2007

TR-2007014: The Schur Aggregation and Extended Iterative Refinement

V. Y. Pan

B. Murphy

R. E. Rosholt

The Schur Aggregation and Extended Iterative Refinement *

V. Y. Pan[a], B. Murphy, R. E. Rosholt
Department of Mathematics and Computer Science
Lehman College, City University of New York
Bronx, NY 10468, USA
[victor.pan] [brian.murphy] [rhys.rosholt]
@lehman.cuny.edu
[a]http://comet.lehman.cuny.edu/vpan/

Abstract

According to our previous theoretical and experimental study, additive preconditioners can be readily computed for ill conditioned matrices, but application of such preconditioners to facilitating matrix computations, in particular to solving linear systems of equations, is not straightforward. In the present paper we develop some nontrivial techniques for the latter task. By applying the Sherman–Morrison–Woodbury formula and its new variations, we confine the original numerical problems to the computation of the Schur aggregates of smaller sizes. Then we overcome these problems by extending the Wilkinson's iterative refinement and applying some advanced semi-symbolic algorithms for multiplication and summation. In particular with these techniques we control precision throughout our computations.

1 Introduction

Our point of departure is *additive preconditioning* in [34]–[38], [40], [41], [44], that is selecting an *additive preconditioner* P (hereafter APC) and mapping an ill conditioned input matrix A into its better conditioned *additive modification* $C = A + P$. Hereafter we write "A-" for "additive" and "APC" for "additive preconditioner".

*Supported by PSC CUNY Awards 66437-0035, 67297–0036 and 68291–0037. Some results of this paper have been presented at the International Conferences on the Matrix Methods and Operator Equations in Moscow, Russia, in June of 2005, 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.

We observe the three following advantages of A-preconditioning over the customary multiplicative preconditioning.

- APCs are readily available for a large class of matrices
- We can readily extend the structure and sparseness of an input matrix to APCs
- A-preconditioning has a wider range of applications, which include eigensolving, the solution of singular and nonsingular linear systems of equations, and the computation of determinants.

According to the theoretical and extensive experimental study in [36]–[41] and [38], one can readily generate a random APC for a given ill conditioned matrix A. In the present paper we use such an APC to facilitate the solution of a linear system of equations $A\mathbf{y} = \mathbf{b}$. This involves some advanced techniques such as modification of the *SMW inversion formula* (by Sherman, Morrison, and Woodbury), extension of Wilkinson's *iterative refinement*, and algorithms for error-free multiplication and summation, for which we use the abbreviation *MSAs*.

We organize our presentation as follows. In the next section we demonstrate our approach by recursively applying rank-one modifications. In Section 3 we introduce basic definitions. In Section 4 we cover the SMW formula and its new variations. In Section 5 we link the singular values of the input matrix and of the auxiliary matrices involved in our computations. In Section 6 we further improve our basic approach of Section 2. In Section 7 we extend iterative refinement. In Section 8 we comment on preserving matrix structure in our computations and Section 9 on MSAs. Our numerical tests have confirmed the predicted performance of our algorithms. The tests have been performed jointly by all authors. Otherwise the paper is due to the first author.

2 Solving a linear system of equations with recursive rank-one modifications

Hereafter 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 [1], [4], [17], [19], [46], [47], e.g., $\mathbf{v}$ is a vector, I_k denotes the $k \times k$ identity matrix, I is I_k for an unspecified k, $\sigma_j(A)$ is the j-th largest singular value of a matrix A of a rank ρ for $j = 1, \ldots, \rho$, $||A|| = \sigma_1(A)$, and $\mathrm{cond}\, A = \sigma_1(A)/\sigma_\rho(A)$ is the condition number of a matrix A. A matrix A is ill conditioned if this number is large and is well conditioned otherwise. "Ops" is our abbreviation for "arithmetic operations".

According to the cited study in [35]–[38], A-preconditioning with a random sparse and/or structured and well conditioned APP P of a rank r is likely to decrease the condition number of an $n \times n$ ill conditioned matrix A to the level

of the ratio $\sigma_1(A)/\sigma_{n-r}(A)$ provided the ratio $||P||/||A||$ is neither large nor small.

Now consider a nonsingular but ill conditioned linear system of n equations with n unknowns, $A\mathbf{y} = \mathbf{b}$, where the ratio $\sigma_1(A)/\sigma_{n-1}(A)$ is not large, whereas $\sigma_{n-1}(A) \gg \sigma_n(A)$. Suppose we have a rank-one APC $P = \mathbf{u}\mathbf{v}^H$ and a well conditioned A-modification $C = A + \mathbf{u}\mathbf{v}^H$. Apply the SMW inversion formula in our Theorem 4.1 in the case of $U = \mathbf{u}$ and $V = \mathbf{v}$ and obtain that

$$A^{-1} = C^{-1} + C^{-1}\mathbf{u}\mathbf{v}^H C^{-1}/g \quad \text{for} \quad g = 1 - \mathbf{v}^H C^{-1}\mathbf{u}.$$

This reduces the solution to well conditioned computations, apart from computing the value g. We arrive at a new instance in the general class of *aggregation methods*. They successively a) aggregate an input I into a smaller input I_1, b) compute the solution for a given task but for the input I_1, and c) disaggregate the solution Y_1 producing the solution Y for the original input I. In our case $I = A$, $I_1 = g$, $Y_1 = 1/g$, and $Y = A^{-1}$. The value $g = 1 - \mathbf{v}^H C^{-1}\mathbf{u}$ is the Gauss transform of the 2×2 block matrix

$$\begin{pmatrix} C & \mathbf{u} \\ \mathbf{v}^H & 1 \end{pmatrix}$$

and the Schur complement of its block C [17, pages 95 and 103]. We call this value a *Schur aggregate* and call the above methods the *(primal) Schur Aggregation*.

Aggregation methods for solving linear systems of equations are well known (see, e.g., the ones in [28], which have served as the springboard for the *Algebraic Multigrid*), but our novelty is the link to A-preconditioning.

The value g is absolutely small in virtue of our Theorem 5.3 (because $\mathrm{cond}_2 A$ is large, whereas $\mathrm{cond}_2 C$ is not) and thus must be computed within a small absolute error. To ensure this, we apply MSAs throughout and extend Wilkinson's iterative refinement when we compute the vectors $C^{-1}\mathbf{b}$ and $C^{-1}\mathbf{u}$ or $C^{-H}\mathbf{v}$ (see Section 7).

The computation of the matrix C requires two matrix-by-vector multiplications and a single matrix addition, that is $5n^2 - 2n$ ops. The computation of the vectors $C^{-1}\mathbf{u}$ and $C^{-1}\mathbf{b}$ by means of Gaussian elimination takes $(2/3)n^3 + O(n^2)$ ops. The subsequent transition to the solution vector $\mathbf{y}$ requires $O(n)$ ops.

Next suppose that both ratios $\sigma_1(A)/\sigma_{n-1}(A)$ and $\sigma_{n-1}(A)/\sigma_n(A)$ are large. Then $\mathrm{cond}\, C$ is likely to be of the order of the former ratio, that is, is likely to satisfy $1 \ll \mathrm{cond}\, C \ll \mathrm{cond}\, A$.

We can apply our A-preconditioning and aggregation to the ill (although better) conditioned linear systems $C\mathbf{z} = \mathbf{u}$ and $C\mathbf{w} = \mathbf{b}$ and continue the process recursively until we arrive at a well conditioned matrix. This is expected to occur in r recursive steps provided the ratio $\sigma_1(A)/\sigma_{n-\rho+1}(A)$ is large but the ratio $\sigma_1(A)/\sigma_{n-\rho}(A)$ is not large. The concept "large" is quantified depending on the context and computer environment (like the customary concepts "well" and "ill conditioned"). We call such an integer r *numerical nullity* of the matrix A

and write $r = \operatorname{nnul} A$, complementing the numerical rank $\operatorname{nrank} A = n - \operatorname{nnul} A$. Overall the r recursive steps require $(2/3)n^3 + 5rn(n+r) + O(rn)$ ops.

In this paper we elaborate upon the above techniques and their modifications. This study is naturally extended to other matrix computations in [34], [41], and [44].

3 Basic definitions

Here are our basic definitions in addition to the ones in the previous sections.

A matrix A is *normalized* if $||A|| = 1$ and is unitary if $A^H A = I$.

A matrix A of a rank ρ has the Frobenius norm $||A||_F^2 = \operatorname{trace}(A^H A) = \sum_{j=1}^{\rho} \sigma_j^2(A)$ such that $||A|| \le ||A||_F \le \sqrt{\rho}||A||$.

Hereafter we use the abbreviation "*SVD*" for "Singular Value Decomposition". The *compact SVD* of an $m \times n$ matrix A of a rank ρ 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, $S^{(\rho)H} S^{(\rho)} = I_\rho$, $T^{(\rho)H} T^{(\rho)} = I_\rho$, $\Sigma^{(\rho)} = \operatorname{diag}(\sigma_j)_{j=1}^{\rho}$ 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 $j = 1, \ldots, \rho$ are the singular values of the matrix A, $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_\rho > 0$.

The *Moore-Penrose generalized inverse* of an $m \times n$ matrix A of a rank ρ (also called its *pseudo inverse*) is the matrix $A^- = \sum_{j=1}^{\rho} \sigma_j^{-1} \mathbf{t}_j \mathbf{s}_j^H$. We write A^- instead of the customary A^+ in [17], [46], [47], and we write A^{-H} for $(A^H)^- = (A^-)^H$.

We have $A^- = A^{-1}$ if $m = n = \rho$,

$$A^- = (A^H A)^{-1} A^H \quad \text{if} \quad m \ge n = \rho, \tag{3.1}$$

$$A^- = A^H (A A^H)^{-1} \quad \text{if} \quad m = \rho \le n, \tag{3.2}$$

$\operatorname{cond} A = \sigma_1/\sigma_\rho = ||A||\ ||A^-||$. It follows that

$$\operatorname{cond}(MN) \le (\operatorname{cond} M) \operatorname{cond} N. \tag{3.3}$$

Hereafter we represent our APCs as the products $P = UV^H$ of rectangular matrices U and V, thus emphasizing the role of the ranks of the APCs.

4 The SMW formula and its variations

4.1 The case of nonsingular matrices

For a 2×2 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 *block Gauss transform* of the matrix B and the *Schur complement* of its northwestern block B_{11} (respectively, southeastern block B_{22}) provided $B_{11}^{-}B_{11} = I$ and/or $B_{11}B_{11}^{-} = I$ (respectively, $B_{22}^{-}B_{22} = I$ and/or $B_{22}B_{22}^{-} = I$) [17, pages 95, 103], [46, page 155]. We immediately verify the following lemma.

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

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

for the same block decomposition of the matrix B and for some matrices W, X, Y and Z. Then $W = G_{11}^{-1}$ (resp. $Z = G_{22}^{-1}$) if the block B_{11} (resp. B_{22}) 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 (block Gauss transforms) of the blocks I_r and C in the matrix*

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

such that

$$\det W = \det A = (\det C) \det G. \tag{4.1}$$

Furthermore [17, page 50], [46, Corollary 4.3.2], if the matrix A is nonsingular, then so is the matrix G, and we have the Sherman–Morrison–Woodbury formula $(C - UV^H)^{-1} = C^{-1} + C^{-1}UG^{-1}V^H C^{-1}$.

Proof. Begin with the factorizations

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

which implies equations (4.1). Invert this factorization to obtain that

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

for some matrices X, Y, and Z. □

Remark 4.1. *Equation (4.1) 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* $\det(I_r - X^HY) = \det(I_n - YX^H)$ *[20, Exercise 1.14] for* $n \times r$ *matrices* $X = V^H$ *and* $Y = C^{-1}U$*. For* $r = 1$*,* $U = \mathbf{u}$*, and* $V = \mathbf{v}$*, (4.1) turns into the equation* $\det A = (1 - \mathbf{v}^HC^{-1}\mathbf{u})\det C$ *(cf. [11] and [20]).*

4.2 The SMW formula for the full rank matrices

Suppose the 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, have full ranks. For $m \le n$ deduce that the matrix $I_m - UV^HC^-$ is nonsingular and

$$A = (I_m - UV^HC^-)C,\ A^- = C^-(I_m - UV^HC^-)^{-1}, \tag{4.2}$$

whereas for $m \ge n$ deduce that the matrix $I_n - C^-UV^H$ is nonsingular and

$$A = C(I_n - C^-UV^H),\ A^- = (I_n - C^-UV^H)^{-1}C^- \tag{4.3}$$

(cf. equations (3.1) and (3.2)).

For $m \ge n$ substitute $C \leftarrow I_m$ and $V^H \leftarrow V^HC^-$ into the SMW formula in Theorem 4.1 and obtain that

$$(I_m - UV^HC^-)^{-1} = I_m + U(I_r - V^HC^-U)^{-1}V^HC^-. \tag{4.4}$$

For $m \le n$ substitute $C \leftarrow I_n$ and $U \leftarrow C^-U$ into the SMW formula and obtain that

$$(I_n - C^-UV^H)^{-1} = I_n + C^-U(I_r - V^HC^-U)^{-1}V^H. \tag{4.5}$$

By combining equations (4.2)–(4.5), extend the SMW formula to rectangular matrices of full rank as follows,

$$A^- = C^- + C^-U(I_r - V^HC^-U)^{-1}V^HC^-. \tag{4.6}$$

Observe that $A^-A = I_n$ for $m \ge n$, $AA^- = I_n$ for $m \le n$, and $G = I_r - V^HC^-U$ is the block Gauss transform of the block matrix

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

and the Schur complement of its block C. We call the matrix G a Schur aggregate and the transition to computations with this matrix the Schur Aggregation. Here is a simple flowchart for computing a Schur aggregate.

Flowchart 4.1. *Given a matrix* A *of full rank, generate an APP* UV^H *and successively compute the matrices*

- $C = A + UV^H$*, which should have full rank,*
- C^-U *or* V^HC^-*,*

- $G = I_r - V^H C^- U$.

For smaller ranks 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 [1], [17, Sections 10.2–10.4], [45], [49], but the conditioning of this matrix can become the central issue for larger ranks r.

Finally suppose we seek a solution Y of a matrix equation $AY = B$ and use an APC UV^H such that $U = BF$ for a matrix F. Then the SMW formula implies that

$$Y = C^- U G^{-1} F \tag{4.7}$$

where $C = A + UV^H$ and $G = I - V^H C^- U$. In particular if $U = B = \mathbf{u} = \mathbf{b}$ is a vector, then $F = 1$, g is a scalar, and

$$Y = \mathbf{y} = C^- \mathbf{b}/g. \tag{4.8}$$

4.3 The dual SMW formula

Assume that the matrices A, U, V, and $C_- = A^- + VU^H$ have full rank and deduce that the matrix $I_n + VU^H A$ is nonsingular and

$$(C_-)^- = A(I_n + VU^H A)^{-1} \quad \text{where} \quad m \geq n, \tag{4.9}$$

whereas the matrix $I_m + AVU^H$ is nonsingular and

$$(C_-)^- = (I_m + AVU^H)^{-1} A \quad \text{where} \quad m \leq n. \tag{4.10}$$

Write $q = \text{rank}(VU^H)$, apply the SMW formula, and obtain that

$$(I_n + VU^H A)^{-1} = I_n - V(I_q + U^H AV)^{-1} U^H A$$

for $m \geq n$ and

$$(I_m + AVU^H)^{-1} = I_m - AV(I_q + U^H AV)^{-1} U^H$$

for $m \leq n$. Substitute these equations into (4.9) and (4.10) and in both cases obtain the *dual SMW formula*

$$(C_-)^- = (A^- + VU^H)^- = A - AVH^{-1}U^H A, \quad H = I_q + U^H AV. \tag{4.11}$$

Equations (4.11) express the matrix $(C_-)^-$ via the inverse H^{-1} 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,

$$\mathbf{y} = \mathbf{z} - VU^H \mathbf{b}, \ (C_-)^- \mathbf{z} = \mathbf{b}. \tag{4.12}$$

For $q < \min\{m, n\}$ we call the matrix H the *dual Schur aggregate* and the transition to the computations with this matrix the *dual Schur Aggregation*.

5 The norm and conditioning of a Schur aggregate

In this section we link the singular values of the matrices A, C and G in the SMW formula (4.6), which implies further estimates for the norm and conditioning of the Schur aggregate G. In particular *the matrix G has a small norm and is well conditioned if* $\operatorname{rank}(UV^H) = \operatorname{nnul} A > 0$ *and the matrix C is well conditioned.*

We deduce from equation (4.7) for $F = I$ and $B = U$ that $A^-U = C^-UG^{-1}$ where A and C are $m \times n$ matrices of full rank and $m \geq n$. Then bound (3.3) and the equation $\operatorname{cond} M = \operatorname{cond}(M^-)$ together imply that $\operatorname{cond}(A^-U) \leq (\operatorname{cond} C)(\operatorname{cond} U)\operatorname{cond} G$. For random well conditioned $m \times r$ matrices U and larger r we can expect that the ratio $\operatorname{cond} A/\operatorname{cond}(A^-U)$ is not very large, and then, informally speaking, numerical problems of computing with matrix A are translated to the computations with the matrices C and G. Next we deduce a similar property in the case of APCs UV^H of any rank provided the matrix C is well conditioned.

First we estimate the jth singular values of the matrix G^{-1}, $j = 1, \ldots, r$, in terms of the singular values $\sigma_j(A^-)$, $\sigma_1(C)$, and $\sigma_1(C^-)$. Theorem 5.2 is a special case of [47, Theorem 3.3.3] where $E = I_n$.

Theorem 5.1. *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 $j = 1, \ldots, \rho$ 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 readily follow from the Courant–Fischer Minimax Characterization [17, Theorem 8.1.2], [47, Theorem 3.3.2]. □

Theorem 5.2. *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 $j = 1, 2, \ldots, n$.*

Theorem 5.3. *For positive integers m, n, and r, a normalized $m \times n$ matrix A, and a pair of matrices U of size $m \times r$ and V of size $n \times r$, write $C = A + UV^H$ and $G = I_r - V^HC^-U$. 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 $\sigma_-(C) = \sigma_\rho(C)$, $\sigma_+(C) = \sigma_1(C) \leq 2$, $\sigma_j(A^-) = 1/\sigma_{\rho-j+1}(A)$, $j = 1, \ldots, r$.

Proof. Let $m \geq n$. Deduce from equation (4.3) 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].

Next combine equation (4.3) with Theorem 5.1 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 $j = 1, \ldots, \rho$. Substitute $\sigma_-(C^-) = 1/\sigma_+(C)$ and $\sigma_+(C^-) = 1/\sigma_-(C)$ and obtain that

$$\sigma_j(A^-)\sigma_-(C) \le \sigma_j(G_n^{-1}) \le \sigma_j(A^-)\sigma_+(C) \text{ for } j = 1, \ldots, \rho. \tag{5.1}$$

Combine Theorem 5.1 for $W = C^-U$ and $N = G^{-1}$ with the equations and inequalities $\sigma_j(C^-UG^{-1}V^H) = \sigma_j(C^-UG^{-1})$ for $j = 1, \ldots, r$, $\sigma_-(C^-U) \ge \sigma_-(C^-) = 1/\sigma_+(C)$, and $\sigma_+(C^-U) \le \sigma_-(C^+) = 1/\sigma_-(C)$ to deduce that

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

for $j = 1, \ldots, r$. Combine the latter bounds with Theorem 5.2 for $W = C^-UG^{-1}V^H$ and equation (4.5) to deduce that

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

and therefore

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

for $j = 1, \ldots, r$. Combine this equation with equation (5.1) and obtain the claimed bounds in the case of $m \ge n$.

For $m \le n$ proceed similarly but use equations (4.2) and (4.4) instead of (4.3) and (4.5), replace G_n with $G_m = I_m - UV^HC^-$ and furthermore, invoking Theorem 5.1 the first and the second time, replace $M = G_n^{-1}$ with $N = G_m^{-1}$ and replace $W = C^-U$ with $W = V^HC^-$, respectively. □

Corollary 5.1. *Under the assumption of Theorem 5.3 we have*

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

$$||G|| = \sigma_1(G) = 1/\sigma_j(G^{-1}) \le 1/(\sigma_r(A^-)\sigma_-^2(C) - \sigma_-(C)).$$

Suppose A is an $n \times n$ nonsingular matrix such that $\operatorname{nnul} A = r$ and UV^H is a random, well conditioned and properly scaled APP of a rank r. Then the values $\sigma_{n-j+1}(A)/\sigma_1(A)$ are small for $j \le r$ and are not small for $j > r$, whereas the value $\sigma_n(C)$ is likely to be of the order of $\sigma_{n-r}(A) \gg \sigma_{n-r+1}(A)$. Therefore, all singular values $\sigma_{r-j+1}(G) = 1/\sigma_j(G^{-1})$ for $j = 1, \ldots, r$ are likely to be of the order of at most $\sigma_{n-j+1}(A)$. Furthermore (cf. Corollary 5.1), $\operatorname{cond} G$ is likely to be of the order of $(\operatorname{cond} C)^2\sigma_{n-r+1}(A)/\sigma_n(A)$, whereas the 2-norm $||G|| = \sigma_1(G)$ is likely to be of the order of $\sigma_{n-r+1}(A)$. The latter value has the order of $\sigma_1(A)/\operatorname{cond} A$. Thus, as we claimed, the matrix G is expected to have a small norm and to be well conditioned if $||A|| \ne 1$ and if $\operatorname{nnul} A = r$.

Finally all our estimates for matrices A, C, and G are readily extended to the dual counterparts A^-, $(C_-)^-$ and H of these matrices.

6 The solution of linear systems with the Schur Aggregation

Our study in the previous sections supports some variations of our recursive rank-one modifications in Section 2 for the solution of a linear system $A\mathbf{y} = \mathbf{b}$. Indeed we can generate

1. APCs based on the dual SMW formula
2. APCs of ranks $r > 1$
3. APCs UV^H where $U\mathbf{f} = \mathbf{b}$ for some vector $\mathbf{f}$ (cf. (4.8)).

1. Recursive application of dual rank-one modifications naturally mimics the recursive process in Section 2 but has an advantage of avoiding divisions and restricting matrix inversions to inverting a single dual A-modification at the last recusive step, where this A-modification is well conditioned. Indeed, we first apply formulae (4.11) and (4.12) for $U = \mathbf{u}$ and $V = \mathbf{v}$ and obtain that

$$h(C_-)^- = hA - A\mathbf{v}\mathbf{u}^H A = A(hI - \mathbf{v}\mathbf{u}^H A), \quad h = 1 - \mathbf{u}^H A\mathbf{v}, \qquad (6.1)$$

$\mathbf{y} = \mathbf{z} - \mathbf{u}^H\mathbf{b}$, and $(C_-)^-\mathbf{z} = \mathbf{b}$. These equations define division-free reduction of a linear system $\{A\mathbf{y} = \mathbf{b}\} \to \{h(C_-)^-\mathbf{z} = h\mathbf{b}\}$.

For a pair of random vectors $\mathbf{u}$ and $\mathbf{v}$ (as well as for a random vector $\mathbf{u}$ and for $\mathbf{v} = \mathbf{u}$) scaled so that the ratio $||\mathbf{v}\mathbf{u}^H||/||A^-||$ is neither large nor small, we can expect (cf. [38]) that $\mathrm{cond}(C_-) = \mathrm{cond}((C_-)^-)$ has the order of $\sigma_2(A)/\sigma_{n-1}(A)$. If this ratio is large, we can apply similar techniques of dual A-preconditioning and dual aggregation to the matrix $h_1(C_{-1})^- = h(C_-)^-$, producing a matrix $h_2(C_{-2})^-$ with the condition number expected to be at the level of $\sigma_3(A)/\sigma_n(A)$. Recursively we can expect to arrive at a well conditioned matrix $h_r(C_{-r})^-$ in r steps provided $\mathrm{nnul}(A^-) = r$.

Overall for this transition we need to multiply $2q$ matrices of the size $n \times n$ by $2q$ vectors and in addition to perform either qn^2 ops including divisions (cf. equation (4.11)) or $2qn^2$ ops division-free. We need $(2/3)n^3 + O(n^2)$ ops to solve the well conditioned linear system $h(C_{-r})^-\mathbf{z}_r = h\mathbf{b}$ by using Gaussian elimination. The subsequent transition to the solution vector $\mathbf{y}$ requires $O(qn)$ ops (cf. equation (4.12)).

Compared to the recursive process in Section 2 for $r = q$, the book-keeping for the back transition to the solution $\mathbf{y}$ is simplified, and we can save the order of q^2n ops at this stage, but we use extra qn^2 ops in a division-free version. Most important difference, however, is that we avoid numerical problem and do not need iterative refinement at the stage of computing the Schur aggregates G where the respective computation of the dual Schur aggregates H is division-free.

To support the dual recursive process we need a crude estimate for the value $\sigma_n(A)$, versus a crude estimate for $\sigma_1(A)$ in the recursive process in Section 2. The known numerically stable algorithms produce both estimates at the cost of $O(n^2)$ ops [17, Sections 2.3.2, 2.3.3, and 3.5.4], [46, Section 5.3].

We can combine q recursive steps of the above dual process with r recursive steps of the primal process in Section 2. This is likely to decrease the condition number of the input matrix A to the level of $\sigma_{q+1}(A)/\sigma_{n-r}(A)$.

2. Suppose A is a nonsingular ill conditioned $n \times n$ matrix such that the ratios σ_1/σ_{n-r+1} and σ_{n-r}/σ_n (resp. σ_1/σ_q and σ_{q+1}/σ_n) are not large, but $\sigma_{n-r} \gg \sigma_{n-r+1}$ (resp. $\sigma_q \gg \sigma_{q+1}$), so that the matrix A is ill conditioned due to a single jump in the spectrum of its singular values. We can find the threshold value r (resp. q) of the rank of the APC by recursively testing the values $0, 1, 2, 4, 8, \ldots$ until we arrive at a well conditioned matrix C (resp. an ill conditioned matrix H) and then applying binary search to decrease this value.

   Under this assuption we can effectively apply a primal APC UV^H of rank r (resp. a dual APC of rank q) instead of r primal (resp. q dual) rank-one recursive steps. Overall, at both A-preconditioning and aggregation stages, we perform about as many ops as in the case of recursive rank-one modifications, except that we need to perform about $2r^3$ (resp. $2q^3$) extra ops to invert the matrix G (resp. H). Moreover, in the dual case, additional care is needed to avoid divisions. These extra cost and effort, however, can be more than compensated by the well known benefits of applying block matrix multiplications [17], [46].

   The approach can be extended recursively. In this case at every recursive step, one ensures that the matrix G (resp. C_-) is well conditioned as long as one chooses not too small (resp. not too large) rank values r (resp. q). In this case numerical problems are confined to the matrix C (resp. H), which is computed division-free, and thus can be computed error-free with MSAs.

3. Equations (4.7) and (4.8) enable us to simplify the computation of the solution vector $\mathbf{y} = A^-\mathbf{b}$ to the linear system $A\mathbf{y} = \mathbf{b}$ versus the primal SMW formula. To incorporate these equations into the recursive process of A-preconditioning and aggregation, we choose the matrices $U = U_k$ and $F = F_k$ at the kth recursive step as follows, $U_1 = \mathbf{b}$, $F_1 = 1$, $U_k = (U_{k-1}, \mathbf{u_k})$, $F^T = (U_{k-1}, \mathbf{0})$, $k = 2, 3, \ldots$. Then the kth recursive step engages a new (random) vector $\mathbf{u}_k$ and outputs the matrix U_{k-1}. The progress with improving the conditioning is the same as before except that the impact of the first step with $U_1 = \mathbf{b}$ decreases (resp. becomes nil) wherever the vector $\mathbf{b}$ lies near (resp. in) the range of the matrix A.

7 Extended iterative refinement

Consider the computation of the Schur aggregate $G = I_r - V^T C^{-1} U$ where the input matrix A is ill conditioned, whereas its A-modification C is not. We rely on Flowchart 4.1 where we compute the matrix $W = C^{-1}U$ from the matrix equation $CW = U$.

Under our assumptions on the matrices A and C, Theorem 5.3 implies that the norm $||G||$ is small, and so the computation of every diagonal entry of the Schur aggregate G annihilates a number of its leading significant bits. Therefore we must compute these entries with a high precision, and so we apply MSAs in this computation and extend Wilkinson's iterative refinement when we compute the matrix $C^{-1}U$.

In its classical form the refinement stops where the matrix $W = C^{-1}U$ is computed with at most double precision. This is generally insufficient in our case. Thus we continue the steps of iterative refinement in the fashion of Hensel's lifting in [27], [5] to improve the approximation further. As in the latter symbolic algorithm, we represent the output values as the sums of fixed-precision numbers (cf. Section 9).

7.1 Extended iterative refinement (Outline)

Let us specify and analyze the extended iterative refinement of the matrices $W = \sum_{i=0}^{k} W_i$ and $G = I_r - V^T W = I_r + \sum_{i=1}^{k} F_i$. Fix a sufficiently large integer k, write $U_0 = U$ and $G_0 = I_r$, and successively compute 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 $i = 0, 1, \ldots, k$. (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 $i = 0, 1, \ldots, k$, so that the norm $||W_i - W||$ recursively decreases until it reaches the limit posed by rounding errors.) Here is a simple example for demonstration of our extension in the case where $n = 4$, $r = 1$, and the $r \times r$ Schur aggregate G turns into a scalar.

Example 7.1.

$$A = \begin{pmatrix} 63419461 & -29226193 & -41333003 & -8964 \\ -17439352 & -22167219 & -14775811 & -3204 \\ -38199953 & -59526299 & -19725060 & -4276 \\ -7074 & 3261 & 4611 & 1 \end{pmatrix}$$

$$U^T = \begin{pmatrix} 75776 & 258048 & 122880 & 118784 \end{pmatrix}$$

$$V^T = \begin{pmatrix} 128 & 148 & 72 & 148 \end{pmatrix}$$

$$C = \begin{pmatrix} 73118789 & -18011345 & -35877131 & 11205884 \\ 15590792 & 16023885 & 3803645 & 38187900 \\ -22471313 & -41340059 & -10877700 & 18181964 \\ 15197278 & 17583293 & 8557059 & 17580033 \end{pmatrix}$$

$$G_0 = 1$$

$$W_0^T = \begin{pmatrix} 0.00000000000008 \\ 0.00000000027075 \\ 0.00000146570198 \\ 0.00675746938214 \end{pmatrix}$$

$$G_1 = 2.190473991081632e - 008$$

$$W_1 = \begin{pmatrix} -0.00000000124834 \\ 0.00000001025780 \\ -0.00000886188400 \\ 0.14800929926118 \end{pmatrix} * 1.0e - 009$$

$$G_2 = 3.174438743663640e - 016$$

$$W_2 = \begin{pmatrix} 0.00000000716215 \\ 0.00000003837446 \\ -0.00002747063331 \\ 0.21450242917524 \end{pmatrix} * 1.0e - 017$$

$$G_3 = -7.918752906512810e - 024$$

$$W_3 = \begin{pmatrix} 0.00000002240260 \\ 0.00000002020945 \\ -0.00010131901406 \\ -0.53500152161636 \end{pmatrix} * 1.0e - 025$$

$$G_4 = -1.475542403337810e - 030$$

$$W_4 = \begin{pmatrix} 0.00000005462336 \\ -0.00000003648089 \\ 0.00018978679691 \\ -0.90511944620263 \end{pmatrix} * 1.0e - 033$$

$$G_5 = -1.341598391541817e - 030$$

$$W_5 = \begin{pmatrix} -0.00000002134640 \\ 0.00000000681126 \\ -0.00005379511619 \\ -0.12995555745450 \end{pmatrix} * 1.0e - 040$$

$$G_6 = -1.341598389618088e - 030$$

$$W_6 = \begin{pmatrix} -0.00000007190044 \\ 0.00000001308189 \\ -0.00004562685927 \\ -0.44933111982233 \end{pmatrix} * 1.0e - 048$$

$$G = G_7 = -1.341598389618088e - 030$$

Theorem 5.3 defines a small upper bound on the norm $||G||$ if A is an ill conditioned matrix and if the matrix C is well conditioned. Therefore, we can have $G_i \approx 0$ for $i = 0, 1, \ldots, k$ and some positive integer k. At the ith step of iterative refinement for $i \leq k$ we can 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 $C \leftarrow A + UV^T$, $U_{i+1} \leftarrow U_i - CW_i$, $F_i \leftarrow -V^T W_i$, and $G_{i+1} \leftarrow G_i + F_i$ for $i = 0, 1, \ldots, k$ we seek error-free output

because even small relative errors can completely corrupt the matrix G. To meet the challenge, we have two tools, namely, a) MSAs and b) the truncation of the entries of the matrices U, V, C, and W_i for all i.

We can choose any pair of matrices U and V up to a perturbation within a fixed small norm as long as this perturbation keeps the A-modification $C = A + UV^H$ well conditioned. Likewise, we require that the matrices C^{-1} and $W_i \leftarrow C^{-1}U_i$ be computed within an error norm bound that ensures the decrease of the residual norms $u_i = ||U_i||$ (and consequently the error norm $e_i = ||E_i||$ since $E_i = C^{-1}U_i$) by a fixed factor ϕ exceeding one in each iteration (cf. Corollary 7.2). For numerical inversion of the matrix C under the desired norm bound, we can apply any direct or iterative algorithm (e.g., Gaussian elimination, possibly combined with the classical numerical iterative refinement, or Newton's iteration in [33, Chapter 6], [39], [43]).

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 set the entries to zero wherever this is compatible with the above requirements to the matrices. Then we truncate the remaining (nonzero) entries to decrease the number of bits in their representation as much as possible under the same requirements to the matrices.

7.2 Estimates for the errors and the parameter

$\theta = 1/\phi$

Theorem 7.1. *Consider the subiteration*

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

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

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

Proof. Due to the assumed equations, we have $CW_i = U_i - U_{i+1}$, $i = 0, 1, \ldots, k-1$. Sum the latter equations to obtain that $C(W_0 + \cdots + W_{k-1}) = U_0 - U_k$. 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 $-E_k$.

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

Theorem 7.2. *Assume that*

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

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 $e_i \le \delta_i u_i$ *for all* i, $e_{i+1} \le \theta_i e_i$, $u_{i+1} \le \theta_i u_i$ *for* $i = 0, 1, \ldots, k-1$.

Proof. We follow [34, Section 8] and begin with some auxiliary results.

Theorem 7.3. *We have* $U_{i+1} = CE_i$ *and consequently* $u_{i+1} \le e_i||C||$ *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 $U_{i+1} - U_i + CW_i = 0$. □

Lemma 7.1. *Let* C *and* $C + E$ *be two nonsingular matrices. Then*

$$||(C+E)^{-1} - C^{-1}|| \le ||(C+E)^{-} - C^{-}||_F$$
$$\le 2||E||_F \max\{||C^{-1}||^2,\ ||(C+E)^{-1}||^2\}.$$

Proof. See [17, Section 5.5.5]. □

Corollary 7.1. *Assume that* $W_i = (C - \tilde{E}_i)^{-1}U_i = C^{-1}U_i - E_i$. *Then* $e_i \le \delta_i u_i$ *where*

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

Proof. Combine Theorem 7.3 and Corollary 7.1 and obtain that $u_{i+1} \le \theta_i u_i$ and $e_{i+1} \le \theta_i e_i$ for $\theta_i = \delta_i||C||$ and for all i. □

Summarize our estimates and obtain Theorem 7.2. □

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

Let us next estimate the values θ_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 $\mathrm{cond}(C - \tilde{E}_i) \approx \mathrm{cond}\, C$ (cf. [17, Section 3.3], [46, Theorem 3.4.9], [19]). In this case the values

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

tend to be significantly less than one.

7.3 Precision bounds

Finally we estimate the precision required in our error-free computation of the residual matrices U_i. Hereafter for a finite precision 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

$$\log_2(n||M||) \le s(M) \le \lfloor \log_2 ||M|| \rfloor, \tag{7.1}$$

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

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

Lemma 7.3. *We have $s(U_{i+1}) \le s(U_i) + \log_2(\theta_i n)$ for all i.*

Proof. The lemma follows from the bounds $u_{i+1} \le \theta_i u_i$ and (7.1). □

Lemma 7.4. *We have $s(U_{i+1}) \le s(CW_i) + \log_2 f_i$ and $s(U_{i+1}) \le s(W_i) + \log_2(f_i||C||)$ for $\theta_i < 1$, $f_i = \frac{\theta_i n}{|1-\theta_i|}$, and all i.*

Proof. First recall that $u_{i+1} \le \theta_i u_i$, so that $|u_i - u_{i+1}| \ge |1/\theta_i - 1| u_{i+1}$. The equation $U_i - U_{i+1} = CW_i$ implies that $||CW_i|| = ||U_i - U_{i+1}|| \ge |u_i - u_{i+1}| \ge |1/\theta_i - 1| u_{i+1}$. Therefore $u_{i+1} \le (f_i/n)||CW_i|| \le (f_i||C||/n)||W_i||$. Combine these bounds with bound (7.1) for $M = U_{i+1}$, $M = CW_i$ and $M = W_i$. □

Corollary 7.2.

$$\begin{aligned}
&a)\ If && t(U_{i+1}) \ge t(U_i),\\
&\quad then && p(U_{i+1}) \le p(U_i) + \log_2(\theta_i n).\\
&b)\ If && t(U_{i+1}) \ge t(CW_i),\\
&\quad then && p(U_{i+1}) \le p(CW_i) + \log_2 f_i.\\
&c)\ If && t(U_{i+1}) \ge t(W_i),\\
&\quad then && p(U_{i+1}) \le p(W_i) + \log_2(f_i||C||).
\end{aligned}$$

Recall that in virtue of Lemma 7.2, 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 7.3. *Suppose for two integers $\widehat{p}$ and $\tilde{p}$ we have the precision bounds $p(W_i) \le \widehat{p}$ and/or $p(CW_i) \le \tilde{p}$ and let this support some bound $\theta_i \le 1/n$ for all i. (This implies convergence with linear rate for the iterative refinement in Theorem 7.1.) Then we have uniform bound $\widehat{p} + \log_2(n/(n-1))$ 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 $\tilde{p} + \log_2(||C||n/(n-1))$.*

We cannot say a priori for which minimum precision bounds $\widehat{p}$ and $\tilde{p}$ the progress in iterative refinement is ensured, but we can find this dynamically, by beginning with the IEEE standard double precision and then increasing it recursively until convergence is observed. MSAs 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 7.4.

7.4 Flop count

To conclude this section, let us estimate the overall number of flops in our computations. Assume a normalized r-matrix A and a well conditioned A-modification $C = A + UV^H$. Then $||G|| = O(1/\operatorname{cond} A)$ (see the end of Section 5), and we yield the matrix G within the error norm

$$\epsilon \text{ in } O((\log \operatorname{cond} A)/\log(1/\epsilon))$$

steps of iterative refinement. We need $O(M_{A,r})$ double precision flops per step and therefore $O((M_{A,r} \log \operatorname{cond} A)/\log(1/\epsilon))$ double precision flops overall provided we can multiply the matrix A by an $n \times r$ matrix in $M_{A,r}$ flops and have a crude approximation to the inverse matrix C^{-1}. The computational cost is low for smaller integers r and, if the matrices A, UV^H, C and G share their structure and are represented with short generators, then also for larger integers r (see Section 8).

8 Matrix structure in the Schur Aggregation

To perform the Schur Aggregation, we apply Flowchart 4.1. For APPs of larger ranks r, the computational complexity dramatically decreases if the APP and matrices A and A^- have the same structure and can be represented with short generators (see [6], [7], [12], [15], [16], [26], [33, Chapters 1 and 4], [42], and the bibliography therein and in [50]). The decrease is usually by the factors of $r/\log^h r$ where h ranges from zero to two, depending on the structure.

We apply two principles to A-preconditioning for structured matrices.

- The operations in Flowchart 4.1 as well as the inversion of the matrix G can be reduced essentially to a small number of matrix multiplications and inversions, which we perform economically by operating with short generators of the structured input and auxiliary matrices rather than their entries.

- If the matrix A has structure, we rely on [33, Section 1.5] and Lemma 4.1 to extend this structure to the matrices involved in Flowchart 4.1 as well as to G^{-1} (whereas matrix structure is easily lost in the SVD-based APCs of larger ranks).

All our comments above can be readily extended to the dual APPs.

Various APPs with most frequently used matrix structures have been presented in [38, Examples 4.1–4.6]. Furthermore, we can apply the *method of displacement transformation* (see the remark below) to extend the power of these APPs to other classes of sparse and/or structured matrices, even to the classes that contain no well conditioned matrices and thus contain no well conditioned APPs [13], [48].

Remark 8.1. *By using appropriate structured multipliers, one can transform a matrix with the structure of a Cauchy, Vandermonde, Toeplitz, or Hankel type into a matrix with any other of these structures and can exploit such transforms to devise more effective algorithms. This method of displacement transformation was proposed in [31] (see its exposition also in [33, Sections 1.7, 4.8, and 4.9]). It was widely recognized due to the papers [14], [18], where the general class of Vandermonde-like multipliers in [31] was specialized to the Fourier transform multipliers, which transform the structures from the Toeplitz/Hankel into the Cauchy/Vandermonde types. This transform was used in [14], [18] for devising fast and numerically stable Gaussian elimination for Toeplitz/Hankel-like linear systems. For A-preconditioning, however, one should rather seek transition into the opposite direction, from Cauchy/Vandermonde-like matrices, which tend to be ill conditioned, to the Toeplitz/ Hankel-like structures. In this case the Fourier multipliers are not generally sufficient, but one can apply the original Vandermonde-like multipliers from [31].*

9 Multiplication/summation algorithms (an outline)

Effective MSAs in [8], [19], [24], [29] and the bibliography therein compute the sum and products with double or k-fold precision for any k, but the computations slow down for $k > 2$. Additive preconditioning for linear systems of equations, however, leads us to operating with the sums $s = t_1 + \cdots + t_h$ that nearly vanish compared to $\max_j |t_j|$. Moreover, in some cases we need these sums error-free, which one can handle by using multi-precision arithmetic, with respective slow down of the computations. We, however, avoid this slow down by applying the algorithms in [40]. The algorithms combine Dekker's splitting algorithm in [3] with the techniques of real modular reduction from [32] (see also [9]) and solve the problem by performing mostly double-precision additions.

In our next comments on the resulting MSAs, "addition" usually stands for "addition or subtraction", "dpn" and "dpn-1" are our abbreviations for "number represented with the IEEE standard double precision", and "dpn-ν" is the set of ν such dpns. Generally their sum is a multi-precision number, but it can be implicitly represented with the set "dpn-ν" by using double precision. Likewise, we can implicitly represent a $((p+1)\nu)$-bit floating point number with a dpn-ν where $p+1$ is the double precision.

The MSAs incorporate the Dekker's and Veltkamp's algorithms in [3] to compute the product of a dpn-μ and a dpn-ν error-free as a dpn-γ for $\gamma \le 2\mu\nu$. To add a dpn-μ and a dpn-ν we just combine them into a dpn-$(\mu+\nu)$.

To save some memory space without losing accuracy, we perform *compressing summation* where we are given a dpn-μ whose absolutely larger elements may immensely exceed the absolute value of their sum. The compressing summation outputs a (compressed) dpn-ν for the nearly minimum $\nu < \mu$ that represents precisely the same sum.

We adopt compressing summation from [40], where we perform some sequences of usual floating-point additions interrupted with the computation of the exponent of the current floating-point approximation of the sum of h numbers that we must compute. We compute this exponent every time when we update the sum, and we always add at least $\theta p - \log_2 h - O(1)$ new correct bits to the sum in every updating. Here $\theta = 1$ or $\theta = 2$ depending on our choice of the basic subroutine for floating-point summation that we apply in our MSAs. Accessing exponents of floating point numbers can be inexpensive. The IEEE floating point standard defines the function $\log b(x)$ to extract the significand and exponent of a floating point number (cf. [10], [21], [40]).

References

[1] 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.

[2] D. Coppersmith, S. Winograd, Matrix Multiplicaton via Arithmetic Progressions, *J. of Symbolic Computation*, **9**, **3**, 251–280, 1990.

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

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

[5] J. D. Dixon, Exact Solution of Linear Equations Using p-adic Expansions, *Numerische Math.*, **40**, 137–141, 1982.

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

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

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

[9] 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.

[10] Agner Fog, *How to Optimize for the Pentium Family of Microprocessors*, www.agner.org, 1996–2004, last updated 2004-04-16.

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

[12] J. R. Gilbert, H. Hafsteinsson, Parallel Symbolic Factorization of Sparse Linear Systems, *Parallel Computing*, **14**, 151–162, 1990.

[13] W. Gautschi, G. Inglese, Lower Bounds for the Condition Number of Vandermonde Matrices, *Numerische Math.*, **52**, 241–250, 1988.

[14] I. Gohberg, T. Kailath, V. Olshevsky, Fast Gaussian Elimination with Partial Pivoting for Matrices with Displacement Structure, *Math. of Computation*, **64**, 1557–1576, 1995.

[15] I. Gohberg, V. Olshevsky, Complexity of Multiplication with Vectors for Structured Matrices, *Linear Algebra and Its Applications*, **202**, 163–192, 1994.

[16] J. R. Gilbert, R. Schreiber, Highly Parallel Sparse Cholesky Factorization, *SIAM J. on Scientific Computing*, **13**, 1151–1172, 1992.

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

[18] G. Heinig, Inversion of Generalized Cauchy Matrices and the Other Classes of Structured Matrices, *Linear Algebra for Sigmal Processing, IMA Volume in Math. and Its Applications*, **69**, 95–114, 1995.

[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] *IA-32 Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture*, (Order Number 245470) Intel Corporation, Mt. Prospect, Illinois, 2001.

[22] I. Kaporin, A Practical Algorithm for Faster Matrix Multiplication, *Numerical Linear Algebra with Applications*, **6, 8**, 687-700, 1999.

[23] I. Kaporin, The Aggregation and Cancellation Techniques As a Practical Tool for Faster Matrix Multiplication, *Theoretical Computer Science*, **315**, **2–3**, 469–510, 2004.

[24] 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/.

[25] J. Laderman, V. Y. Pan, H. X. Sha, On Practical Algorithms for Accelerated Matrix Multiplication, *Linear Algebra and Its Applications*, **162–164**, 557–588, 1992.

[26] R. J. Lipton, D. Rose, R. E. Tarjan, Generalized Nested Dissection, *SIAM J. on Numerical Analysis*, **16**, **2**, 346–358, 1979.

[27] 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.

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

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

[30] V. Y. Pan, How Can We Speed up Matrix Multiplication? *SIAM Rev.,* **26**, **3**, 393–415, 1984.

[31] V. Y. Pan, On Computations with Dense Structured Matrices, *Math. of Computation*, **55, 191**, 179–190, 1990.

[32] V. Y. Pan, Can We Utilize the Cancelation of the Most Significant Digits? Tech. Report TR 92 061, *The International Computer Science Institute*, Berkeley, California, 1992.

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

[34] V. Y. Pan, Null Aggregation and Extensions, Technical Report TR 2007009, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, April 2007.

[35] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, I. Taj-Eddin, Y. Tang, X. Yan, Additive Preconditioning and Aggregation in Matrix Computations, Technical Report TR 2006006, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, May 2006.

[36] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning in Matrix Computations, Technical Report TR 2005009, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, July 2005.

[37] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning and Aggregation in Matrix Computations, Technical Report TR 2007002, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, March 2007.

[38] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, Additive Preconditioning for Matrix Computations, Technical Report TR 2007003, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, April 2007.

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

[40] V. Y. Pan, B. Murphy, G. Qian, R. E. Rosholt, Error-free Computations via Floating-Point Operations, Technical Report TR 2007010, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, April 2007.

[41] V. Y. Pan, B. Murphy, G. Qian, R. E. Rosholt, I. Taj-Eddin, Numerical Computation of Determinants with Additive Preconditioning, Technical Report TR 2007011, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, April 2007.

[42] V. Y. Pan, J. Reif, Fast and Efficient Parallel Solution of Sparse Linear Systems, *SIAM J. on Computing*, **22**, **6**, 1227–1250, 1993.

[43] 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.

[44] V. Y. Pan, X. Yan, Additive Preconditioning, Eigenspaces, and the Inverse Iteration, Technical Report TR 2007004, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, March 2007.

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

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

[47] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, 1998 (first edition), 2001 (second edition).

[48] E. E. Tyrtyshnikov, How Bad Are Hankel Matrices? *Numerische Math.*, **67, 2**, 261–269, 1994.

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

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