of the inverses  $G^{-1}$  and  $A^{-1}$  at Stages 5 and 6. If we only seek a solution  $\mathbf{y}$  to a linear system of equations  $A\mathbf{y} = \mathbf{b}$ , then besides performing matrix multiplications we only solve the linear systems  $CW = U$ ,  $C\mathbf{z} = \mathbf{b}$ , and  $G\mathbf{x} = V^H\mathbf{z}$ , but we can further simplify the computations as follows, based on the SMW formulae (6) and (7).

## Algorithm 2. Schur Aggregation for solving linear systems.

INPUT: two integers  $n$  and  $r$ ,  $n > r > 0$ , a nonsingular  $n \times n$  matrix  $A$ , and a vector  $\mathbf{b}$  of dimension  $n$ .

OUTPUT: FAILURE or the vector  $\mathbf{y} = A^{-1}\mathbf{b}$ .

COMPUTATIONS:

1. Generate the pair of normalized  $n \times r$  matrices  $U$  and  $\tilde{V}$  such that  $\|U\| = \|\tilde{V}\| = 1$  and  $\mathbf{b} = U\mathbf{e}_1$  where  $\mathbf{e}_1$  denotes the first coordinate vector of dimension  $r$ ,  $\mathbf{e}_1 = (1, 0, \dots, 0)^T$ .
2. Compute a crude estimate  $\nu$  for the norm  $\|A\|$ .
3. Compute the matrix  $V = \nu\tilde{V}$ .
4. Compute the matrices  $C = A + UV^H$ ,  $W = C^{-1}U$ , and  $G = I_r - V^H W$ . (Output FAILURE if at least one of the matrices  $C$  or  $G$  is singular.)
5. Compute the solution  $\mathbf{x}$  to the linear system  $G\mathbf{x} = \mathbf{e}_1$ .
6. Compute and output the vector  $\mathbf{y} = W\mathbf{x}$  and stop.

Correctness of the algorithm follows from equation (6).

If  $r = 1$ , then  $g$  is a scalar  $1 - \mathbf{v}^H C^{-1} \mathbf{e}_1$  and we can replace both Stages 5 and 6 with the following simpler stage (cf. (7)),

5. Compute and output the vector  $\mathbf{y} = \frac{1}{g} C^{-1} \mathbf{b}$  and stop.

**Remark 5.1.** At the end of Section 3 we discussed the selection of the ranks and the norms of our weakly random APCs. This process involved estimation of the norms  $\|C^+\|$  for recursively updated A-modifications  $C = A + UV^H$ . The SMW formula enables rapid updating of the matrix  $C^+$  (by using  $O(hmn)$  flops for updating  $C_+$  defined by a rank- $h$  updating of the matrix  $C$ ), and this cost estimate also holds for updating QR factorization of the matrix  $C$  (cf. [18, Section 12.5]).

## 6 Rounding errors

Assume that Algorithm 1 has been implemented numerically with rounding to a fixed precision. We can readily estimate the output error norm by combining the known bounds on rounding errors of matrix multiplication and solving linear systems of equations (see such bounds in [19, Section 7.1, page 121; Theorems 8.5, 9.3, 9.4, 10.3, 10.5, 10.6, 19.4, 19.5, 19.13, and 20.1, and Sections 20.2–20.4]). The estimates vary depending on the algorithm used, but consistently imply bounds on the relative error norm of the output that are roughly proportional to  $u \operatorname{cond} C$  where  $u$  is the unit roundoff.

More precisely, let  $\Delta(M) = \text{fl}(M) - M$  denote the error matrix in floating-point computation of a matrix  $M$  with rounding to a fixed precision (e.g., the IEEE standard double precision). Assume that the matrices  $A$ ,  $U$ , and  $V$  have been scaled so that  $\|A\| = \|U\| = \|V\| = 1$  and therefore  $\|C\| \leq 2$ . Further assume that the matrices  $A$  and  $C$  are nonsingular and that  $\Delta(C) = 0$ , that is, ignore the smaller errors in computing the matrix  $C$ . Write  $\kappa_- = \|C^{-1}\|$ . To simplify the estimates write  $c_n$  for the values depending on the dimension  $n$ , but independent of the input matrix  $C$ .

By combining the cited estimates for matrix multiplication and solving linear systems of equations we obtain that  $\|\Delta(C^{-1})\| \leq c_n u \kappa_-$ ,  $\|\Delta(G)\| \leq c_n u \kappa_-$ ,  $\|\Delta(G^{-1})\| \leq c_n u \|G^{-1}\|$ , and thus, due to the SMW formula,  $\|\Delta(A^{-1})\| \leq c_n u \kappa_- (1 + \kappa_- \|G^{-1}\|)$ . We can decrease the value  $u$  and therefore decrease the output error norm bound by increasing the precision of computing or by applying algorithms that emulate such an increase (see Sections 8 and 9).

## 7 The norm and conditioning of Schur aggregates

In the case of ill conditioned input  $A$  and well conditioned A-modification  $C$ , the Schur aggregate  $G$  can be ill conditioned, but it has a smaller size and is likely to be better conditioned than the input matrix  $A$ . Recursive application of our approach to such aggregates ultimately reduces the original task to well conditioned matrix computations, so that the numerical problems due to ill conditioning of the input  $A$  are confined to the stages of computing the aggregates. Our next results specify these observations quantitatively by linking to each other the singular values (and therefore the norms and condition numbers) of the matrices  $A$ ,  $C$  and  $G$ , involved into the SMW formula (2).

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

**Theorem 7.1.** 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 we have  $\sigma_j(M)\sigma_-(W) \leq \sigma_j(MW) \leq \sigma_j(M)\sigma_+(W)$  and  $\sigma_j(N)\sigma_-(W) \leq \sigma_j(WN) \leq \sigma_j(N)\sigma_+(W)$  for  $j = 1, \dots, \rho$  and  $\rho \times \rho$  matrices  $M$  and  $N$ .

*Proof.* The singular values are invariant in multiplication by a unitary matrix, and so we can consider just the case of a positive diagonal matrix  $W$ . In this case the claimed bounds follow from the Courant–Fischer Minimax Theorem [18, Theorems 8.1.2 and 8.6.1], [49, Theorem 3.3.2].  $\square$

The next simple result follows from the Courant–Fischer Minimax Theorem as

well as from the Wielandt–Hoffman theorem [18, Corollary 8.6.2 and Theorem 8.6.4] and is also a special case of [49, Theorem 3.3.3] for  $E = I_n$ .

**Theorem 7.2.** We have  $\sigma_j(W) - 1 \leq \sigma_j(W + I_n) \leq \sigma_j(W) + 1$  for an  $n \times n$  matrix  $W$  and for  $j = 1, 2, \dots, n$ .

Now we are ready to deduce our main estimates of this section.

**Theorem 7.3.** For positive integers  $m, n$ , and  $r$ , a  $m \times n$  matrix  $A$ , and a pair of unitary matrices  $U$  of size  $m \times r$  and  $V$  of size  $n \times r$ , write  $C = A + UV^H$  and  $G = I_r - V^H C^+ U$ . Suppose the matrices  $A$  and  $C = A + UV^H$  have full rank  $\rho \geq r$ . Then the matrix  $G$  is nonsingular, and we have

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

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

*Proof.* Let  $m \geq n$  and observe that the matrix  $G_n = I_n - C^+ UV^H$  is nonsingular. So is the matrix  $G$  as well because  $\det G = \det G_n$  [20, Exercise 1.14].

Now combine the equation  $A^+ = G_n^{-1} C^+$  with Theorem 7.1 for  $M = G_n^{-1}$ ,  $W = C^+$ , and  $A^+ = MW$  to obtain that

$$\sigma_j(G_n^{-1}) \sigma_-(C^+) \leq \sigma_j(A^+) \leq \sigma_j(G_n^{-1}) \sigma_+(C^+)$$

for  $j = 1, \dots, \rho$ . Substitute  $\sigma_-(C^+) = 1/\sigma_+(C)$  and  $\sigma_+(C^+) = 1/\sigma_-(C)$  and obtain that

$$\sigma_j(A^+) \sigma_-(C) \leq \sigma_j(G_n^{-1}) \leq \sigma_j(A^+) \sigma_+(C) \text{ for } j = 1, \dots, \rho. \quad (10)$$

Combine Theorem 7.1 for  $W = C^+ U$  and  $N = G^{-1}$  with the equations  $\sigma_j(C^+ U G^{-1} V^H) = \sigma_j(C^+ U G^{-1})$  for  $j = 1, \dots, r$ ,  $\sigma_-(C^+ U) = \sigma_-(C^+) = 1/\sigma_+(C)$ , and  $\sigma_+(C^+ U) = \sigma_-(C^+) = 1/\sigma_-(C)$  to deduce that

$$\sigma_j(G^{-1})/\sigma_+(C) \leq \sigma_j(C^+ U G^{-1} V^H) \leq \sigma_j(G^{-1})/\sigma_-(C)$$

for  $j = 1, \dots, r$ . Combine the latter bounds with Theorem 7.2 for  $W = C^+ U G^{-1} V^H$  and equation (3) to deduce that

$$\sigma_j(G^{-1})/\sigma_+(C) - 1 \leq \sigma_j(G_n^{-1}) \leq \sigma_j(G^{-1})/\sigma_-(C) + 1$$

and therefore

$$(\sigma_j(G_n^{-1}) - 1) \sigma_-(C) \leq \sigma_j(G^{-1}) \leq (\sigma_j(G_n^{-1}) + 1) \sigma_+(C)$$

for  $j = 1, \dots, r$ . Combine this equation with equation (10) and obtain the claimed bounds in the case of  $m \geq n$ .

For  $m \leq n$  proceed similarly, but replace  $G_n$  with  $G_m = I_m - UV^H C^+$ , use the equation  $A^+ = C^+ G_m^{-1}$  instead of  $A^+ = G_n^{-1} C^+$ , use equation (4) instead of (3), and furthermore, invoking Theorem 7.1 the first and the second time, replace  $M = G_n^{-1}$  with  $N = G_m^{-1}$  and replace  $W = C^+ U$  with  $W = V^H C^+$ , respectively.  $\square$

**Corollary 7.1.** a) Under the assumption of Theorem 7.3 we have

$$\|G\| = \sigma_1(G) = 1/\sigma_r(G^{-1}) \leq \nu_+(G) = 1/(\sigma_r(A^+) \sigma_-^2(C) - \sigma_-(C)),$$

$$\text{cond } G = \text{cond}(G^{-1}) \leq \kappa_+(G) = \frac{\sigma_1(A^+) \sigma_+(C) + 1}{\sigma_r(A^+) \sigma_-(C) - 1} \text{cond } C.$$

b) Write  $\theta_- = \sigma_r(A^+) \sigma_-(C) = \sigma_-(C) / \sigma_{\rho-r+1}(A)$ ,

$$\theta_+ = \sigma_1(A^+) \sigma_+(C) = \sigma_+(C) / \sigma_\rho(A), \quad \text{so that } \theta_+ \geq \theta_-.$$

$$\text{Then } \nu_+(G) = \frac{1}{(\theta_- - 1) \sigma_-(C)}, \quad \kappa_+(G) = \frac{\theta_+ + 1}{\theta_- - 1} \text{cond } C.$$

Furthermore if  $\theta_- \geq 2$ , then  $\theta_+ + 1 \leq 1.5 \theta_+$ ,  $\theta_- - 1 \geq 0.5 \theta_-$ , and therefore

$$\nu_+(G) \leq \frac{2\sigma_{\rho-r+1}(A)}{\sigma_-^2(C)}, \quad \kappa_+(G) \leq 3 \frac{\sigma_{\rho-r+1}(A)}{\sigma_\rho(A)} \text{cond}^2 C.$$

Let us summarize the estimates in the theorem, the corollary, and the papers [35] and [36]. Assume an  $n \times n$  normalized nonsingular ill conditioned matrix  $A$  and a properly scaled weakly random APC  $UV^H$  of rank  $r < n$  and deduce that

- (1) the matrix  $C$  is expected to be well conditioned if and only if the ratio  $\frac{\sigma_1(A)}{\sigma_{n-r}(A)}$  is not large,
- (2) if  $\sigma_n(C)$  is not small (that is if the matrix  $C$  is well conditioned), whereas  $\sigma_n(C) \gg \sigma_{n-r+1}(A)$ , then the matrix  $G$  is expected to have a small norm,
- (3) if the matrix  $C$  is well conditioned and if the ratio  $\frac{\sigma_{n-r+1}(A)}{\sigma_n(A)}$  is not large, then the matrix  $G$  is expected to be well conditioned.

If the matrix  $G = I_r - V^H C^{-1} U$  has a small norm, then in its computation the leading (most significant) bits in the representation of its diagonal entries

must be canceled, and so the computation requires high accuracy (see the next section).

**Remark 7.1.** It is attractive to apply the dual SMW formula (8) because it involves the solution of linear systems of equations only with the matrices  $H$  of smaller sizes. In this case, however, proper scaling of the generator matrices  $U$  and  $V$  involves the norm  $\|A^+\|$  rather than  $\|A\|$ . For some inputs  $A$  estimates for the norms  $\|A^+\|$  can be given to us or can be readily computed at a lower cost (very crude estimates would suffice), but generally such estimation can be about as hard as our original matrix computations. By extending our study in this section, we observe that for a large class of ill conditioned input matrices  $A$  and well conditioned and dual A-modifications  $C_-$ , the dual Schur aggregates  $H = I_q + U^H A V$  in equation (8) have small norms  $\|H\|$ . Can this observation help us to find proper scaling for our dual APCs?

## 8 High accuracy via numerical computations with double precision (an outline)

The algorithms in [10], [19], [21], [25], [38], [45], and the bibliography therein rapidly compute the sums and products with any fixed precision by operating with double precision numbers. This supports all our computations that are based on recursive application of the SMW formula, except for the solution of some well conditioned linear systems of equations.

Solving them we take advantage of having well conditioned inputs and apply iterative refinement (cf. [18, Section 3.5.3], [19, Chapter 11], and [48, Sections 3.3.4 and 3.4.5]). Every refinement loop consists essentially in multiplication of two pairs of  $n \times n$  by  $n \times r$  matrices and of an  $r \times n$  matrix by an  $n \times r$  matrix and performing  $(n+r)r$  additional ops (see Algorithm 3 in the next section). Such a loop requires  $O(rn^2)$  ops and produces about  $d = p - \log_2 \text{cond } M$  new correct bits per an output entry, where  $p$  is the precision of computing and  $M$  is the  $n \times n$  coefficient matrix [18, Section 3.5], [19, Chapter 12], [48, Sections 3.3.4 and 3.4.5]. For well conditioned matrices  $M$  the incremental value  $d$  is large enough and the progress is rapid.

The overall number of the  $p$ -precision ops in the entire refinement process is of the order  $rn^2 p_{out}/d$ , which for smaller ranks  $r$  makes it an attractive alternative to direct solution of the original ill conditioned linear system  $A\mathbf{y} = \mathbf{b}$ .

Let us further examine application of iterative refinement to computing the  $r \times r$  Schur aggregate  $G = I_r - V^H C^{-1} U$ , where we have cancellation of the leading bits in the binary representation of some output values (see the previous section). In this case we must extend iterative refinement beyond the

usual goal of obtaining the solution vector with double precision. We compute this vector with extended precision, as a sum of double precision pieces, but we avoid storing all these pieces. Instead we compute the double precision pieces that represent the aggregate  $G$ .

The number of stored entries decreases by the factor of  $n/r$  versus the computation of the  $n \times r$  matrix  $W = C^{-1}U$ , but the memory space for the storage decreases further because we begin storing the bits of the entries of the matrix  $G$  only when they stabilize at some nonzero values. If the norm  $\|G\|$  is small, such a stabilization first occurs deep into the refinement process (so that the previously computed double precision pieces would have occupied a larger memory, but we avoid their storage). This is typically the case where the matrix  $A$  is ill conditioned and the matrices  $C$  and  $G$  are not (in virtue of Theorem 7.3 and Corollary 7.1). We specify and analyze the extended iterative refinement in the next section.

## 9 Extended iterative refinement

In its classical form iterative refinement stops where the matrix  $W = C^{-1}U$  is computed with at most double precision. This can be insufficient in our case. Then we continue the steps of iterative refinement in the fashion of Hensel's lifting in [8] and [22] to improve the approximation further. As in the latter symbolic algorithm, we represent the output values as the sums of fixed-precision numbers.

Let us next specify and analyze our extended iterative refinement.

### Algorithm 3. Extended Iterative Refinement.

INPUT: two integers  $n$  and  $r$  such that  $n > r > 0$ , a nonsingular  $n \times n$  well conditioned matrix  $C$ , its approximate inverse  $X$  (see Remark 9.1), a pair of  $n \times r$  matrices  $U$  and  $V$ , and a stopping criterion (see Remark 9.2).

OUTPUT: FAILURE or an  $r \times r$  matrix  $\tilde{G} \approx G = I_r - V^H C^{-1}U$  satisfying the stopping criterion.

INITIALIZATION: Choose a sufficiently large integer  $\nu$  and set  $i \leftarrow 0$ ,  $U_0 \leftarrow U$ , and  $G_0 \leftarrow I_r$ .

COMPUTATIONS:

1. If  $i > \nu$ , output FAILURE and stop. Otherwise compute the matrices  $W_i \leftarrow XU_i$ ,  $U_{i+1} \leftarrow U_i - CW_i$ ,  $F_i \leftarrow V^T W_i$ , and  $G_{i+1} \leftarrow G_i - F_i$ .
2. If the fixed stopping criterion is satisfied, then output the matrix  $\tilde{G} = G_{i+1}$  and stop. Otherwise set  $i \leftarrow i + 1$  and go to Stage 1.

**Remark 9.1.** To approximate the inverse of the matrix  $C$  under the desired

norm bound, we can apply any direct or iterative algorithm such as Gaussian elimination, possibly combined with the classical numerical iterative refinement or Newton's iteration in [30, Chapter 6], [37], [41] (also cf. Section 12.2). Alternatively we can compute the matrix  $X$  implicitly, e.g., via the factorization of the matrix  $C$  that would enable us to compute the matrix  $XU_i$  in  $O(rn^2)$  ops for a given  $n \times r$  matrix  $U_i$ .

**Remark 9.2.** We can stop the algorithm as soon as the computed matrix  $G_{i+1}$  is stabilized versus  $G_i$  so that  $\|G_{i+1} - G_i\|_l < t\|G_{i+1}\|_l$  for a fixed  $l$  equal to 1, 2, or  $\infty$  and a fixed positive tolerance  $t$ .

For comparison, the classical algorithm begins with a crude approximation  $W_0 \approx W = C^{-1}U$  and recursively computes the matrices  $U_i \leftarrow U - CW_{i-1}$ ,  $E_i \leftarrow C^{-1}U_i$ , and  $W_i \leftarrow W_{i-1} + E_i$  for  $i = 0, 1, \dots, k$ , so that the norm  $\|W_i - W\|$  recursively decreases until it reaches the limit posed by rounding errors.

Theorem 7.3 defines a small upper bound on the norm  $\|G\|$  if  $A$  is an ill conditioned matrix and if the matrices  $C$  and  $G$  are well conditioned. Therefore, we can have  $G_i \approx 0$  for  $i = 0, 1, \dots, k$  and some positive integer  $k$ . At the  $i$ th step of iterative refinement for  $i \leq k$  we need to store only the most recently computed matrix  $G_{i+1}$  overwriting  $G_i$ , and similarly we can overwrite the matrices  $W_{i-1}$ ,  $U_i$ , and  $F_{i-1}$  with their updates  $W_i$ ,  $U_{i+1}$ , and  $F_i$ , to save the memory space.

At the stages of computing the matrices  $C \leftarrow A + UV^T$ ,  $U_{i+1} \leftarrow U_i - CW_i$ ,  $F_i \leftarrow -V^TW_i$ , and  $G_{i+1} \leftarrow G_i + F_i$  for  $i = 0, 1, \dots, k$ , we seek the error-free output because even small relative errors can completely corrupt the matrix  $G$ . To meet the challenge, we compute the sums and products error-free.

We require that in each iteration the matrices  $XW_i \approx C^{-1}U_i$  be computed within an error norm bound that ensures the desired decrease of the residual norms  $u_i = \|U_i\|$  by a fixed factor  $\theta < 1$  (cf. Theorem 9.2 for  $\theta = \max_i \theta_i$ ). Respectively the error norm  $e_i = \|E_i\|$  decreases since  $E_i = C^{-1}U_i$ .

Within the allowed perturbation norm, we vary the matrices  $U$ ,  $V$ ,  $C^{-1}$ , and  $W_i$  for all  $i$  to decrease the number of bits in the binary representation of their entries. We first estimate from above the norm of the input perturbation and the precision of computing that together keep the output error norm within the fixed tolerance bound. Then we perturb the input within the estimated norm bound to represent it with fewer bits. We can just round every entry to fewer bits or we can first set to zero the absolutely smaller input entries. Finally we perform iterative refinement and verify that it converges as expected. Otherwise we correct our policy of input perturbation.

## Estimates for the errors and the parameter $\theta$

**Theorem 9.1.** Consider the subiteration

$$\begin{aligned} W_i &\leftarrow \text{fl}(C^{-1}U_i) = C^{-1}U_i - E_i \\ U_{i+1} &\leftarrow U_i - CW_i \end{aligned}$$

for  $i = 0, 1, \dots, k$  and  $U = U_0$ . Then

$$C(W_0 + \dots + W_k) = U - CE_k.$$

*Proof.* Due to the assumed equations, we have  $CW_i = U_i - U_{i+1}$ ,  $i = 0, 1, \dots, k-1$ . Sum the latter equations to obtain that  $C(W_0 + \dots + W_{k-1}) = U_0 - U_k$ . Substitute the equations  $U_0 = U$  and  $U_k = CW_k + CE_k$  and obtain the theorem.  $\square$

The theorem implies that the sum  $W_0 + \dots + W_k$  approximates the matrix  $W = C^{-1}U$  with the error matrix  $-E_k$ .

It remains to show that the error term  $E_i$  converges to zero as  $i \rightarrow \infty$ .

**Theorem 9.2.** Assume that

$$W_i = (C - \tilde{E}_i)^{-1}U_i = C^{-1}U_i - E_i \quad \text{for all } i.$$

Write  $e_i = \|E_i\|$ ,  $u_i = \|U_i\|$ , and  $\theta_i = \delta_i \|C\|$  where

$$\delta_i = \delta(C, \tilde{E}_i) = 2\|\tilde{E}_i\|_F \max\{\|C^{-1}\|^2, \|(C - \tilde{E}_i)^{-1}\|^2\}.$$

Then we have  $e_i \leq \delta_i u_i$  for all  $i$ ,  $e_{i+1} \leq \theta_i e_i$  and  $u_{i+1} \leq \theta_i u_i$  for  $i = 0, 1, \dots, k-1$ .

*Proof.* We begin with some auxiliary results.

**Theorem 9.3.** We have  $U_{i+1} = CE_i$  and consequently  $u_{i+1} \leq e_i \|C\|$  for all  $i$ .

*Proof.* Pre-multiply the matrix equation  $C^{-1}U_i - W_i = E_i$  by  $C$  and add the resulting equation to the equation  $U_{i+1} - U_i + CW_i = 0$ .  $\square$

**Lemma 9.1.** Let  $C$  and  $C + E$  be two nonsingular matrices. Then

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

*Proof.* See [18, Section 5.5.5].  $\square$

**Corollary 9.1.** Assume that  $W_i = (C - \tilde{E}_i)^{-1}U_i = C^{-1}U_i - E_i$ . Then  $e_i \leq \delta_i u_i$  where

$$\delta_i = \delta(C, \tilde{E}_i) = 2\|\tilde{E}_i\|_F \max\{\|C^{-1}\|^2, \|(C - \tilde{E}_i)^{-1}\|^2\}.$$

Combine Theorem 9.3 and Corollary 9.1 and obtain that  $u_{i+1} \leq \theta_i u_i$  and  $e_{i+1} \leq \theta_i e_i$  for  $\theta_i = \delta_i \|C\|$  and for all  $i$ . Summarize our estimates and obtain Theorem 9.2. □ □

The theorem shows linear convergence of the error norms  $e_i$  to zero as  $i \rightarrow \infty$  provided  $\theta = \max_i \theta_i < 1$ . This implies linear convergence of the matrices  $W_0 + \dots + W_i$  to  $W$ ,  $U_0 + \dots + U_i$  to  $U$ ,  $F_0 + \dots + F_i$  to  $F$ , and  $G_{i+1}$  to  $G$ . The theorem also shows local quadratic convergence, that is doubling the number of correct bits in every step. We enjoy such a rapid progress, however, only until this number reaches the working precision of computing, and then we shift back to performing linearly convergent computations with this precision (see Corollary 9.3).

Let us next estimate the values  $\theta_i$ . We assume dealing with a well conditioned matrix  $C$ , and so the ratios  $r_i = \|\tilde{E}_i\|_F / \|C\|_F$  are small and  $\text{cond}(C - \tilde{E}_i) \approx \text{cond } C$  (cf. [18, Section 3.3], [19], [48, Theorem 3.4.9]). In this case the values

$$\begin{aligned} \theta_i &= \delta_i \|C\| \\ &= 2r_i \max\{\text{cond}^2 C, \text{cond}^2(C - E_i)\} \|C\|_F / \|C\| \\ &\approx 2(\text{cond } C)^2 r_i \|C\|_F / \|C\| \\ &\leq 2(\text{cond } C)^2 r_i n \end{aligned}$$

tend to be significantly less than one.

**Remark 9.3.** In view of the latter estimates, the smaller  $\text{cond } C$ , the faster convergence of Algorithm 3. If the value  $\text{cond } C$  is not as small as we wish, we can apply multiplicative preconditioning  $C \rightarrow XC$  or  $C \rightarrow CX$  where  $X \approx C^{-1}$ , so that  $C^{-1} \approx (XC)^{-1}X$  and  $C^{-1} \approx X(CX)^{-1}$ .

### Precision bounds

Finally we estimate the precision required in our error-free computation of the residual matrices  $U_i$ . Hereafter for a binary number  $b = \sigma \sum_{k=t}^s b_k 2^k$ , where  $\sigma = 1$  or  $\sigma = -1$  and each  $b_k$  is zero or one, we write  $t(b) = t$ ,  $s(b) = s = \lfloor \log_2 |b| \rfloor$ , and  $p(b) = s - t + 1$ , so that  $p(b)$  is the precision in the binary representation of  $b$ . For an  $n \times n$  matrix  $M = (m_{i,j})_{i,j}$  we write  $s(M) = \max_{i,j} s(m_{i,j})$ ,  $t(M) = \min_{i,j} t(m_{i,j})$ ,  $p(M) = s(M) - t(M) + 1$ . Then

$$\log_2(n\|M\|) \leq s(M) \leq \lfloor \log_2 \|M\| \rfloor, \quad (11)$$

and the absolute value of each entry of the matrix  $M$  is the sum of some powers  $2^k$  for integers  $k$  selected in the range  $[t(M), s(M)]$ .

**Lemma 9.2.** We have  $t(U_{i+1}) \geq \min\{t(U_i), t(CW_i)\}$  for all  $i$ . Moreover  $t(CW_i) \geq t(W_i)$  if the (scaled) matrix  $C$  is filled with integers.

*Proof.* The lemma follows from the equations  $U_{i+1} = U_i - CW_i$ .  $\square$

**Lemma 9.3.** We have  $s(U_{i+1}) \leq s(U_i) + \log_2(\theta_i n)$  for all  $i$ .

*Proof.* The lemma follows from the bounds  $u_{i+1} \leq \theta_i u_i$  and (11).  $\square$

**Lemma 9.4.** We have  $s(U_{i+1}) \leq s(CW_i) + \log_2 f_i$  and  $s(U_{i+1}) \leq s(W_i) + \log_2(f_i \|C\|)$  for  $\theta_i < 1$ ,  $f_i = \frac{\theta_i n}{|1-\theta_i|}$ , and all  $i$ .

*Proof.* First recall that  $u_{i+1} \leq \theta_i u_i$ , so that  $|u_i - u_{i+1}| \geq |\frac{1}{\theta_i} - 1| u_{i+1}$ . The equation  $U_i - U_{i+1} = CW_i$  implies that  $\|CW_i\| = \|U_i - U_{i+1}\| \geq |u_i - u_{i+1}| \geq |\frac{1}{\theta_i} - 1| u_{i+1}$ . Therefore  $u_{i+1} \leq \frac{f_i}{n} \|CW_i\| \leq \frac{f_i \|C\|}{n} \|W_i\|$ . Combine these bounds with bound (11) for  $M = U_{i+1}$ ,  $M = CW_i$  and  $M = W_i$ .  $\square$

**Corollary 9.2.**

- a) If  $t(U_{i+1}) \geq t(U_i)$ ,  
then  $p(U_{i+1}) \leq p(U_i) + \log_2(\theta_i n)$ .
- b) If  $t(U_{i+1}) \geq t(CW_i)$ ,  
then  $p(U_{i+1}) \leq p(CW_i) + \log_2 f_i$ .
- c) If  $t(U_{i+1}) \geq t(W_i)$ ,  
then  $p(U_{i+1}) \leq p(W_i) + \log_2(f_i \|C\|)$ .

Recall that in virtue of Lemma 9.2, at least one of assumptions a) and b) is always satisfied, and if the matrix  $C$  is filled with integers, then so is one of assumptions a) and c) as well.

**Corollary 9.3.** Assume the precision bound  $p(W_i) \leq \hat{p}$  or  $p(CW_i) \leq \tilde{p}$  for an integer  $\hat{p}$  or  $\tilde{p}$ , respectively. Let this bound imply that  $\theta_i \leq \frac{1}{n}$  for all  $i$ . (In this case we have convergence with global linear rate for the iterative refinement in Theorem 9.1.) Then we have the uniform bound  $\hat{p} + \log_2 \frac{n}{n-1}$  on the precision  $p(U_{i+1})$  of the representation of all matrices  $U_{i+1}$  for all  $i$ . If the matrix  $C$  is filled with integers, then we also have the bound  $\tilde{p} + \log_2(\frac{n}{n-1} \|C\|)$ .

Unless we know some estimates for  $\theta_i$  for all  $i$ , we cannot say a priori for which minimum precision bounds  $\hat{p}$  and  $\tilde{p}$  we can ensure the progress in iterative refinement, but we can find these bounds dynamically, by first performing the computations with the IEEE standard double precision and then (if needed) increasing it recursively until convergence is observed. The cited fast advanced algorithms for sums and products can handle any precision growth, but in our

tests the growth was limited. We used the double precision for  $W_i$  and regularly observed that  $s(U_{i+1}) < s(W_i) + \log_2 n$ , which was in line with Lemma 9.4.

## 10 Matrix structure in the Schur Aggregation

Sparse and structured matrices can be multiplied by a vector fast, and so for sparse and structured linear systems the Conjugate Gradient algorithms are highly attractive as long as they converge fast. This is known to be the case for well conditioned sparse or structured inputs  $A$ . For ill conditioned sparse or structured inputs  $A$ , we shift to well conditioned matrices  $C = A + UV^H$  choosing the generator matrices  $U$  and  $V$  sparse or structured. Then we can multiply the matrices  $C$  by vectors fast, and the Conjugate Gradient algorithms rapidly compute the matrices  $C^+U$  (or  $V^HC^+$ ) and  $G = I_r - V^HC^+U$ .

In the case of a matrix  $A$  with displacement structure and APCs of larger ranks, we can substantially simplify the subsequent computations by endowing the generators  $U$  and  $V$  with consistent structure (see [30, Chapters 1 and 4] and the bibliography therein). These structures are preserved in the inverses and, with slow deterioration, in the sums and products of pairs of matrices with consistent structures. Therefore, we can extend the consistent structure of an input matrix  $A$  and its APC to the A-modification  $C$  and the Schur aggregates  $G$ . We refer the readers to [35, Examples 4.1–4.6] and [36, Examples 1-6] on APCs with some most frequently used matrix structures, in particular of Toeplitz and Hankel types as well as sparse APCs.

These examples do not include APCs with the popular and important structures of Vandermonde and Cauchy types, but if the input matrix  $A$  has such structure, we can multiply it by appropriate Vandermonde multipliers to transform it into a matrix with the structure of Toeplitz or Hankel type, and then we can readily apply our APCs. This is a particular application of the general *method of displacement transformation*, due to [27] (see its exposition also in [30, Sections 1.7, 4.8, and 4.9]).

## 11 Numerical tests

We tested our Algorithm 1 for computing determinants, which incorporated high accuracy summation and multiplication in [38] and [7], respectively, and which solved linear systems as by-product. For comparison we also applied the Matlab Subroutine **det** to the same input.

We generated the input matrices  $A = PML$  by following [44]. Here  $P$  were permutation matrices, each swapping  $k$  random pairs of the rows of the matrix  $A$ , whereas  $L$  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  $[-\eta, \eta]$  for a fixed positive  $\eta$ . It followed that  $\det A = (-1)^k$ . We generated such matrices for  $\eta = 5,000$ ,  $n = 4, 8, 16, 32, 64$ ,  $k = 2n$  and  $k = 2n - 1$ .

We first generated a random candidate APCs  $UV^T$  of rank one, and then recursively increased the rank until we arrived at a well conditioned A-modification  $C = A + UV^T$ . More precisely, we generated two random  $n \times r$  unitary matrices  $U_0$  and  $V_0$ , then truncated their entries to represent them with the precision of 20 bits, denoted the resulting matrices  $\tilde{U}$  and  $\tilde{V}$ , and computed the APC  $\hat{U}\hat{V}^T = 2^q\tilde{U}\tilde{V}^T$  and the A-modification  $\tilde{C} = A + \hat{U}\hat{V}^T$  for an integer  $q$  such that  $\frac{1}{2} < \frac{\|\hat{U}\hat{V}^T\|}{\|A\|} \leq 2$ . If  $\text{cond } \tilde{C}$  was small enough, we accepted the matrix  $\hat{U}\hat{V}^T$  as the desired APC  $UV^T$ . Otherwise we regenerated APC in the same way. If this has not produced a desired APC, we recomputed it according to the following recipe from [40], [35], and [43] (see our Section 3),

$$(U \leftarrow Q(C^{-1}U), \quad V^T \leftarrow Q(V^TC^{-1})).$$

If this did not help either, we incremented  $r$  by one and repeated the computations. We encountered overflows and underflows for larger  $n$ , but overcame the problems by simultaneously scaling the matrix  $U$  by factor  $2^k$  and the matrix  $V$  by factor  $2^{-k}$  for an appropriate integer  $k$  and by temporarily scaling the matrices  $I_r$  and  $U$  by the same factor  $2^h$  for an appropriate integer  $h$ .

The selected matrices  $A$  were ill conditioned for all integers  $n$  in our range (with  $\text{cond } A$  quite steadily in the range from  $10^{17}$  to  $10^{25}$  for all  $n$ ) and turned out to be hard inputs for the numerical Subroutine **det** in Matlab. Already for  $n = 4$  and  $\eta = 5,000$ , the Matlab's numerical outputs had wrong sign in over 45% out of 100,000 runs and were off from the true value of  $\det A$  by the factor of two or more in over 90% of the runs. Like this subroutine, our algorithms also relied on the double precision computations, but have output the correct sign of the determinant and have approximated its value with relative errors within 0.001 in all our runs for  $n = 4, 8, 16, 32, 64$  and for the same value of  $\eta$ .

We expect that the algorithm that supports the Subroutine **det** could have produced correct outputs if we performed its order of  $n^3$  ops with a sufficiently high precision  $p_{comp} \geq p_{output} + \log_2 \text{cond } A$ , but to our advantage we yield the correct output by performing  $O(\frac{rn^2p}{p_{double} - \log_2 \text{cond } C})$  ops with double precision  $p_{double}$ , and this is dramatically faster in the present day computer environment.

## 12 Discussion

### 12.1 Preconditioning by expansion

Here is a modification of our approach where the generator  $V$  is simplified at the expense of a small increase of the input size. Instead of adding a preconditioner  $P = UV^H$  to the input matrix  $A$ , we expand this matrix according to the following maps,

$$A \rightarrow M = \begin{pmatrix} -\theta I_r & B^H \\ 0 & A \end{pmatrix} \rightarrow C = M + UV^H = \begin{pmatrix} -\theta I_r & B^H \\ F & A \end{pmatrix},$$

where  $U = \begin{pmatrix} 0 \\ F \end{pmatrix}$  and  $V = (I_r, 0)$ .

We choose the scalar  $\theta$  and the matrices  $B$  and  $F$  such that the ratios  $\frac{\theta}{\|A\|}$ ,  $\frac{\|B\|}{\|A\|}$ , and  $\frac{\|F\|}{\|A\|}$  are neither large nor small.  $C$  is a Hermitian matrix for  $F = B$ .

We observe that  $\text{cond } A = \text{cond} \begin{pmatrix} 0 & 0 \\ 0 & A \end{pmatrix}$ , and then we estimate  $\text{cond } C$  by

extending the analysis in [35], [36] to the map  $\begin{pmatrix} 0 & 0 \\ 0 & A \end{pmatrix} \rightarrow C = \begin{pmatrix} -\theta I_r & B^H \\ F & A \end{pmatrix}$ .

We obtain that for weakly random matrices  $B$  and  $F$  (or for a weakly random matrix  $B = F$ ) we can expect that matrix  $C$  is well conditioned provided the integer  $r$  is equal to or exceeds the number of the singular values of the matrix  $A$  that are small relative to the 2-norm  $\|A\| = \|A\|_2$ . The dimension of the input increases from  $n$  to  $n + r$ , but multiplications with the matrix  $V^H = (I_r, 0)^H$  require no ops.

The Sherman–Morrison–Woodbury formula in Theorem 4.1 expresses the inverse matrix  $M^{-1}$  via the inverses  $C^{-1}$  and  $G^{-1}$ , and we also have  $\det M =$

$(\det C) \det G$ , whereas  $(-\theta)^r \det A = \det M$ ,  $M^{-1} = \begin{pmatrix} -\frac{1}{\theta} I_r & B^H A^{-1} \\ 0 & A^{-1} \end{pmatrix}$ , and

the solution  $\mathbf{y}$  of a linear system  $A\mathbf{y} = \mathbf{b}$  is the projection of the vector  $\mathbf{z} = M^{-1} \begin{pmatrix} \mathbf{0} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$  onto the subvector made up of the last  $n$  coordinates

of this vector  $\mathbf{z}$ . Therefore, we can apply Algorithms 1 and 2 to the matrix  $M$  (rather than  $A$ ). Then we can recover  $A^{-1}$  from  $M^{-1}$  and  $\mathbf{y}$  from  $\mathbf{z}$  by

using no ops and recover  $\det A$  from  $\det M$  by using just a single multiplication. The treatment of the input  $M$  with our APC  $UV^H$  is simplified because  $V = (I_r, 0)$ .

### 12.2 Weakly randomized multiplicative preconditioning versus pivoting

Experiments performed by Guoliang Qian in the Graduate Center of the the City University of New York indicate that Gaussian elimination without pivoting is quite safe for random matrices. Namely he observed the following relative residual norms  $\|A\mathbf{y} - \mathbf{b}\|/\|\mathbf{b}\|$  in his 1000 test runs for Gaussian elimination (with and without pivoting) applied to the linear systems  $A\mathbf{y} = \mathbf{b}$  for matrices  $A$  of the size  $128 \times 128$  and vectors  $\mathbf{b}$  of dimension 128, both filled with random integers from the range  $[-10^4, 10^4]$ .

Class	with no pivoting	with pivoting ( Matlab routine)
min	1.776999149990865e-012	2.104228576466744e-013
average	2.984974763052528e-010	2.104228576466744e-013
max	1.941634938870705e-008	2.001174233358663e-012

This suggests using multiplicative preconditioning  $A \rightarrow MAN$  with weakly random matrices  $M$  and  $N$  as a tentative alternative to pivoting. Recall that pivoting counters the problems of numerical stability in the process of Gaussian elimination and LU factorization, but "usually degrades the performance" [18, page 119]). One can choose matrices  $M$  and  $N$  structured (say, to be random circulant matrices or just the matrices of sine or cosine transforms) to simplify preconditioning.

**Acknowledgement.** The expert comments of the reviewers were most valuable for improving our presentation.

### References

- [1] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, England, 1994.
- [2] 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.
- [3] M. Benzi, Preconditioning Techniques for Large Linear Systems: a Survey, *J. of Computational Physics*, **182**, 418–477, 2002.

- [4] H. Brönnimann, I. Z. Emiris, V. Y. Pan, S. Pion, Sign Determination in Residue Number Systems, *Theoretical Computer Science*, **210**, **1**, 173–197, 1999.
- [5] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.
- [6] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, Cambridge, England, 2005.
- [7] T. J. Dekker, A Floating-point Technique for Extending the Available Precision, *Numerische Mathematik*, **18**, 224–242, 1971.
- [8] J. D. Dixon, Exact Solution of Linear Equations Using  $p$ -adic Expansions, *Numerische Math.*, **40**, 137–141, 1982.
- [9] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
- [10] J. Demmel, Y. Hida, Accurate and Efficient Floating Point Summation, *SIAM J. on Scientific Computing*, **25**, 1214–1248, 2003.
- [11] J. Demmel and Y. Hida, Fast and Accurate Floating Point Summation with Application to Computational Geometry, *Numerical Algorithms*, **37**, 101–112, 2004.
- [12] R. A. Demillo, R. J. Lipton, A Probabilistic Remark on Algebraic Program Testing, *Information Processing Letters*, **7**, **4**, 193–195, 1978.
- [13] W. Eberly, M. Giesbrecht, G. Villard, On Computing the Determinant and Smith Form of an Integer Matrix, *Proc. 41st Annual Symp. on Foundations of Computer Science (FOCS'2000)*, 675–685, IEEE Computer Society Press, Los Alamitos, CA, 2000.
- [14] I. Z. Emiris, V. Y. Pan, Symbolic and Numerical Methods for Exploiting Structure in Constructing Resultant Matrices, *J. of Symbolic Computation*, **33**, 393–413, 2002.
- [15] I. Z. Emiris, V. Y. Pan, Improved Algorithms for Computing Determinants and Resultants, *J. of Complexity*, **21**, **1**, 43–71, 2005.
- [16] G. H. Golub, Some Modified Matrix Eigenvalue Problems, *SIAM Review*, **15**, 318–334, 1973.
- [17] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.
- [18] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 3rd edition, The Johns Hopkins University Press, Baltimore, Maryland, 1996.
- [19] N. J. Higham, *Accuracy and Stability in Numerical Analysis*, SIAM, Philadelphia, 2002 (second edition).
- [20] A. S. Householder, *The Theory of Matrices in Numerical Analysis*, Dover, New York, 1964.

- [21] 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.
- [22] R. T. Moenck, J. H. Carter, Approximate Algorithms to Derive Exact Solutions to Systems of Linear Equations, *Proceedings of EUROSAM, Lecture Notes in Computer Science*, **72**, 63–73, Springer, Berlin, 1979.
- [23] W. L. Miranker, V. Y. Pan, Methods of Aggregations, *Linear Algebra and Its Applications*, **29**, 231–257, 1980.
- [24] B. Mourrain, V. Y. Pan, Multivariate Polynomials, Duality and Structured Matrices, *J. of Complexity*, **16**, **1**, 110–180, 2000.
- [25] T. Ogita, S. M. Rump, S. Oishi, Accurate Sum and Dot Product, *SIAM Journal on Scientific Computing*, **26**, **6**, 1955–1988, 2005.
- [26] V. Y. Pan, How Can We Speed up Matrix Multiplication? *SIAM Review*, **26**, **3**, 393–415, 1984.
- [27] V. Y. Pan, On Computations with Dense Structured Matrices, *Math. of Computation*, **55**, **191**, 179–190, 1990.
- [28] V. Y. Pan, Complexity of Computations with Matrices and Polynomials, *SIAM Review*, **34**, **2**, 225–262, 1992.
- [29] V. Y. Pan, Solving a Polynomial Equation: Some History and Recent Progress, *SIAM Review*, **39**, **2**, 187–220, 1997.
- [30] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.
- [31] V. Y. Pan, Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *Journal of Symbolic Computations*, **33**, **5**, 701–733, 2002.
- [32] 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. Also available at <http://comet.lehman.cuny.edu/vpan/>
- [33] 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)*, **55**, **8**, 1870–1886, 2008.
- [34] 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.

- [35] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, Tech. Report TR 2008004, *Ph.D. Program in Computer Science, Graduate Center, the City University of New York*, 2008. Available at <http://www.cs.gc.cuny.edu/tr/techreport.php?id=352>
- [36] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, *Proc. of the Third International Computer Science Symposium in Russia (CSR 2008), Lecture Notes in Computer Science (LNCS)*, **5010**, 372–383, 2008.
- [37] V. Y. Pan, M. Kunin, R. Rosholt, H. Kodal, Homotopic Residual Correction Processes, *Math. of Computation*, **75**, 345–368, 2006.
- [38] V. Y. Pan, B. Murphy, G. Qian, R. E. Rosholt, Error-free Computations via Floating-Point Operations, *Computers and Math. (with Applications)*, in press, and Technical Report TR 2007010, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2007. Available at <http://www.cs.gc.cuny.edu/tr/techreport.php?id=352>
- [39] V. Y. Pan, B. Murphy, R. E. Rosholt, M. Tabanjeh, The Schur Aggregation for Solving Linear Systems of Equations, *Proceedings of the Third International Workshop on Symbolic–Numeric Computation (SNC 2007)*, 142–151, July 2007, London, Ontario, Canada, (Jan Verschelde and Stephen Watt eds.), ACM Press, New York, 2007.
- [40] V. Y. Pan, G. Qian, Solving Homogeneous Linear Systems with Weakly Random Additive Preprocessing, Technical Report TR 2008009, *CUNY Ph.D. Program in Computer Science, Graduate Center, the City University of New York*, 2008. Available at <http://www.cs.gc.cuny.edu/tr/techreport.php?id=352>
- [41] V. Y. Pan, R. Schreiber, An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications, *SIAM J. on Scientific and Statistical Computing*, **12**, 5, 1109–1131, 1991.
- [42] V. Y. Pan, X. Yan, Additive Preconditioning, Eigenspaces, and the Inverse Iteration, submitted to *Linear Algebra and Its Applications*. Also Technical Report TR 2008006, *CUNY Ph.D. Program in Computer Science, Graduate Center, the City University of New York*, 2008. Available at <http://www.cs.gc.cuny.edu/tr/techreport.php?id=352>
- [43] V. Y. Pan, X. Yan, Null Space and Eigenspace Computations with Additive Preprocessing, *Proceedings of the Third International Workshop on Symbolic–Numeric Computation (SNC 2007)*, 152–160, July 2007, London, Ontario, Canada, (Jan Verschelde and Stephen Watt eds.), ACM Press, New York, 2007.
- [44] V. Y. Pan, Y. Yu, Certification of Numerical Computation of the Sign of the Determinant of a Matrix, *Algorithmica*, **30**, 708–724, 2001.
- [45] 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* (41 pages), 2005, and *SIAM Journal on Scientific Computing*, in print.

- [46] J. T. Schwartz, Fast Probabilistic Algorithms for Verification of Polynomial Identities, *Journal of ACM*, **27**, **4**, 701–717, 1980.
- [47] J. Shevchuk, Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates, *Discrete and Computational Geometry*, **18**, 305–363, 1997, available at [www.cs.cmu.edu/quake/robust.html](http://www.cs.cmu.edu/quake/robust.html)
- [48] G. W. Stewart, *Matrix Algorithms, Vol I: Basic Decompositions*, SIAM, Philadelphia, 1998.
- [49] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, 1998 (first edition), 2001 (second edition).
- [50] A. Storjohann, The Shifted Number System for Fast Linear Algebra on Integer Matrices, *J. of Complexity*, **21**(4), 609–650, 2005.
- [51] X. Wang, Affect of Small Rank Modification on the Condition Number of a Matrix, *Computer and Math. (with Applications)*, **54**, 819–825, 2007.
- [52] R. E. Zippel, Probabilistic Algorithms for Sparse Polynomials, *Proceedings of EUROSAM'79, Lecture Notes in Computer Science*, **72**, 216–226, Springer, Berlin, 1979.