Towards a more robust algorithm for computing the restricted singular value decomposition*

Ian N. Zwaan†

Abstract. A new algorithm to compute the restricted singular value decomposition of dense matrices is presented. Like Zha's method [29], the new algorithm uses an implicit Kogbetliantz iteration, but with four major innovations. The first innovation is a useful quasi-upper triangular generalized Schur form that just requires orthonormal transformations to compute. Depending on the application, this Schur form can be used instead of the full decomposition. The second innovation is a new preprocessing phase that requires fewer rank determinations than previous methods. The third innovation is a numerically stable RSVD algorithm for 2×2 upper-triangular matrices, which forms a key component of the implicit Kogbetliantz iteration. The fourth innovation is an alternative scaling for the restricted singular triplets that results in elegant formulas for their computation. Beyond these four innovations, the qualitative (numerical) characteristics of the algorithm are discussed extensively. Some numerical challenges in the (optional) postprocessing phase are considered too; though, their solutions require further research. Numerical tests and examples confirm the effectiveness of the method.

Key words. Restricted singular value decomposition, RSVD, Kogbetliantz, numerically stable, rank decisions.

AMS subject classification. 65F15; 65F22; 65F30; 65F50; 65R30; 65R32

1 Introduction. The restricted singular value decomposition (RSVD) is a generalization of the ordinary singular value decomposition (SVD or OSVD) to matrix triplets. Applications of the RSVD include, for example, rank minimization of structured perturbations, unitarily invariant norm minimization with rank constraints, low rank approximation of partitioned matrices, restricted total least squares, generalized Gauss–Markov models, etc. See, e.g., Zha [28, 29], De Moor and Golub [11], and their references for more information. The problem is that computing the RSVD accurately and robustly is challenging, and avoiding numerical pitfalls is hard. The goal of the ideas and algorithms presented in this work is to improve upon existing computation methods in these areas, even though some numerical challenges remain.

The RSVD is a little-known generalization of the OSVD. Better-known is "the" generalized singular value decomposition (GSVD) for matrix pairs; see, e.g., Bai [3]. For now, it suffices to think of the RSVD as a GSVD for three matrices instead of two, with a formal definition following in Section 2. Since the RSVD and GSVD are just two out of infinitely many generalizations of the OSVD [9], a more appropriate name for the GSVD is the "quotient singular value decomposition" (QSVD). Thus, we adopt the mnemonics O-Q-R-SVD as standardized nomenclature for the rest of the text, as suggested by De Moor and Golub [10]. For more information on the relation between the SVD, QSVD, and RSVD, see, for example, De Moor and Golub [11, Sec. 2.2.4].

One illustration of the fact that computing the RSVD in a numerically sound way is not straightforward is Zha's constructive proof [28, Thm. 3.2], which he describes as unsuitable for computation. This is because it uses transformations with potentially ill-conditioned matrices in intermediate steps. Zha addresses this issue by deriving an implicit Kogbetliantz algorithm [29], but this algorithm lacks a (numerically) stable method for computing 2×2 RSVDs (cf. the 2×2 QSVD from Bai and Demmel [4]). Furthermore, the preprocessing phase of his implicit Kogbetliantz algorithm requires a sequence of up to four rank decisions, where each depends on the previous one. These dependencies,

*Version: February 10, 2020. This work was supported in part by the Deutsche Forschungsgemeinschaft through the collaborative research centre SFB-TRR55.

†Faculty of Mathematics and Natural Sciences, Bergische Universität Wuppertal, ianzwaan.com.

and the fact that rank determination is an ill-posed problem in floating-point arithmetic, make the preprocessing prone to errors. For example, it would be straightforward to construct a matrix triplet where we should have a clear gap in the singular values for each rank decision in exact arithmetic, but no longer have any gap (or a gap in the right place) for the fourth, or even third, rank decision in floating-point arithmetic. These faulty rank decisions can even show up if we use OSVD instead of, e.g., QR with pivoting, for the rank decisions.

Chu, De Lathauwer, and De Moor [8] present a QR based method which does not require a numerically stable 2×2 RSVD. Still, their method requires a sequence of up to five mutually dependent rank decisions, and it may also require nonorthonormal transformations in the preprocessing phase.

Another algorithm to compute the restricted singular values (RSVs) is due to Drmač [15], who uses both a Jacobi-type iteration and nonorthonormal transformations. Despite the latter, the algorithm still has favorable numerical properties, such as independence of certain types of diagonal scaling. Drmač also provides a bound on the backward error of his method, and discusses when one can expect the computed singular values to have high relative accuracy. A potential downside of this method is that it does not compute the "full" RSVD. Another issue is that the algorithm requires the assumption that one of the input matrices is nonsingular, which needs not be true in the general case.

Like Zha's algorithm, the algorithm in this work centers around an implicit Kogbetliantz iteration, but with four main innovations over the existing algorithms. The first main innovation is a generalized Schur-form RSVD consisting of a triplet of quasi upper-triangular matrices that we can compute just with orthonormal transformations. In particular, this Schur form allows us to skip the postprocessing necessary to get the full decomposition, while still being useful for certain applications; see Section 2 for details. The second main innovation is a new preprocessing phase, discussed in Section 3, that uses fewer transformations and rank decisions, and has fewer dependencies between the rank decisions. The third main innovation is a numerically stable 2×2 RSVD algorithm like Bai and Demmel's backward stable 2×2 QSVD algorithm [4]. This 2×2 algorithm is a crucial part of the implicit Kogbetliantz iteration, and we investigate its numerical properties in exact and floating-point arithmetic in Sections 4.2, 4.3, and 5. The fourth main innovation is primarily discussed in Section 6 and consists of an alternative scaling of the restricted singular value triplets. This new scaling leads to mathematically and numerically elegant formulas for the computation of the triplets.

Since rank revealing decompositions do not necessarily need to use orthonormal transformations, we can combine ideas from Drmač's nonorthogonal algorithm with the new preprocessing phase and the implicit Kogbetliantz iteration from this work. Section 8 contains an overview of how this hybrid algorithm would work. While the use of nonorthogonal transformations in the earlier phases is optional, the postprocessing phase generally requires nonorthogonal transformations, as we will see in Section 7.

The numerical tests in Section 9 consist of three parts. The first part is dedicated to verifying the numerical properties of the 2×2 RSVD algorithm. The second part focuses on the rate of convergence of the implicit Kogbetliantz iteration. The third part compares the accuracy of the new RSVD method with existing methods, and also compares the effect of different implementations of the preprocessing phase on the accuracy. The results show that the new 2×2 RSVD is numerically stable, and that the implicit Kogbetliantz iteration typically converges rapidly and can compute the RSVs with high accuracy. In fact, for ill-conditioned matrices the accuracy the new method can exceed that of existing methods by several orders of magnitude.

Throughout this work we use uppercase letters for matrices, lowercase letters for their elements and for scalars, and bold lowercase letters for vectors. The matrix I is always an identity matrix, $\boldsymbol{e}_j$ the jth canonical basis vector, 0 a zero matrix or scalar, and $\times$ an arbitrary matrix or scalar that can be nonzero. These quantities always have a size that is appropriate for the context in which they are used. In some places we use Matlab notation when stacking block matrices vertically; for

example, $[A;\ C] = [A^T\ C^T]^T$. As usual, $\|\cdot\|_p$ denotes the (induced) p-norm for $1 \le p \le \infty$, and we sometimes drop the index for $p = 2$ when no other norms are used in the same context. Other norms that we use are the Frobenius norm $\|\cdot\|_F$ and the max norm $\|\cdot\|_{\max}$, where the latter equals the largest magnitude of any element in the matrix. Finally, the absolute value notation $|\cdot|$ acts elementwise on matrices.

2 Background and theory. The definition of the RSVD given by the theorem below combines the ones from Zha [28, Lem. 4.1] and De Moor and Golub [11, Thm. 1], but with some small changes. In particular, some of the blocks in (1) are in a different position, which helps with the computation of the decomposition, and some of the trivial triplets are counted differently. Furthermore, the theorem below and the theory and algorithms in the rest of this work focus on the real case for simplicity and clarity, although we can compute the RSVD of a triplet of complex matrices too.

Theorem 1 (RSVD — Diagonal Form)**.** *Let $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$, and $C \in \mathbb{R}^{n\times q}$, and define $r_A = \operatorname{rank} A$, $r_B = \operatorname{rank} B$, $r_C = \operatorname{rank} C$, $r_{AB} = \operatorname{rank}[A\ B]$, $r_{AC} = \operatorname{rank}[A;\ C]$, and $r_{ABC} = \operatorname{rank}\left[\begin{smallmatrix} A & B \\ C & 0 \end{smallmatrix}\right]$. Then the triplet of matrices (A, B, C) can be factorized as $A = X^{-T}\Sigma_\alpha Y^{-1}$, $B = X^{-T}\Sigma_\beta U^T$, and $C = V\Sigma_\gamma Y^{-1}$, where $X \in \mathbb{R}^{p\times p}$ and $Y \in \mathbb{R}^{q\times q}$ are nonsingular, and $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthonormal. Furthermore, Σ_α, Σ_β, and Σ_γ are quasi-diagonal[1] with nonnegative entries, and are such that $\left[\begin{smallmatrix} \Sigma_\alpha & \Sigma_\beta \\ \Sigma_\gamma & \end{smallmatrix}\right]$ can be written as*

(1)

	q_1	q_2	q_3	q_4	q_5	q_6	m_1	m_2	m_3	m_4	
p_1	0	0	D_α	0	0	0	D_β	0	0	0	$p_1 = q_3 = r_{ABC} + r_A - r_{AB} - r_{AC}$
p_2	0	0	0	I	0	0	0	I	0	0	$p_2 = q_4 = r_{AC} + r_B - r_{ABC}$
p_3	0	0	0	0	I	0	0	0	0	0	$p_3 = q_5 = r_{AB} + r_C - r_{ABC}$
p_4	0	0	0	0	0	I	0	0	0	0	$p_4 = q_6 = r_{ABC} - r_B - r_C$
p_5	0	0	0	0	0	0	0	0	0	I	$p_5 = r_{AB} - r_A$, $q_2 = r_{AC} - r_A$
p_6	0	0	0	0	0	0	0	0	0	0	$p_6 = p - r_{AB}$, $q_1 = q - r_{AC}$
n_1	0	I	0	0	0	0					$n_1 = q_2$, $m_4 = p_5$
n_2	0	0	D_γ	0	0	0					$n_2 = m_1 = p_1 = q_3$
n_3	0	0	0	0	I	0					$n_3 = p_3 = q_5$, $m_2 = p_2 = q_4$
n_4	0	0	0	0	0	0					$n_4 = n - r_C$, $m_3 = m - r_B$,

where $D_\alpha = \operatorname{diag}(\alpha_1, \ldots, \alpha_{p_1})$, $D_\beta = \operatorname{diag}(\beta_1, \ldots, \beta_{p_1})$, and $D_\gamma = \operatorname{diag}(\gamma_1, \ldots, \gamma_{p_1})$. Moreover, α_j, β_j, and γ_j are scaled such that $\alpha_j^2 + \beta_j^2\gamma_j^2 = 1$ for $i = 1, \ldots, p_1$. Besides the p_1 triplets $(\alpha_j, \beta_j, \gamma_j)$, there are p_2 triplets $(1, 1, 0)$, p_3 triplets $(1, 0, 1)$, p_4 triplets $(1, 0, 0)$, and $\min\{p_5, q_2\}$ triplets $(0, 1, 1)$. This leads to a total of $p_1 + p_2 + p_3 + p_4 + \min\{p_5, q_2\} = r_A + \min\{p_5, q_2\} = \min\{r_{AB}, r_{AC}\}$ regular triplets of the form (α, β, γ) with $\alpha^2 + \beta^2\gamma^2 = 1$. Each of these triplets corresponds to a restricted singular value *$\sigma = \alpha/(\beta\gamma)$, where the result is ∞ by convention if $\alpha \neq 0$ and $\beta\gamma = 0$. Finally, the triplet has a right (or column) trivial block of dimension $q_1 = \dim(\mathcal{N}(A) \cap \mathcal{N}(C))$, and a left (or row) trivial block of dimension $p_6 = \dim(\mathcal{N}(A^T) \cap \mathcal{N}(B^T))$.*

Remark 2. Zha [28, Sec. 4] and De Moor and Golub [11, Sec. 2.1] differ in the triplets that they list. In particular, the former does not list any of the triplets $(0, 0, 0)$, $(0, 0, 1)$, and $(0, 1, 0)$; whereas the latter do not list $(0, 1, 1)$, but instead the equivalent of p_5 triplets $(0, 1, 0)$ and q_2 triplets $(0, 0, 1)$. Theorem 1 adopts Zha's definition of $(0, 1, 1)$, because of [28, Thm. 4.2] and the simple example $(A, B, C) = (0, 1, 1)$, and avoids problematic definitions of trivial triplets by listing the left and right trivial blocks.

[1] A quasi-diagonal matrix, in this work, is a matrix that is diagonal after removing all zero rows and columns.

Remark 3. The typical scaling of the triplets $(\alpha_i, \beta_i, \gamma_i)$ in literature is such that $\alpha_i^2 + \beta_i^2 + \gamma_i^2 = 1$, rather than $\alpha_i^2 + \beta_i^2\gamma_i^2 = 1$ as in the theorem above. But we will see in Section 6 that the latter scaling has theoretical and computational benefits. Besides, if $\alpha_i^2 + \beta_i^2\gamma_i^2 = 1$, then we can compute (as in Zha [28, Thm. 4.1]) $\widetilde{\alpha}_i = \alpha_i^2(1+\alpha_i^2)^{-1/2}$, $\widetilde{\beta}_i = \beta\gamma_i$, and $\widetilde{\gamma}_i = \alpha_i(1+\alpha_i^2)^{-1/2}$, so that $\widetilde{\alpha}_i^2 + \widetilde{\beta}_i^2 + \widetilde{\gamma}_i^2 = 1$.

Corollary 4 (RSVD — Triangular Form)**.** *Let $X^{-T} = PS$ and $Y^{-1} = TQ^T$, where $P \in \mathbb{R}^{p\times p}$ and $Q \in \mathbb{R}^{q\times q}$ are orthonormal, and $S \in \mathbb{R}^{p\times p}$ and $T \in \mathbb{R}^{q\times q}$ are nonsingular and upper triangular. Then the triplet (A, B, C) can be factorized as $A = P(S\Sigma_\alpha T)Q^T$, $B = P(S\Sigma_\beta)U^T$, and $C = V(\Sigma_\gamma T)Q^T$.*

Suppose that A, B, and C are nonsingular and have compatible sizes; then the restricted singular values of the triplet (A, B, C) are the ordinary singular values of $B^{-1}AC^{-1}$. Just like Drmač's algorithm [15], the method described in this work is to look at the singular values of $CA^{-1}B$ instead. The benefit for more general matrices is that it suffices to "extract" a triplet with a nonsingular A during the preprocessing, rather than having to extract a triplet with nonsingular B and C (cf. Zha's algorithm [29]). It turns out that this alternative extraction requires fewer transformations, and more importantly, fewer rank decisions. To see why we can change our perspective like this, we first need the following definition of the regular RSVs.

Theorem 5 (Zha [28, Def. 2.1])**.** *The regular restricted singular values of the matrix triplet A, B, and C can be characterized as*

$$\sigma_i = \min_D\{\|D\| : \operatorname{rank}(A + BDC) \le i - 1\} \qquad (i = 1, \ldots, r_A + \min\{p_5, q_2\}).$$

The value $\sigma_i = \infty$ corresponds to the situation that we cannot find any matrix D to make the rank of $A + BDC$ less than or equal to $i - 1$.

Now we can prove the following proposition, which formalizes the idea of working with $CA^{-1}B$ instead of $B^{-1}AC^{-1}$ for general matrices.

Proposition 6. *Suppose that the matrices A, B, and C have compatible sizes and that the restricted singular values of the triplet (A, B, C) are defined as in Theorem 5. Then, for the nonzero restricted singular values it holds that*

$$\sigma_i^{-1} = \min_D\{\|D\| \mid \operatorname{rank}(D + CA^\dagger B) \le r_A - i\} \qquad (i = 1, \ldots, r_A), \tag{2}$$

where $A^\dagger$ denotes the Moore–Penrose pseudoinverse of A and $\infty^{-1} = 0$ by convention. The remaining $\min\{p_5, q_2\}$ regular RSVs can be characterized as $0^{-1} = \infty$.

Proof. Suppose that A, B, and C are decomposed as in Theorem 1 and let $E = V^T DU$; then $V^T(CA^\dagger B)U = \Sigma_\gamma\Sigma_\alpha^\dagger\Sigma_\beta$ and $\operatorname{rank}(D + CA^\dagger B) = \operatorname{rank}(E + \Sigma_\gamma\Sigma_\alpha^\dagger\Sigma_\beta)$. By using the definition of the Σs and the fact that $\|D\| = \|E\|$, it follows that the minimization in (2) is equivalent to

$$\min_E \left\{ \|E\| \mid \operatorname{rank} \begin{bmatrix} E_{11} & E_{12} \\ E_{21} + D_\gamma D_\alpha^{-1} D_\beta & E_{22} \\ E_{31} & E_{32} \end{bmatrix} \le r_A - i \right\},$$

which equals σ_i^{-1} for the nonzero RSVs. In particular $\sigma_i^{-1} = 0$ for $i = 1, \ldots, p_2 + p_3 + p_4$, and $\sigma_i^{-1} > 0$ for $i = p_2 + p_3 + p_4 + 1, \ldots, r_A$. □

We can interpret the proposition above as a generalization of Zha [28, Cor 4.1], but it is also related to the analysis of generalized Schur complements in De Moor and Golub [11, Sec. 3.2.1]. An important observation is that we do not need the full RSVD to compute $CA^\dagger B$. In fact, the outputs of the new algorithm after the preprocessing phase and the implicit Kogbetliantz iteration are the matrices P, Q, U, and V, and the products P^TAQ, P^TBU, and V^TCQ, which are such that

$$(V^TCQ)(P^TAQ)^\dagger(P^TBU) = \Sigma_\gamma\Sigma_\alpha^\dagger\Sigma_\beta$$

is quasi-diagonal and easily determined. The postprocessing is only necessary to get the individual factors S and T, and it depends on the application if we need those. This suggest the following decomposition, which can be thought of as kind of generalized Schur decomposition like the QZ decomposition for generalized eigenvalue problems.

Theorem 7 (RSVD — Generalized Schur form)**.** *Let $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$, and $C \in \mathbb{R}^{n\times q}$; then there exist orthonormal matrices P, Q, U, and V, such that*

$$P^TAQ = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccccc} q_1 & q_2 & q_3 & q_4 & q_5 \end{array} \\ \begin{bmatrix} 0 & 0 & A_{13} & A_{14} & A_{15} \\ 0 & 0 & 0 & A_{24} & A_{25} \\ 0 & 0 & 0 & 0 & A_{35} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{3}$$

$$P^TBU = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{bmatrix} 0 & B_{12} & B_{13} & B_{14} \\ 0 & 0 & B_{23} & B_{24} \\ 0 & 0 & 0 & B_{34} \\ 0 & 0 & 0 & B_{44} \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{4}$$

$$V^TCQ = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \\ n_4 \end{array} \begin{array}{c} \begin{array}{ccccc} q_1 & q_2 & q_3 & q_4 & q_5 \end{array} \\ \begin{bmatrix} 0 & C_{12} & C_{13} & C_{14} & C_{15} \\ 0 & 0 & 0 & C_{24} & C_{25} \\ 0 & 0 & 0 & 0 & C_{35} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{5}$$

where A_{13}, A_{24}, A_{34}, B_{44}, and C_{12} are nonsingular and upper triangular; B_{23} and C_{24} are square and upper triangular; and B_{12} and C_{35} are upper trapezoidal with $p_1 \geq m_2$ and $n_3 \leq q_5$, respectively. Here, upper trapezoidal with the given dimensions means that B_{12} and C_{35} are structured as

$$\begin{bmatrix} \times & \cdots & \times \\ \vdots & & \vdots \\ \times & \cdots & \times \\ & \ddots & \vdots \\ & & \times \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \times & \cdots & \times & \cdots & \times \\ & \ddots & \vdots & & \vdots \\ & & \times & \cdots & \times \end{bmatrix},$$

respectively. Furthermore, the matrices A_{24}, B_{23}, and C_{24} are such that $C_{24}A_{24}^{-1}B_{23} = \Sigma$, where Σ is diagonal with nonnegative entries.

Proof. The structure of the matrices P^TAQ, P^TBU, and V^TCQ follows from the preprocessing phase from Section 3. For the claim that $C_{24}A_{24}^{-1}B_{23} = \Sigma$, we can use the following limit argument. Suppose that $\widehat{P}$, $\widehat{Q}$, $\widehat{U}$, and $\widehat{V}$ are such that $\widehat{P}^TA\widehat{Q}$, $\widehat{P}^TB\widehat{U}$, and $\widehat{V}^TC\widehat{Q}$ have the structure from (3)–(4), but that $C_{24}A_{24}^{-1}B_{23}$ is not yet diagonal. Then let $\{B_k\}$ be a bounded sequence of nonsingular upper-triangular matrices that converge to B_{23}. For each k, let $\widetilde{V}_k^T(C_{24}A_{24}^{-1}B_k)\widetilde{U}_k = \Sigma_k$ be an SVD of $C_{24}A_{24}^{-1}B_k$. Furthermore, let $\widetilde{P}_k$ and $\widetilde{Q}_k$ be orthonormal matrices such that $\widetilde{P}_k^TB_k\widetilde{U}_k$ and $\widetilde{P}_k^TA_{23}\widetilde{Q}_k$ are upper triangular, respectively. Then $C_{24} = \Sigma_k(\widetilde{P}_k^TB_k\widetilde{U}_k)^{-1}(\widetilde{P}_k^TA_{24}\widetilde{Q}_k)$ is also upper triangular. Using

the Bolzano–Weierstrass theorem, we know that the bounded sequence $\{(\widetilde{P}_k, \widetilde{Q}_k, \widetilde{U}_k, \widetilde{V}_k, \Sigma_k)\}$ has a converging subsequence

$$\lim_{i\to\infty} (\widetilde{P}_{k_i}, \widetilde{Q}_{k_i}, \widetilde{U}_{k_i}, \widetilde{V}_{k_i}, \Sigma_{k_i}) = (\widetilde{P}, \widetilde{Q}, \widetilde{U}, \widetilde{V}, \Sigma).$$

It is easy to show that $\widetilde{P}$, $\widetilde{Q}$, $\widetilde{U}$, and $\widetilde{V}$ are orthonormal and that $\widetilde{P}^T A_{24}\widetilde{Q}$, $\widetilde{P}^T B_{23}\widetilde{U}$, and $\widetilde{V}^T C_{24}\widetilde{Q}$ are upper triangular, and satisfy $\widetilde{V}^T C_{24} A_{24}^{-1} B_{23}\widetilde{U} = \Sigma$, where Σ is a diagonal matrix with nonnegative entries. Hence, the products

$$P = \widehat{P}\,\mathrm{diag}(I, \widetilde{P}, I, I, I), \quad U = \widehat{U}\,\mathrm{diag}(I, I, \widetilde{U}, I),$$
$$Q = \widehat{Q}\,\mathrm{diag}(I, I, I, \widetilde{Q}, I), \quad V = \widehat{V}\,\mathrm{diag}(I, \widetilde{V}, I, I),$$

are the sought after orthonormal transformations. □

The P, Q, U, V, p_i, q_i, m_i, and n_i from the above theorem are not necessarily equal to their counterparts from Theorem 1 and Corollary 4. The main benefit of the Schur-form RSVD is that we can compute it with only orthonormal transformations and at most three rank decisions, while we can still use it to compute, e.g., $CA^\dagger B$ and the RSVs. The computation of the Schur form is the subject of later sections, but to see how we can use it, consider the following proposition first.

Proposition 8. *Let A, B, C, P, Q, U, and V be as in Theorem 7; then there exist nonsingular upper-triangular matrices S and T so that $S^{-1}(P^TAQ)T^{-1}$, $S^{-1}(P^TBU)$, and $(V^TCQ)T^{-1}$ have the form*

$$(6)\quad \begin{bmatrix} 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & A_{24} & 0 \\ 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & B_{12} & B_{13} & 0 \\ 0 & 0 & B_{23} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \textit{and} \quad \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{24} & C_{25} \\ 0 & 0 & 0 & 0 & C_{35} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

respectively.

Proof. Let

$$(7)\quad S = \begin{bmatrix} I & & \frac{1}{2}A_{15} & B_{14} & \\ & I & A_{25} & B_{24} & \\ & & A_{35} & B_{34} & \\ & & & B_{44} & \\ & & & & I \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} I & & & & \\ & C_{12} & C_{13} & C_{14} & C_{15} \\ & & A_{13} & A_{14} & \frac{1}{2}A_{15} \\ & & & I & \\ & & & & I \end{bmatrix};$$

then direct verification concludes the proof. □

Now, with the Schur-form RSVD and Proposition 8, we see that

$$V^T C A^\dagger B U = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & C_{24}A_{24}^{-1}B_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \Sigma & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

is indeed easily determined. Likewise, if D is a 4×4 block matrix of compatible dimensions, then

$$\begin{aligned}\operatorname{rank}(A + BDC) &= \operatorname{rank} \begin{bmatrix} 0 & \times & I & \times & \times \\ 0 & B_{23}D_{31} & 0 & A_{24} + B_{23}D_{32}C_{24} & \times \\ 0 & 0 & 0 & 0 & I \\ 0 & D_{41} & 0 & D_{42}C_{24} & \times \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ &= p_1 + p_3 + \operatorname{rank}\left(\begin{bmatrix} 0 & A_{24} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} B_{23} & \\ & I \end{bmatrix} \begin{bmatrix} D_{31} & D_{32} \\ D_{41} & D_{42} \end{bmatrix} \begin{bmatrix} I & \\ & C_{24} \end{bmatrix} \right).\end{aligned}$$

This yields $p_1 + p_3$ RSVs at ∞, $\min\{p_4, q_2\}$ at 0, and p_2 reciprocals from the diagonal elements of $C_{24}A_{24}^{-1}B_{23} = \Sigma$ (where $0^{-1} = \infty$ by convention).

3 The preprocessing phase. Zha's algorithm for computing the RSVD starts with a preprocessing phase to extract a triplet (A', B', C') from a given matrix triplet (A, B, C), where A' is square and upper triangular, and B' and C' are nonsingular and upper triangular. The goal of the preprocessing phase described in this section is similar to Zha's in that it extracts square and upper triangular matrices. But unlike Zha's approach, we only need A' to be nonsingular, rather than both B' and C'. Moreover, this triplet should correspond to the nonzero regular RSVs of (A, B, C) so that we can (implicitly) apply the Kogbetliantz iteration to the product $C'(A')^{-1}B'$ in the next phase. The result is a triplet of matrices with the structure from (3)–(5).

Two key procedures in Zha's preprocessing phase are so called *row compressions* and *column compressions*. For simplicity, we only use the combined action of both compressions, which leads to the definition below.

Definition 9. Let M be a real matrix; then we refer to the URV decomposition $U^TMV = \left[\begin{smallmatrix} 0 & R \\ 0 & 0 \end{smallmatrix}\right]$ as the *compression* of M if R is nonsingular and U and V are orthonormal matrices.

One way to compress a matrix is to use the SVD, although any rank-revealing URV decomposition would work. For example, the QR decomposition with column pivoting is a popular and fast alternative to the SVD in this context. See, e.g., Fierro, Hansen, and Hansen [18] for an overview of the qualitative differences between different URV decompositions in floating-point arithmetic.

We use compressions, along with other orthonormal transformations, in the preprocessing phase to compute a sequence of orthonormal matrices $P^{(\ell)}$, $Q^{(\ell)}$, $U^{(\ell)}$, and $V^{(\ell)}$. These matrices are such that if we start with $A^{(1)} = A$, $B^{(1)} = B$, and $C^{(1)} = C$, and compute

$$A^{(\ell+1)} = P^{(\ell)T}A^{(\ell)}Q^{(\ell)},\ B^{(\ell+1)} = P^{(\ell)T}B^{(\ell)}U^{(\ell)},\ \text{and } C^{(\ell+1)} = V^{(\ell)T}C^{(\ell)}Q^{(\ell)}; \tag{8}$$

we will ultimately get partitioned matrices $A^{(i)}$, $B^{(i)}$, and $C^{(i)}$ from which we can take specific blocks as the sought-after triplet. That is, the matrices $A^{(i)}$, $B^{(i)}$, $C^{(i)}$ have blocks that are square and upper triangular, with $A^{(i)}$ nonsingular, that correspond to the nonzero RSVs of (A, B, C).

Since partitioned matrices will be important, both in the preprocessing phase and the postprocessing phase, the following implicit notations help to simplify the presentation. First, all blocks have indices that mark their respective positions in their matrix and are of no further importance. Second, if M is a partitioned matrix and if M_{ij} denotes the block at the ith block row and jth block column, then any transformation of M_{ij} into $X_{ii}^TM_{ij}Y_{jj}$ has a corresponding transformation of M into X^TMY. In particular, $X = \operatorname{diag}(I, X_{ii}, I)$ and $Y = \operatorname{diag}(I, Y_{jj}, I)$ so that $(X^TMY)_{ij} = X_{ii}^TM_{ij}Y_{jj}$, unless stated otherwise. Third, the latter notation is understood to work recursively, so that if $M^{(\ell)}$ is a submatrix of $M^{(\ell-1)}$, then any transformation applied to $M^{(\ell)}$, or a block of $M^{(\ell)}$, has a corresponding transformation applied to $M^{(\ell-1)}$.

Now for the first step of the preprocessing phase, let $P^{(1)}$ and $Q^{(1)}$ compress $A^{(1)}$, and take $U^{(1)} = I$ and $V^{(1)} = I$, then

$$A^{(2)} = \begin{bmatrix} 0 & A_{12}^{(2)} \\ 0 & 0 \end{bmatrix}, \qquad B^{(2)} = \begin{bmatrix} B_{11}^{(2)} \\ B_{21}^{(2)} \end{bmatrix}, \quad \text{and} \quad C^{(2)} = \begin{bmatrix} C_{11}^{(2)} & C_{12}^{(2)} \end{bmatrix}.$$

Let $P_{22}^{(2)}$ and $U_{11}^{(2)}$ compress $C_{11}^{(2)}$, and let $V_{11}^{(2)}$ and $Q_{11}^{(2)}$ to compress $B_{21}^{(2)}$, so that

$$A^{(3)} = \begin{bmatrix} 0 & 0 & A_{13}^{(3)} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad B^{(3)} = \begin{bmatrix} B_{11}^{(3)} & B_{12}^{(3)} \\ 0 & B_{22}^{(3)} \\ 0 & 0 \end{bmatrix}, \quad \text{and} \quad C^{(3)} = \begin{bmatrix} 0 & C_{12}^{(3)} & C_{13}^{(3)} \\ 0 & 0 & C_{23}^{(3)} \end{bmatrix}.$$

The compressions of $A^{(1)}$, $B_{21}^{(2)}$, and $C_{11}^{(2)}$ contain the only three rank decisions necessary in the preprocessing phase. Now, by plugging $C^{(3)}A^{(3)\dagger}B^{(3)}$ into Theorem 5, we see that we may focus on the triplet $(A_{13}^{(3)}, B_{11}^{(3)}, C_{23}^{(3)})$, where $A_{13}^{(3)}$ is nonsingular. The matrices $B_{11}^{(3)}$ and $C_{23}^{(3)}$ are not necessarily square or upper triangular at this point, so we are not yet finished. For convenience set $A^{(4)} = A_{13}^{(3)}$, $B^{(4)} = B_{11}^{(3)}$, and $C^{(4)} = C_{23}^{(3)}$, and suppose that $A^{(4)}$ is $p' \times p'$, $B^{(4)}$ is $p' \times m'$, and $C^{(4)}$ is $n' \times p'$, and let $l = \min\{m', n', p'\}$. Then there are three possibilities to consider (where the choice is free when there is overlap).

1. If $m', n' \geq p' = l$, then take $P^{(4)} = Q^{(4)} = I$ and $A^{(5)} = A^{(4)}$. Furthermore, use a QR decomposition to compute $V^{(4)}$ such that $V^{(4)T}C^{(4)} = [C_{11}^{(5)};\ 0]$, where $C_{11}^{(5)}$ is upper triangular; and use an RQ decomposition[2] to compute $U^{(4)}$ such that $B^{(4)}U^{(4)} = [0\ B_{12}^{(5)}]$, where $B_{12}^{(5)}$ is upper triangular. Since

$$\begin{bmatrix} C_{11}^{(5)} \\ 0 \end{bmatrix} (A^{(5)})^{-1} \begin{bmatrix} 0 & B_{12}^{(5)} \end{bmatrix} = \begin{bmatrix} 0 & C_{11}^{(5)}(A^{(5)})^{-1}B_{12}^{(5)} \\ 0 & 0 \end{bmatrix}$$

   we see that we can restrict our attention to the triplet $(A^{(5)}, B_{12}^{(5)}, C_{11}^{(5)})$.

2. If $r = \min\{n', p'\} \geq m' = l$. First use a QR decomposition to compute $P^{(4)}$ such that $P^{(4)T}B^{(4)} = [B_{11}^{(5)};\ 0]$, where $B_{11}^{(5)}$ is an $m' \times m'$ upper-triangular matrix. Next, compute $Q^{(4)}$ with an RQ decomposition so that $A^{(5)} = P^{(4)T}A^{(4)}Q^{(4)}$ is upper triangular. Then use a QR decomposition to compute $V^{(4)}$ such that $C^{(5)} = V^{(4)T}C^{(4)}Q^{(4)}$ has the form

$$\begin{array}{l} \\ m' \\ r - m' \\ \max\{0, n' - r\} \end{array} \begin{array}{c} \begin{array}{cc} m' & p' - m' \end{array} \\ \begin{bmatrix} C_{11}^{(5)} & C_{12}^{(5)} \\ 0 & C_{22}^{(5)} \\ 0 & 0 \end{bmatrix} \end{array},$$

   where $C_{11}^{(5)}$ is upper triangular and $C_{22}^{(5)}$ is upper trapezoidal with $r - m' \leq p' - m'$. Assuming $A^{(5)}$ is partitioned conformally,

$$C^{(5)}(A^{(5)})^{-1}B^{(5)} = \begin{bmatrix} C_{11}^{(5)}(A_{11}^{(5)})^{-1}B_{11}^{(5)} \\ 0 \\ 0 \end{bmatrix},$$

[2]The RQ decomposition of a real $m \times n$ matrix A is like a QR decomposition, but with the factors in the opposite order. That is, if $m > n$, then $A = RQ^T$ for some upper-trapezoidal matrix R and some Q satisfying $Q^TQ = I$. If $m \geq n$, then $A = [0\ R]Q^T$ or $A = [R\ 0]Q^T$, where Q is as before and R is upper triangular. See, e.g., the routine xGERQF in LAPACK [2, Sec. 2.4.2.5].

which shows that we can restrict our attention to $(A_{11}^{(5)}, B_{11}^{(5)}, C_{11}^{(5)})$. We can save work if we just want to compute the restricted singular triplets, because then it suffices to only compute $C_{11}^{(5)}$ with a QR QR decomposition of the m' left-most columns of $C^{(4)}Q^{(4)}$.

3. If $r = \min\{m', p'\} \geq n' = l$. First use an RQ decomposition to compute $Q^{(4)}$ such that $C^{(4)}Q^{(4)} = [0\ C_{12}^{(5)}]$, where $C_{12}^{(5)}$ is a $n' \times n'$ upper-triangular matrix. Next, compute $P^{(4)}$ such that $A_{14}^{(5)} = P^{(4)T}A^{(4)}Q^{(4)}$ is upper triangular. Then use an RQ decomposition to compute $U^{(4)}$ such that $B^{(5)} = P^{(4)T}B^{(4)}U^{(4)}$ has the form

$$\begin{array}{c} \\ p'-n' \\ n' \end{array} \begin{array}{c} \begin{array}{ccc} \max\{0, m'-r\} & r-n' & n' \end{array} \\ \begin{bmatrix} 0 & B_{12}^{(5)} & B_{13}^{(5)} \\ 0 & 0 & B_{23}^{(5)} \end{bmatrix} \end{array},$$

where $B_{23}^{(5)}$ is upper triangular and $B_{12}^{(5)}$ is upper trapezoidal with $p'-n' \geq r-n'$. Assuming $A^{(5)}$ is partitioned conformally,

$$C^{(5)}(A^{(5)})^{-1}B^{(5)} = \begin{bmatrix} 0 & 0 & C_{12}^{(5)}(A_{22}^{(5)})^{-1}B_{23}^{(5)} \end{bmatrix},$$

which shows that we can restrict our attention to $(A_{22}^{(5)}, B_{23}^{(5)}, C_{12}^{(5)})$. We can save work if we just want to compute the restricted singular triplets, because then it suffices to only compute $B_{23}^{(5)}$ with an RQ decomposition of the bottom n' rows of $P^{(4)T}B^{(4)}$.

The Schur form from Theorem 7 corresponds to the above three cases with

1. $p_1 = q_3 = 0$ and $p_3 = q_5 = 0$ and $m_1 = n_4 = 0$ (or $m_2 = n_3 = 0$),
2. $p_1 = q_3 = 0$ and $m_1 = m_2 = 0$,
3. $p_3 = q_5 = 0$ and $n_3 = n_4 = 0$,

respectively, and $m_3 = n_2 = p_2 = q_4 = l$. In principle, we may assume that we always have the first or second case, because we can transform the input triplet (A, B, C) to $(\Pi_r A^T \Pi_c, \Pi_r C^T \Pi_c, \Pi_r B^T \Pi_c)$, where Π_r and Π_c are the antidiagonal permutation matrices that reverse the order of the rows and columns. In any case, we can compute the nonzero restricted singular triplets of (A, B, C) from specific square and upper-triangular blocks of $A^{(5)}$, $B^{(5)}$, and $B^{(5)}$, where the block coming from $A^{(5)}$ is nonsingular. These three blocks correspond to the A_{24}, B_{23}, and C_{24} from (3)–(5), respectively, and have exactly the form we need for the implicit Kogbetliantz iteration described in the next section.

4 The Kogbetliantz phase.

4.1 The implicit Kogbetliantz method. For a given triplet of upper-triangular $l \times l$ matrices A, B, and C, where A is nonsingular, the goal of the Kogbetliantz phase is to find orthonormal matrices P, Q, U, and V, so that P^TAQ, P^TBU, and V^TCQ are upper-triangular and $V^TCA^{-1}BU$ is diagonal. The essence of this phase is to implicitly apply a Kogbetliantz-type iteration to $M = CA^{-1}B$; that is, to compute the SVD of M without forming M or computing A^{-1}. This is different from Zha's approach [28], who implicitly applies the iteration to the product $B^{-1}AC^{-1}$. A description of the new procedure follows below; for more background and details see, e.g., Bai and Demmel [4], Charlier, Vanbegin, and Van Dooren [7], Forsythe and Henrici [19], Hansen [21], Heath et al. [22], Paige [26], and Zha [29], and their references.

The implicit Kogbetliantz method iterates over pairs (i, j) with $i < j \leq l$, and for each pair applies rotations to the ith and jth rows and columns of A, B, and C. This is done in such a way that a_{ij}, b_{ij}, c_{ij}, and also m_{ij} become zero, while the corresponding (j, i)th elements (may) become nonzero. We refer to this as annihilating the (i, j)th elements. A sequence of iterations over all $n(n-1)/2$ pairs (i, j) is called a cycle, and a cycle can progress through the pairs in different orderings. Some of these orderings, but not all, are proven to lead to converging methods [21]. That is, M converges to a diagonal matrix after sufficiently many cycles. A common ordering, and the one that we will focus on, is the row-cyclic ordering $(1, 2)$, $(1, 3)$, $\ldots$, $(1, l)$, $(2, 3)$, $(2, 4)$, $\ldots$, $(l-1, l)$. A row sweep is what we call a series of transformations that annihilate all the off-diagonal elements in a single row. During a cycle of sweeps in a row-cyclic ordering, the ith row sweep produces fill-in in the ith column, so that a full cycle turns the initially upper-triangular matrices into lower-triangular matrices. In the following cycle, we effectively consider the triplet (A^T, C^T, B^T) as the input, which recovers the upper-triangular structure of the matrices. This leads to a sequence of alternating odd and even cycles that we repeat either until convergence, or until we reach a predefined maximum number of cycles.

To see how we can implicitly work with M, suppose that (i, j) is our pivot and that we want to annihilate m_{ij}. At this point in a cycle with row-cyclic ordering, A and A^{-1} have the form

$$A = \begin{array}{c} \\ i-1 \\ j-i+1 \\ l-j \end{array} \begin{array}{c} \begin{array}{ccc} i-1 & j-i+1 & l-j \end{array} \\ \begin{bmatrix} A_{11} & 0 & 0 \\ A_{21} & A_{22} & A_{23} \\ A_{31} & 0 & A_{33} \end{bmatrix} \end{array} \quad \text{and} \quad A^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 & 0 \\ \widetilde{A}_{21} & A_{22}^{-1} & \widetilde{A}_{23} \\ \widetilde{A}_{31} & 0 & A_{33}^{-1} \end{bmatrix}$$

for appropriate $\widetilde{A}_{21}$, $\widetilde{A}_{31}$, $\widetilde{A}_{23}$, where A_{11} is lower triangular and A_{33} upper triangular. Furthermore, for some vector $\boldsymbol{a}$ with $\boldsymbol{e}_{j-i}^T \boldsymbol{a} = 0$ and upper-triangular R_A with $\boldsymbol{e}_{j-i}^T R_A = a_{jj} \boldsymbol{e}_{j-i}^T$, we have that

$$A_{22} = \begin{bmatrix} a_{ii} & a_{ij} \boldsymbol{e}_{j-i}^T \\ \boldsymbol{a} & R_A \end{bmatrix} \quad \text{and} \quad A_{22}^{-1} = \frac{1}{a_{ii} a_{jj}} \begin{bmatrix} a_{jj} & -a_{ij} \boldsymbol{e}_{j-i}^T \\ -a_{jj} R_A^{-1} \boldsymbol{a} & a_{ii} a_{jj} R_A^{-1} + a_{ij} R_A^{-1} \boldsymbol{a} \boldsymbol{e}_{j-i}^T \end{bmatrix}.$$

Since B and C have the same structure as A, we can partition their blocks identically and use a similar notation for the blocks B_{22} and C_{22}. If we now ignore the previous subscript indices of the matrix blocks and define $M_{ij} = \left[\begin{smallmatrix} m_{ii} & m_{ij} \\ 0 & m_{jj} \end{smallmatrix}\right]$ and A_{ij}, B_{ij}, and C_{ij} likewise, then we can check that

$$\begin{aligned} M_{ij} &= \frac{1}{a_{ii} a_{jj}} \begin{bmatrix} c_{ii} & c_{ij} \boldsymbol{e}_{j-i}^T \\ 0 & c_{jj} \boldsymbol{e}_{j-i}^T \end{bmatrix} \begin{bmatrix} a_{jj} & -a_{ij} \boldsymbol{e}_{j-i}^T \\ -a_{jj} R_A^{-1} \boldsymbol{a} & a_{ii} a_{jj} R_A^{-1} + a_{ij} R_A^{-1} \boldsymbol{a} \boldsymbol{e}_{j-i}^T \end{bmatrix} \begin{bmatrix} b_{ii} & b_{ij} \\ \boldsymbol{b} & R_B \boldsymbol{e}_{j-i} \end{bmatrix} \\ &= \frac{1}{a_{ii} a_{jj}} \begin{bmatrix} c_{ii} a_{jj} b_{ii} & c_{ii} a_{jj} b_{ij} + (c_{ij} a_{ii} - c_{ii} a_{ij}) b_{jj} \\ 0 & c_{jj} a_{ii} b_{jj} \end{bmatrix} \\ &= C_{ij} A_{ij}^{-1} B_{ij}. \end{aligned}$$

We can even replace the inverse A_{ij}^{-1} by the adjugate matrix $\operatorname{adj}(A_{ij})$, because the scaling of M_{ij} does not matter when computing the rotations. Thus, we will henceforth define

$$M_{ij} = C_{ij} \operatorname{adj}(A_{ij}) B_{ij} = \begin{bmatrix} c_{ii} & c_{ij} \\ 0 & c_{jj} \end{bmatrix} \begin{bmatrix} a_{jj} & -a_{ij} \\ 0 & a_{ii} \end{bmatrix} \begin{bmatrix} b_{ii} & b_{ij} \\ 0 & b_{jj} \end{bmatrix}, \tag{9}$$

while stressing that this definition is only correct when annihilating m_{ij}.

Computing M_{ij} is the first step to computing the rotations that annihilate the (i, j)th elements. The second step is to compute an SVD $V^T M_{ij} U = \operatorname{diag}(\sigma_1, \sigma_2)$, where σ_1 and σ_2 are real, and

$U = \mathrm{rot}(\phi)$ and $V = \mathrm{rot}(\psi)$ for appropriate angles ϕ, ψ, and $\mathrm{rot}(\theta)$ denotes the rotation matrix $\left[\begin{smallmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{smallmatrix}\right]$. This SVD may be unnormalized, which means that its singular values are not necessarily nonnegative, nor sorted by magnitude. The next step is to compute rotations P and Q such that $V^T C_{ij} Q$, $P^T A_{ij} Q$, $P^T B_{ij} U$ are lower triangular. For the final step, let P_{ij}, Q_{ij}, U_{ij}, and V_{ij} be identity matrices with the (i,i), (i,j), (j,i), and (j,j) elements replaced by the $(1,1)$, $(1,2)$, $(2,1)$, and $(2,2)$ elements of P, Q, U, and V, respectively, and compute the transformations $P_{ij}^T A Q_{ij}$, $P_{ij}^T B U_{ij}$, and $V_{ij}^T C Q_{ij}$ as in (8). Accumulating the matrices P_{ij}, Q_{ij}, U_{ij}, and V_{ij} is optional, but necessary if we need the restricted singular vectors. See Algorithm 1 for a summary of the procedure.

Algorithm 1 (An implicit Kogbetliantz iteration for the RSVD)**.**
Input: Square and upper-triangular $l \times l$ matrices A, B, C, and A nonsingular.
Output: P, Q, U, V, A', B', and C' such that $A' = P^T A Q$, $B' = P^T B U$, and $C' = V^T C Q$ are upper triangular and $V^T C A^{-1} B U$ is diagonal.

1. **while** *# cycles is odd* **or** (*# cycles < max cycles* **and** *not converged*) **do**
2. **for** $i = 1, 2, \ldots, l-1$ **do**
3. **for** $j = i+1, i+2, \ldots, l$ **do**
4. Select A_{ij}, B_{ij}, and C_{ij} as outlined in the text.
5. In odd cycles: set $(A_{ij}, B_{ij}, C_{ij}) = (A_{ij}^T, C_{ij}^T, B_{ij}^T)$.
6. Compute P_{ij}, Q_{ij}, U_{ij}, and V_{ij} from A_{ij}, B_{ij}, and C_{ij}.
7. In odd cycles: swap P_{ij} with Q_{ij} and U_{ij} with V_{ij}.
8. Update $A = P_{ij}^T A Q_{ij}$, $B = P_{ij}^T B U_{ij}$, $C = V_{ij}^T C Q_{ij}$.
9. Accumulate $P = P P_{ij}$, $Q = Q Q_{ij}$, $U = U U_{ij}$, and $V = V V_{ij}$.

Forsythe and Henrici [19] prove that row-cyclic sweeps lead to (fast) convergence when a fixed closed interval within $(-\pi/2, \pi/2)$ contains all angles ϕ and ψ. Since this condition is impossible to guarantee while simultaneously diagonalizing M_{ij} exactly, Forsythe and Henrici also prove that a set of weaker requirements suffice for linear convergence. The benefit of these weaker requirements is that they are almost always satisfied in floating-point arithmetic. In any case, Heath et al. [22, Sec. 3] argue for the use of an unnormalized SVD as it simplifies the algorithm and they found it to be just as effective. This observation relies on the fact that Forsythe and Henrici's convergence proof only considers the magnitude of the matrix entries. In practical term this means that we may work with $-U$ or $-V$ instead of U and V, and thus, also with half period shifts and angles in a fixed closed interval of $(\pi/2, 3/2\pi)$. In other words, the angles just need to stay away from an open interval around $\pm\pi/2$.

Still, Brent, Luk, and Van Loan [6, Sec. 4] conjecture "that the smaller the rotation angles are the faster the procedure will converge". One way to adjust the angles is with a quarter period shift; that is, by replacing U and V with UJ and VJ, respectively, where $J = \mathrm{rot}(\pi/2) = \left[\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right]$. For example, the routine xLAGS2 of the current version of LAPACK[3] compute the upper-triangular 2×2 SVDs with xLASV2, and ensures that $\min\{|\phi|, |\psi|\} \leq \pi/4$ in essence by multiplying U and V with J if $|u_{11}| < |u_{12}|$ or $|v_{11}| < |v_{12}|$. Since this condition appears suboptimal if, say, $|u_{11}| > |u_{12}|$ and $|v_{11}| < |v_{12}|$, we will instead try to minimize the maximum angle. That is, we will replace U and V with UJ and VJ, respectively, if and only if $\max\{|u_{11}|, |v_{11}|\} < \max\{|u_{12}|, |v_{12}|\}$. This strategy ensures that we both have $\max\{|u_{11}|, |v_{11}|\} \geq 1/\sqrt{2}$ and $\max\{|u_{11}|, |v_{11}|\} \geq \max\{|u_{12}|, |v_{12}|\}$, although a downside is that we cannot always guarantee a particular ordering of the singular values during the cycles. However, this can also not be guaranteed with other conditions that stay away from the rotation angles $\pm\pi/2$.

A standard approach to check for convergence is to define $\rho_{ij} = 0$ if $M_{ij} = 0$ and $\rho_{ij} = |m_{ij}|/\|M_{ij}\|_{\max} \leq 1$ otherwise, and to stop if all $\rho_{ij} < \tau$ for some tolerance τ. Another option, suggested by Demmel and Veselić [13], is to use $\rho_{ij} = |m_{ij}|(|m_{ii}||m_{jj}|)^{-1/2} \leq \infty$ (if $M_{ij} \neq 0$)

[3] Version 3.8.0 at the time of writing.

instead. The problem for implicit Kogbetliantz-type iterations with both of these definitions of ρ_{ij}, is that $|m_{ij}|$ may not become "small" enough in floating-point arithmetic for the stopping criterion to be fulfilled. This unfortunate discrepancy between theory and practice exists, at least in part, because the implicit Kogbetliantz method forms each M_{ij} on-the-fly. This means that the relative error in the computed $|m_{ij}|$ can be of order $\mathcal{O}(\boldsymbol{\varepsilon}\|A_{ij}\|\|B_{ij}\|\|C_{ij}\|)$, where $\boldsymbol{\varepsilon}$ is the unit roundoff, rather than $\mathcal{O}(\boldsymbol{\varepsilon}\|M_{ij}\|)$, even if the former is often pessimistic. Hence, a τ picked based on the former may be too large, and a τ picked based on the latter may be too small. This does not even take other sources of roundoff errors into account yet, such as, for example, perturbations in the computed rotations and the roundoff errors from the application of the rotations.

Bai and Demmel [4, Sec. 4] use a different approach and measure the parallelism between corresponding rows of two matrices A and B. The theoretical justification is simple: when all corresponding rows of A and B are parallel, then there must exist diagonal matrices C and S and an upper-triangular matrix R such that $A = CR$ and $B = SR$. This justification and the corresponding implementation are appealing, but the generalization to matrix triplets and the RSVD is not obvious. A simplified approach without a similar theoretical justification is to consider the angle between two-dimensional vectors and if $m_{ij} \neq 0$ to take

$$\rho_{ij} = \max\left\{ \frac{|\boldsymbol{e}_1^T C_{ij} \operatorname{adj}(A_{ij}) B_{ij} \boldsymbol{e}_2|}{\|\boldsymbol{e}_1^T C_{ij}\| \|\operatorname{adj}(A_{ij}) B_{ij} \boldsymbol{e}_2\|}, \frac{|\boldsymbol{e}_1^T C_{ij} \operatorname{adj}(A_{ij}) B_{ij} \boldsymbol{e}_2|}{\|\boldsymbol{e}_1^T C_{ij} \operatorname{adj}(A_{ij})\| \|B_{ij} \boldsymbol{e}_2\|} \right\} \tag{10}$$

for each pair of i and j. Although the relative scaling is still not ideal because the roundoff errors in $|m_{ij}|$ may be as big as $\mathcal{O}(\boldsymbol{\varepsilon}\|\boldsymbol{e}_1^T C_{ij}\|\|A_{ij}\|\|B_{ij}\boldsymbol{e}_2\|)$, this ρ_{ij} strikes a balance that appears to work well in our limited testing.

Regardless of the choice of ρ_{ij}, we may want to stop iterating before convergence when progress is too slow and before reaching a predefined maximum number of cycles. To decide on this, one option is to compute $\rho = \max_{i,j} \rho_{ij}$ during each cycle, let $\rho_{\min}$ be the smallest ρ of all previous cycles, and stop iterating (after an even number of cycles) if (after an even number of cycles) if $\rho_{\min} \lesssim \rho \ll 1$. That is, stop when both ρ and the improvement between cycles are small.

4.2 The 2-by-2 RSVD in exact arithmetic. Algorithm 1 does not tell us how to compute the RSVD of (upper-)triangular 2-by-2 matrices. But this is an easier problem to solve than computing the RSVD of larger matrices. See, for example, the theorem below.

Proposition 10. *Let A, B, and C be arbitrary 2×2 upper-triangular matrices, and define $M = C \operatorname{adj}(A) B$; then there exist orthonormal matrices P, Q, U, and V, such that $P^T A Q$, $P^T B U$, and $V^T C Q$ are lower triangular, and $V^T M U = \Sigma$ is diagonal.*

Proof. If any two of the three matrices A, B, and C are nonsingular, then the result is straightforward. For example, if B and C are nonsingular, then we can find U and V by computing the SVD of M, and letting P and Q zero the $(1,2)$ entries of BU and $V^T C$, respectively. Then $Q^T \operatorname{adj}(A) P = (V^T C Q)^{-1} \Sigma (P^T B U)^{-1}$ and it follows that $P^T A Q$ must be lower triangular. By noting that $\operatorname{adj}(A)$ is nonsingular if and only if A is nonsingular, we see that similar arguments hold when A and B are nonsingular or when A and C are nonsingular.

If B is singular and A and C are arbitrary, we can compute P and U such that $P^T B U$ has the form $\left[\begin{smallmatrix} 0 & 0 \\ 0 & \times \end{smallmatrix}\right]$, compute Q so that $P^T A Q$ is lower triangular, and compute V so that $V^T C Q$ is lower triangular. By using the fact that $\operatorname{adj}(P^T A Q)$ is a scalar multiple of $P^T \operatorname{adj}(A) Q$, we then see that $V^T M U$ is a scalar multiple of $(V^T C Q) \cdot \operatorname{adj}(P^T A Q) \cdot (P^T B U)$, which is of the form $\left[\begin{smallmatrix} 0 & 0 \\ 0 & \times \end{smallmatrix}\right]$.

If C is singular and A and B are arbitrary, we can compute V and Q such that $V^T C Q$ is of the form $\left[\begin{smallmatrix} \times & 0 \\ 0 & 0 \end{smallmatrix}\right]$, compute P so that $P^T A Q$ is lower triangular, and compute U so that $P^T B U$ is lower triangular. Then $V^T M U$ is a scalar multiple of $(V^T C Q) \cdot \operatorname{adj}(P^T A Q) \cdot (P^T B U)$, which is of the form $\left[\begin{smallmatrix} \times & 0 \\ 0 & 0 \end{smallmatrix}\right]$. □

The theorem above does not tell us anything about the angles of the rotations, nor about the numerical stability of the computations. Rather, the theorem shows that computing the 2×2 RSVD is possible for any triplet of upper-triangular matrices, even when A is singular. Knowing what is possible, the question that remains is how to do it in a numerically sound way.

Bojanczyk et al. [5] propose a recursive algorithm for accurately computing the SVD of a product of three upper-triangular 2×2 matrices that is close to the 2×2 RSVD needed for Algorithm 1. But their diagonalization is not guaranteed to have high *relative* accuracy, as demonstrated by Bai and Demmel [4] for the QSVD. Adams, Bojanczyk, and Luk address this issue for the product of two matrices in [1] with a modified version of their algorithm that they call "half-recursive", and which they show is related to Bai and Demmel's algorithm in exact arithmetic. Though, they did not provide an improved version of their algorithm for the product of three matrices.

We can generalize Bai and Demmel's algorithm for the 2×2 QSVD to the 2×2 RSVD, as shown below in Algorithm 2. Informally, the idea of the algorithm is to apply a modified version of Bai and Demmel's GSVD22 to the pairs $(C, \operatorname{adj}(A)B)$ and $(C \operatorname{adj}(A), B)$, but some of the details require further attention. For example, what to do when $c_{11} = b_{22} = 0$, and when to replace U and V by UJ and VJ. For the latter in particular, there are qualitative differences between postmultiplying by J when $c_{\max} < s_{\max}$ or when $c_{\max} \leq s_{\max}$ if B or C are singular, and the choice between the two conditions is not obvious. The condition we ultimately use in the algorithm below ensures that Lemma 11 and Lemma 21 hold.

Algorithm 2 (2×2 upper-triangular RSVD (RSVD22))**.**
Input: 2×2 upper-triangular matrices A, B, and C, with A nonsingular.
Output: Orthonormal matrices P, Q, U, and V, and lower-triangular matrices $A' = P^T A Q$, $B' = P^T B U$, and $C' = V^T C Q$, such that $C' \operatorname{adj}(A') B'$ is diagonal.

1. **if** $c_{11} = 0$ and $b_{22} = 0$ **then**
2. Compute V such that $(V^T C)_{22} = 0$ and let $Q = J$.
3. Compute U such that $(BU)_{11} = 0$ and let $P = J$.
4. Let $A' = P^T A Q$, $B' = P^T B U$, $C' = V^T C Q$, and $b'_{21} = c'_{21} = 0$.
5. **return**
6. **endif**
7. Use xLASV2 to compute $M = V \Sigma U^T$, where $M = C \operatorname{adj}(A) B$.
8. Define $c_{\max} = \max\{|u_{11}|, |v_{11}|\}$ and $s_{\max} = \max\{|u_{12}|, |v_{12}|\}$.
9. **if** $c_{11} \neq 0$ and $c_{22} \neq 0$ and $b_{11} \neq 0$ and $b_{22} \neq 0$ and $c_{\max} < s_{\max}$ **then**
10. Let $U = UJ$ and $V = VJ$.
11. **endif**
12. Let $G = V^T C$ and (optionally; see text) set g_{22} to zero if $c_{11} = 0$.
13. Let $L = BU$ and (optionally; see text) set l_{12} to zero if $b_{22} = 0$.
14. Let $\widehat{G} = |V|^T |C|$, $H = \operatorname{adj}(A) L$, and $\widehat{H} = |\operatorname{adj}(A)|\, |B|\, |U|$.
15. Let $K = G \operatorname{adj}(A)$, $\widehat{K} = |V|^T |C|\, |\operatorname{adj}(A)|$, and $\widehat{L} = |B|\, |U|$.
16. Let $\eta_g = (\widehat{g}_{11} + \widehat{g}_{12})/(|g_{11}| + |g_{12}|)$ and $\eta_h = (\widehat{h}_{12} + \widehat{h}_{22})/(|h_{12}| + |h_{22}|)$.
17. Let $\eta_k = (\widehat{k}_{11} + \widehat{k}_{12})/(|k_{11}| + |k_{12}|)$ and $\eta_l = (\widehat{l}_{12} + \widehat{l}_{22})/(|l_{12}| + |l_{22}|)$.
18. **if** $|h_{12}| + |h_{22}| = 0$ or ($|g_{11}| + |g_{12}| \neq 0$ and $\eta_g \leq \eta_h$) **then**
19. Use xLARTG to compute Q such that GQ is lower triangular.
20. **else**
21. Use xLARTG to compute Q such that $Q^T H$ is lower triangular.
22. **endif**
23. **if** $|k_{11}| + |k_{12}| = 0$ or ($|l_{12}| + |l_{22}| \neq 0$ and $\eta_l \leq \eta_k$) **then**
24. Use xLARTG to compute P such that $P^T L$ is lower triangular.
25. **else**
26. Use xLARTG to compute P such that KP is lower triangular.

27. **endif**
28. Let $A' = P^T A Q$, $B' = P^T L$, $C' = GQ$, and $a'_{12} = b'_{12} = c'_{12} = 0$.

Since B and C can be singular, there may be zeros on their diagonals. If this is the case, and if the factors X and Y from Theorem 1 or the factors S and T from Corollary 4 are desired, then we need to know the nonzero structure of B and C after convergence. Paige [26, Sec. 5] describes the nonzero structure for the QSVD in a similar case, and has a proof which is, in his own words, "hard going". The proof for the RSVD is tedious also, and is split into two parts. The first part is a lemma that gives the output of RSVD22 for a given input, and the second part is a proposition that uses the lemma to prove what kind of nonzero structure we get for the RSVD after a series of cycles.

In principle, we have to consider a total of 25 different cases when investigating the nonzero structure of the outputs of Algorithm 2. For B alone, for instance, we must already consider the following five cases:

$$\begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}, \quad \begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}, \quad \begin{bmatrix} 0 & \underline{b_{12}} \\ 0 & 0 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

where the underlined entries are nonzero. Fortunately, we can treat some of the 25 cases simultaneously and reduce them to 13 cases.

Lemma 11. *Let A, B, and C be upper-triangular 2×2 matrices, and suppose that A is nonsingular. If the SVD in Algorithm 2 computes $U = V = I$ whenever $M = 0$, and is such that $|\sigma_1| \geq |\sigma_2|$; then the cases given below describe the output of Algorithm 2. Each case shows (in sequence) the structure of the input matrices B and C (A is always upper triangular with nonzero diagonal entries), the matrix $M = C\,\mathrm{adj}(A)B$, and the output matrices B' and C' (A' is always lower triangular with nonzero diagonal entries). Each case also shows P, Q, U, and V when they take specific values. Furthermore, underlined matrix entries are nonzero.*

1. $\begin{bmatrix} \underline{c_{11}} & c_{12} \\ 0 & \underline{c_{22}} \end{bmatrix}, \begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}, \begin{bmatrix} \underline{m_{11}} & m_{12} \\ 0 & \underline{m_{22}} \end{bmatrix}, \begin{bmatrix} \underline{c'_{11}} & 0 \\ c'_{21} & \underline{c'_{22}} \end{bmatrix}, \begin{bmatrix} \underline{b'_{11}} & 0 \\ b'_{21} & \underline{b'_{22}} \end{bmatrix}.$
2. $\begin{bmatrix} 0 & \underline{c_{12}} \\ 0 & c_{22} \end{bmatrix}, \begin{bmatrix} b_{11} & \underline{b_{12}} \\ 0 & 0 \end{bmatrix}, 0, \begin{bmatrix} \underline{c'_{11}} & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & \underline{b'_{22}} \end{bmatrix}, P = Q = J.$
3. $\begin{bmatrix} \underline{c_{11}} & c_{12} \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, 0, \begin{bmatrix} \underline{c'_{11}} & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, U = V = I.$
4. $\begin{bmatrix} \underline{c_{11}} & c_{12} \\ 0 & \underline{c_{22}} \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, 0, \begin{bmatrix} \underline{c'_{11}} & 0 \\ c'_{21} & \underline{c'_{22}} \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, U = V = I.$
5. $\begin{bmatrix} \underline{c_{11}} & c_{12} \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & 0 \end{bmatrix} \neq 0, c_{11}a_{22}B \neq 0, \begin{bmatrix} \underline{c'_{11}} & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \underline{b'_{11}} & 0 \\ b'_{21} & 0 \end{bmatrix}, \text{and } |V| = I.$
6. $\begin{bmatrix} \underline{c_{11}} & c_{12} \\ 0 & \underline{c_{22}} \end{bmatrix}, \begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & 0 \end{bmatrix} \neq 0, c_{11}a_{22}B \neq 0, \begin{bmatrix} \underline{c'_{11}} & 0 \\ c'_{21} & \underline{c_{22}'} \end{bmatrix}, \begin{bmatrix} \underline{b'_{11}} & 0 \\ b'_{21} & 0 \end{bmatrix}, \text{and } |V| = I.$
7. $\begin{bmatrix} \underline{c_{11}} & c_{12} \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}, \begin{bmatrix} \underline{m_{11}} & m_{12} \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \underline{c'_{11}} & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \underline{b'_{11}} & 0 \\ b'_{21} & \underline{b'_{22}} \end{bmatrix}, |U| \neq |J| \text{ and } |V| = I.$
8. $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & \underline{b_{12}} \\ 0 & \underline{b_{22}} \end{bmatrix}, 0, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & \underline{b'_{22}} \end{bmatrix}, U = V = I.$
9. $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}, 0, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \underline{b'_{11}} & 0 \\ b'_{21} & \underline{b'_{22}} \end{bmatrix}, U = V = I.$
10. $\begin{bmatrix} 0 & \underline{c_{12}} \\ 0 & c_{22} \end{bmatrix} \neq 0, \begin{bmatrix} 0 & \underline{b_{12}} \\ 0 & \underline{b_{22}} \end{bmatrix}, a_{11}b_{22}C \neq 0, \begin{bmatrix} \underline{c'_{11}} & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \underline{b'_{11}} & 0 \\ b'_{21} & 0 \end{bmatrix}, \text{and } |P| = |Q| = |U| = |J|.$
11. $\begin{bmatrix} 0 & \underline{c_{12}} \\ 0 & c_{22} \end{bmatrix} \neq 0, \begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}, a_{11}b_{22}C \neq 0, \begin{bmatrix} \underline{c'_{11}} & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \underline{b'_{11}} & 0 \\ b'_{21} & \underline{b'_{22}} \end{bmatrix}, \text{and } |P| = |Q| = |U| = |J|.$

12. $\left[\begin{smallmatrix} c_{11} & c_{12} \\ 0 & \underline{c_{22}} \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & b_{12} \\ 0 & \underline{b_{22}} \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & m_{12} \\ 0 & \underline{m_{22}} \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} c'_{11} & 0 \\ c'_{21} & \underline{c'_{22}} \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} b'_{11} & 0 \\ b'_{21} & 0 \end{smallmatrix}\right]$, $|U| = |J|$ *and* $|V| \neq I$.

13. $\left[\begin{smallmatrix} c_{11} & c_{12} \\ 0 & 0 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & b_{12} \\ 0 & \underline{b_{22}} \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & m_{12} \\ 0 & 0 \end{smallmatrix}\right]$*; if* $m_{12} = 0$*, then* $C' = \left[\begin{smallmatrix} c'_{11} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ *and* $B' = \left[\begin{smallmatrix} 0 & 0 \\ 0 & \underline{b'_{22}} \end{smallmatrix}\right]$ *with* $U = V = I$*. If* $m_{12} \neq 0$*, then* $C' = \left[\begin{smallmatrix} c'_{11} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ *and* $B' = \left[\begin{smallmatrix} b'_{11} & 0 \\ b_{21} & 0 \end{smallmatrix}\right]$ *with* $|U| = |J|$ *and* $|V| = I$.

Proof. Following Algorithm 2 step-by-step for each case yields the desired results; see Appendix A for details. □

Cases 10, 11, and 12 may violate the min-max–angle condition; however, these cases can no longer occur in later iterations. This is a consequence of the following proposition, which describes the nonzero structure of A, B, and C after an odd and an even cycle. Furthermore, the positions of the zeros and nonzeros in the output of Lemma 11 are only guaranteed in floating-point arithmetic with the optional zeroing of g_{22} if $c_{11} = 0$ and l_{12} if $b_{22} = 0$ in Algorithm 2. Still, the theoretical results from later sections hold with or without this explicit zeroing; see Section 7.2 for further discussion.

Proposition 12. *Suppose that A, B, and C are square and upper-triangular $l \times l$ matrices, A is nonsingular, and $M = CA^{-1}B = \left[\begin{smallmatrix} M_{11} & M_{12} \\ 0 & 0 \end{smallmatrix}\right]$, where M_{11} is nonsingular and upper triangular. Then a pair of an odd and even cycle of Algorithm 1, with Algorithm 2 for the 2×2 RSVDs, transforms the structure of A, B, and C into*

$$(11)\quad \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ & A_{22} & A_{23} & A_{24} \\ & & A_{33} & A_{34} \\ & & & A_{44} \end{bmatrix}, \quad \begin{bmatrix} B_{11} & B_{12} & B_{13} & B_{14} \\ & B_{22} & B_{23} & B_{24} \\ & & 0 & 0 \\ & & & 0 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ & 0 & 0 & C_{24} \\ & & 0 & C_{34} \\ & & & C_{44} \end{bmatrix},$$

where all nonzero diagonal blocks are nonsingular and upper triangular.

Proof. Any triplet of 1×1 matrices A, B, and C satisfies (11) when A is nonsingular. For larger matrices, the upper triangularity of the matrices is a result of the row-cyclic cycles. Now, let $\mathrm{kog}(A, B, C)$ denote the output of a single odd cycle of Algorithm 1, and let $A^{(0)} = A$, $B^{(0)} = B$, and $C^{(0)} = C$; then we can write the desired pair of cycles as

$$(A^{(0.5)}, B^{(0.5)}, C^{(0.5)}) = \mathrm{kog}(A^{(0)}, B^{(0)}, C^{(0)}) \quad (A^{(1)}, B^{(1)}, C^{(1)}) = (A^{(0.5)T}, C^{(0.5)T}, B^{(0.5)T})$$
$$(A^{(1.5)}, B^{(1.5)}, C^{(1.5)}) = \mathrm{kog}(A^{(1)}, B^{(1)}, C^{(1)}) \quad (A^{(2)}, B^{(2)}, C^{(2)}) = (A^{(1.5)T}, C^{(1.5)T}, B^{(1.5)T}).$$

Define $M^{(\ell)} = C^{(\ell)}(A^{(\ell)})^{-1}B^{(\ell)}$ for $\ell = 0, 0.5, 1, \ldots, 2$, and for any M, $j > i$, and just before annihilating m_{ij}, define $M_{ij} = \left[\begin{smallmatrix} m_{ii} & m_{ij} \\ 0 & m_{jj} \end{smallmatrix}\right]$ as in (9). Moreover, we assume for the rest of the proof that the indices i, j, and k are always such that $1 \leq i, j, k \leq l$.

We start by proving that sweeping the first row of $M = M^{(0)}$ transforms

$$M = \begin{bmatrix} m_{11} & \times & \times \\ 0 & M_{22} & \times \\ 0 & 0 & 0 \end{bmatrix} \quad \text{into} \quad \widetilde{M} = \begin{bmatrix} \widetilde{m}_{11} & 0 & 0 \\ \times & \widetilde{M}_{22} & \times \\ \times & 0 & 0 \end{bmatrix},$$

where $m_{11} \neq 0$ and $\widetilde{m}_{11} \neq 0$, and M_{22} and $\widetilde{M}_{22}$ are nonsingular and upper triangular and not to be confused with M_{ij}. Now suppose that we are about to annihilate m_{1j} for some $j > 1$; then we have the following.

1. If $m_{11} \neq 0$ and $m_{jj} \neq 0$, then both remain nonzero after annihilating m_{1j}. This follows from Case 1 of Lemma 11. Moreover, the rotations only transform the nonsingular diagonal block of M, which preserves the desired block structure.

2. If $m_{11} \neq 0$ and $m_{jj} = 0$, then m_{11} stays nonzero and we get $|V| = I$. This follows from Cases 5, 6, and 7 of Lemma 11. Since $|V| = I$, the fact that $m_{jk} = 0$ for every $k \geq j$ remains true after annihilating m_{1j}; that is, the transformations do not introduce nonzeros in row j.

3. If $m_{11} = m_{jj} = 0$, then $M = 0$ and M is still the zero matrix after annihilating m_{1j}.

An induction argument shows that after sweeping the ith row of M, the unswept trailing submatrix starting at the $(i+1, i+1)$th element has a block structure similar to $M^{(0)}$. Moreover, since $M^{(1)} = M^{(0.5)T}$ we see that $M^{(1)}$ has the same block structure as $M^{(0)}$, and that the same is true for the structure of $M^{(2)}$.

Next we will prove that after the first cycle

$$B^{(0.5)} = \begin{bmatrix} B_{11}^{(0.5)} & & & \\ B_{21}^{(0.5)} & 0 & & \\ B_{31}^{(0.5)} & 0 & 0 & \\ B_{41}^{(0.5)} & B_{42}^{(0.5)} & B_{43}^{(0.5)} & B_{44}^{(0.5)} \end{bmatrix},$$

where $B_{11}^{(0.5)}$ and $B_{44}^{(0.5)}$ are nonsingular. That is, if $b_{ii}^{(0.5)} = b_{kk}^{(0.5)} = 0$, then $b_{jj}^{(0.5)} = b_{ji}^{(0.5)} = 0$ for all $i \leq j \leq k$. Now, let us drop the superscript indices as we consider the row-cyclic sweeps that transform the input $A^{(0)}, B^{(0)}, C^{(0)}$, and their corresponding $M^{(0)}$ to the output $A^{(0.5)}, B^{(0.5)}, C^{(0.5)}$, and their corresponding $M^{(0.5)}$. Furthermore, suppose that we have swept $i-1$ rows, that i is such that $m_{ii} = 0$, and that we are about to annihilate m_{ij} for some $j > i$.

1. As a result of the structure of M, we have that $M_{ij} = 0$ and we cannot have Cases 1, 5, 6, 7, 10, 11, and 12 of Lemma 11. Moreover, Case 13 can only occur with $m_{ij} = 0$.

2. If $b_{ii} = 0$ or becomes zero after annihilating m_{ij}, then $b_{ii} = b_{ji} = 0$. This follows from Cases 2, 3, 4, 8, and 13 of Lemma 11.

3. If $b_{jj} \neq 0$, then b_{jj} stays nonzero; that is, b_{jj} stays nonzero at least until the jth row sweep. This follows from Cases 8 and 9 of Lemma 11.

4. Suppose that $b_{ii} \neq 0$ at the start of the ith row sweep, and that j is the first $j > i$ such that $b_{jj} = 0$. Then it follows from Case 2 of Lemma 11 that b_{ii} becomes zero and b_{jj} nonzero after annihilating m_{ij}. When this happens, b_{ii} remains zero for the rest of the row sweep, and thus for the rest of the cycle, and b_{jj} remains nonzero at least until the jth row sweep. We have two possibilities before we annihilate m_{ij} that we must consider. Either $b_{i_0 i_0} \neq 0$ for every $1 \leq i_0 < i$, in which case we are done since b_{ii} becomes the first zero on the diagonal of B, or there exists some $1 \leq i_0 < i$ such that $b_{i_0 i_0} = 0$. In the latter case, $b_{i_0 i_0}$ must have been zero at the start of the current cycle, or must have become zero before annihilating $b_{i_0 j}$. This follows from the previous two points, which imply that the algorithm would otherwise have made b_{jj} nonzero when annihilating $b_{i_0 j}$ during the i_0th row sweep, and that b_{jj} would have stayed nonzero at least until the jth row sweep. Hence, we can conclude that $b_{ji_0} = 0$ before annihilating m_{ij}, and since Case 2 gives $|P| = |J|$, that $b_{ii_0} = 0$ after annihilating m_{ij}.

5. As a result of the previous point, if $b_{ii} \neq 0$ after sweeping the ith row, then $b_{jj} \neq 0$ for all $j > i$.

For the second cycle we need to prove that

$$C^{(1.5)} = \begin{bmatrix} C_{11}^{(1.5)} & & & \\ C_{21}^{(1.5)} & C_{22}^{(1.5)} & & \\ C_{31}^{(1.5)} & C_{32}^{(1.5)} & 0 & \\ C_{41}^{(1.5)} & C_{42}^{(1.5)} & 0 & 0 \end{bmatrix},$$

where $C_{11}^{(1.5)}$ and $C_{22}^{(1.5)}$ are nonsingular. That is, if $c_{ii}^{(1.5)} = 0$, then $c_{jj}^{(1.5)} = c_{ji}^{(1.5)} = 0$ for all $i \le j \le l$. Due to the previous sweeps we may assume that if $c_{ii}^{(1)} = c_{kk}^{(1)} = 0$, then $c_{jj}^{(1)} = c_{ij}^{(1)} = 0$ for all $i \le j \le k$. Now, let us again drop the superscript indices as we consider the matrices during the row-cyclic sweeps, and suppose that we have swept $i-1$ rows, that i is such that $m_{ii} = 0$, and that we are about to annihilate m_{ij} for some $j > i$.

1. As in the previous cycle, we have that $M_{ij} = 0$ and we cannot have Cases 1, 5, 6, 7, 10, 11, and 12 of Lemma 11. Moreover, Case 13 can only occur with $m_{ij} = 0$.

2. If $c_{ii} \neq 0$ or becomes nonzero when sweeping row i, then it stays nonzero for the rest of the sweep. This follows from Cases 3, 4, and 13 of Lemma 11.

3. If $c_{jj} = 0$ before annihilating m_{ij}, then it stays zero after. Furthermore, no nonzeros are introduced in the zero blocks of C. This follows from Cases 2, 3, 8, 9, and 13 of Lemma 11, while noting that we get $|V| = I$ in each case.

4. If $c_{ii} = 0$, then it becomes nonzero for the smallest integer $k > i$ such that $c_{ik} \neq 0$ or $c_{kk} \neq 0$. This follows from Case 2 of Lemma 11. Since it holds that $c_{jj} = c_{ij} = 0$ for all $i \le j < k$ before annihilating m_{ik}, and because we get $|Q| = |J|$, we must have $c_{ii} \neq 0$ and $c_{jj} = c_{ij} = 0$ for all $i < j \le k$ afterwards (note that the inequality symbols are swapped).

5. As a result of the previous point, if $c_{ii} = 0$ after sweeping the ith row, then $c_{ij} = c_{jj} = 0$ for all $j > i$ and the remaining row sweeps do not introduce nonzeros in column i below c_{ii}. □

The block structure of $A = S\Sigma_\alpha T$, $B = S\Sigma_\beta$, and $C = \Sigma_\gamma T$, for some upper-triangular S and T and block-diagonal $\Sigma_\alpha = \operatorname{diag}(D_\alpha, I, I, I)$, $\Sigma_\beta = \operatorname{diag}(D_\beta, I, 0, 0)$, and $\Sigma_\gamma = \operatorname{diag}(D_\gamma, 0, 0, I)$ is

$$(12) \quad \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ & A_{22} & A_{23} & A_{24} \\ & & A_{33} & A_{34} \\ & & & A_{44} \end{bmatrix}, \quad \begin{bmatrix} B_{11} & B_{12} & 0 & 0 \\ & B_{22} & 0 & 0 \\ & & 0 & 0 \\ & & & 0 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ & 0 & 0 & 0 \\ & & 0 & 0 \\ & & & C_{44} \end{bmatrix},$$

where the nonzero diagonal blocks are nonsingular. Hence, if we want to compute the factors S and T, then we must extend the Kogbetliantz phase to turn (11) into (12). The constructive proof of the proposition below shows how we can do so.

Proposition 13. *Let A, B, and C be structured as in* (11)*, and suppose that $CA^{-1}B$ equals $\left[\begin{smallmatrix} M_{11} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ for some nonsingular M_{11}. Then there exists orthonormal matrices U and V such that*

$$BU = \begin{bmatrix} B_{11} & B_{12}B_{22}^{-1}R_B & 0 & 0 \\ & R_B & 0 & 0 \\ & & 0 & 0 \\ & & & 0 \end{bmatrix} \quad \textit{and} \quad V^TC = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ & 0 & 0 & 0 \\ & & 0 & 0 \\ & & & R_C \end{bmatrix}.$$

where R_B and R_C are nonsingular and upper triangular.

Proof. Suppose $A = I$, then

$$CA^{-1}B = \begin{bmatrix} C_{11}B_{11} & C_{11}B_{12} + C_{12}B_{22} & C_{11}B_{13} + C_{12}B_{23} & C_{11}B_{14} + C_{12}B_{24} \\ & 0 & 0 & 0 \\ & & 0 & 0 \\ & & & 0 \end{bmatrix}$$

is block diagonal by the assumption on the structure of $CA^{-1}B$. It follows that $C_{12} = -C_{11}B_{12}B_{22}^{-1}$ and that $B_{1j} = -C_{11}^{-1}C_{12}B_{2j} = B_{12}B_{22}^{-1}B_{2j}$ for $j = 2$, 3, and 4. In other words,

$$B = \begin{bmatrix} B_{11} & B_{12}B_{22}^{-1}B_{22} & B_{12}B_{22}^{-1}B_{23} & B_{12}B_{22}^{-1}B_{24} \\ & B_{22} & B_{23} & B_{24} \\ & & 0 & 0 \\ & & & 0 \end{bmatrix}.$$

Hence, if $\widetilde{U}$ is such that $[B_{22}\ B_{23}\ B_{24}]\widetilde{U} = [R_B\ 0\ 0]$, then $U = \left[\begin{smallmatrix} I & 0 \\ 0 & \widetilde{U} \end{smallmatrix}\right]$ is the desired U. When $A \neq I$, the product CA^{-1} has the same block structure as C and the proof is similar. For C we can use a QR decomposition to compute a $\widetilde{V}$ such that $\widetilde{V}^T[C_{24};\ C_{34};\ C_{44}]$ has the form $[0;\ 0;\ R_C]$, so that $V = \left[\begin{smallmatrix} I & 0 \\ 0 & \widetilde{V} \end{smallmatrix}\right]$ is the sought after V. □

The assumption in Proposition 12 that $M = \left[\begin{smallmatrix} M_{11} & M_{12} \\ 0 & 0 \end{smallmatrix}\right]$ for some nonsingular M_{11} is not necessarily satisfied directly after the preprocessing phase from Section 3. If M does have this form, then M_{12} converges to zero if the implicit Kogbetliantz iteration converges, so that Proposition 13 applies. We suspect that a finite number of cycles from Algorithm 1 with Algorithm 2 for the 2×2 RSVDs will bring M into with the desired form; however, we could not come up with a proof yet. The reason for this suspicion is that Algorithm 2 computes rotations that move nonzero entries of M_{ij} to the upper-left corner if M_{ij} is singular.

In any case, we can ensure that M has the desired structure with the transformations that follow; though, this approach is only of theoretical interest when we want the factors S and T, and requires (at least) two more and unwanted rank decisions in floating-point arithmetic. We start with compressing B by comping $P^{(1)}$ and $U^{(1)}$ such that $B^{(2)} = P^TBU = \left[\begin{smallmatrix} B_{11}^{(2)} & 0 \\ 0 & 0 \end{smallmatrix}\right]$, where $B_{11}^{(2)}$ is nonsingular. Next, we compute $Q^{(1)}$ such that $A^{(2)} = P^{(1)T}AQ^{(1)}$ is upper triangular, $V^{(1)}$ such that $C^{(2)} = V^{(1)T}CQ^{(1)}$ is upper triangular, and partition both matrices into blocks with block sizes matching the blocks of $B^{(2)}$. Then, we compress $C_{11}^{(2)}$ by computing $V_{11}^{(2)}$ and $Q_{11}^{(2)}$ such that $V_{11}^{(2)T}C_{11}^{(2)}Q_{11}^{(2)} = \left[\begin{smallmatrix} C_{11}^{(3)} & 0 \\ 0 & 0 \end{smallmatrix}\right]$. Finally, we compute $P_{11}^{(2)}$ such that $P_{11}^{(2)T}A_{11}^{(2)}Q_{11}^{(2)}$ is upper triangular, and compute $U_{11}^{(2)}$ such that $P_{11}^{(2)T}B_{11}^{(2)}U_{11}^{(2)}$ is upper triangular. We can now partition the resulting $A^{(3)}$, $B^{(3)}$, and $C^{(3)}$ as

$$\begin{bmatrix} A_{11}^{(3)} & A_{12}^{(3)} & A_{13}^{(3)} \\ & A_{22}^{(3)} & A_{23}^{(3)} \\ & & A_{33}^{(3)} \end{bmatrix}, \quad \begin{bmatrix} B_{11}^{(3)} & B_{12}^{(3)} & 0 \\ & B_{22}^{(3)} & 0 \\ & & 0 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} C_{11}^{(3)} & 0 & C_{13}^{(3)} \\ & 0 & C_{23}^{(3)} \\ & & C_{33}^{(3)} \end{bmatrix},$$

respectively, from which we can see that $M^{(3)}$ has the desired structure. If desired, we can even get the structure from (12) without the Kogbetliantz iteration by computing

$$\begin{bmatrix} V_{22}^{(3)} & V_{23}^{(3)} \\ V_{32}^{(3)} & V_{33}^{(3)} \end{bmatrix}^T \begin{bmatrix} C_{23}^{(3)} \\ C_{33}^{(3)} \end{bmatrix} Q_{33}^{(3)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & C_{44}^{(4)} \end{bmatrix},$$

and by computing $P_{33}^{(3)}$ such that $P_{33}^{(3)T}A_{33}^{(3)}Q_{33}^{(3)}$ is upper triangular.

4.3 The 2-by-2 RSVD in floating point arithmetic. Thus far, we have only considered the 2×2 RSVD in exact arithmetic. The goal of this section is to show that Algorithm 2 computes a numerically stable result in floating-point arithmetic under the assumptions of the standard model from, e.g., Higham [23, Ch. 2] or the *LAPACK Users' Guide* [2, Sec. 4.1.1]. That is, given two floating-point numbers a and b, and some operation $\circ \in \{+, -, \cdot, /\}$, we assume that $\mathrm{fl}(a \circ b) = (a \circ b)(1+\epsilon)$, where $|\epsilon| \le \boldsymbol{\varepsilon}$ and $\boldsymbol{\varepsilon}$ is the unit roundoff (2^{-53} in case of IEEE 754 double precision arithmetic). We additionally assume that taking the absolute value of a floating-point number is exact, as well as multiplying by zero or ± 1. We ignore overflow, underflow, and higher-order terms, as usual, unless stated otherwise. For convenience, different occurrences of ϵ and error matrices do not need to have the same value unless they have subscript indices. Another convention is that overlined quantities denote the "computed" version of quantities; for example, if $c = a \circ b$, then $\overline{c} = \mathrm{fl}(a \circ b)$.

To prove the main results from this section, we first need the bounds from the following two lemmas. The first lemma bounds a sum of elements from the product of two particular nonnegative matrices. The second lemma bounds the norms of the backward perturbations in the computed product $\mathrm{fl}(C\,\mathrm{adj}(A)B)$, where A, B, and C are upper-triangular 2×2 matrices.

Lemma 14. *Given a 2×2 upper-triangular matrix R and an orthonormal matrix Q, let $Z = |Q^T|\,|R|$; then $z_{11} + z_{12} \le \sqrt{3}\|R\|$. Likewise, if $Z = |R||Q|$, then $z_{12} + z_{22} \le \sqrt{3}\|R\|$.*

Proof. For the first result, we have for some α and β satisfying $\alpha^2 + \beta^2 = 1$ that

$$Z = \begin{bmatrix} |\alpha| & |\beta| \\ |\beta| & |\alpha| \end{bmatrix} \begin{bmatrix} |r_{11}| & |r_{12}| \\ & |r_{22}| \end{bmatrix} = \begin{bmatrix} |\alpha||r_{11}| & |\alpha||r_{12}| + |\beta||r_{12}| \\ |\beta||r_{11}| & |\beta||r_{12}| + |\alpha||r_{12}| \end{bmatrix}.$$

It follows that

$$\begin{aligned} z_{11} + z_{12} &= |\alpha|(|r_{11}| + |r_{12}|) + \beta|r_{12}| \\ &\le |\alpha|\|R\|_\infty + |\beta|\|R\|_2 \le (|\alpha|\sqrt{2} + \sqrt{1-\alpha^2})\|R\|_2 \le \sqrt{3}\|R\|_2, \end{aligned}$$

where we used the fact that the bound reaches its maximum for $\alpha = \pm\sqrt{2/3}$. The proof of the second result is similar. □

Lemma 15. *Suppose $\overline{M} = \mathrm{fl}(C\,\mathrm{adj}(A)B)$ is computed as*

$$\overline{M} = \begin{bmatrix} \mathrm{fl}(\mathrm{fl}(c_{11}a_{22})b_{11}) & m_{12} \\ & \mathrm{fl}(c_{22}\,\mathrm{fl}(a_{11}b_{22})) \end{bmatrix},$$

where $m_{12} = \mathrm{fl}(\mathrm{fl}(\mathrm{fl}(\mathrm{fl}(c_{11}a_{22})b_{12}) + \mathrm{fl}(c_{12}\,\mathrm{fl}(a_{11}b_{22}))) - \mathrm{fl}(\mathrm{fl}(c_{11}a_{12})b_{22}))$. Then there exist small relative perturbations δA_0, δB_0, and δC_0 of A, B, and C, respectively, such that

$$\overline{M} = (C + \delta C_0)\,\mathrm{adj}(A + \delta A_0)(B + \delta B_0). \tag{13}$$

Specifically, δA_0, δB_0, and δC_0 satisfy $\|\delta A_0\| \le 3.5\boldsymbol{\varepsilon}\|A\|$, $\|\delta B_0\| \le 3\boldsymbol{\varepsilon}\|B\|$, and $\|\delta C_0\| \le 3\boldsymbol{\varepsilon}\|C\|$.

Proof. Ignoring second order terms, we have that

$$\overline{M} = \begin{bmatrix} (c_{11}a_{22})b_{11}(1+\epsilon_1+\epsilon_2) & \overline{m}_{12} \\ & c_{22}(a_{11}b_{22})(1+\epsilon_4+\epsilon_5) \end{bmatrix},$$

where

$$\begin{aligned} \overline{m}_{12} = (c_{11}a_{22})b_{12}&(1+\epsilon_1+\epsilon_3+\epsilon_9+\epsilon_{10}) \\ &+ c_{12}(a_{11}b_{22})(1+\epsilon_5+\epsilon_6+\epsilon_9+\epsilon_{10}) - c_{11}a_{12}b_{22}(1+\epsilon_7+\epsilon_8+\epsilon_{10}), \end{aligned}$$

and $|\epsilon_i| \leq \boldsymbol{\varepsilon}$ for $i = 1, \ldots, 10$. We can get the same $\overline{M}$ in exact arithmetic with the following relative perturbations:

$$\begin{aligned}
\delta a_{11}/a_{11} &= \epsilon_5, & \delta a_{12}/a_{12} &= (\epsilon_7 + \epsilon_8 + \epsilon_{10}), & \delta a_{22}/a_{22} &= \epsilon_1, \\
\delta b_{11}/b_{11} &= \epsilon_2, & \delta b_{12}/b_{12} &= (\epsilon_3 + \epsilon_9 + \epsilon_{10}), & \delta b_{22}/b_{22} &= 0, \\
\delta c_{11}/c_{11} &= 0, & \delta c_{12}/c_{12} &= (\epsilon_6 + \epsilon_9 + \epsilon_{10}), & \delta c_{22}/c_{22} &= \epsilon_4,
\end{aligned}$$

and $\delta a_{21} = \delta b_{21} = \delta c_{21} = 0$, proving (13). Using the equivalence of norms and the definition of the Frobenius norm, we get the bound:

$$\|\delta A\|_2 \leq \|\delta A\|_F \leq \boldsymbol{\varepsilon}\sqrt{11}\max\{|a_{11}|, |a_{12}|, |a_{22}|\} \leq \boldsymbol{\varepsilon}\sqrt{11}\|A\|_2 < 3.5\boldsymbol{\varepsilon}\|A\|_2.$$

The perturbations δB and δC are of rank one and satisfy

$$\|\delta B\| \leq 3\boldsymbol{\varepsilon}(b_{11}^2 + b_{12}^2)^{1/2} \leq 3\boldsymbol{\varepsilon}\|B\|_2 \quad \text{and} \quad \|\delta C\| \leq 3\boldsymbol{\varepsilon}(c_{12}^2 + c_{22}^2)^{1/2} \leq 3\boldsymbol{\varepsilon}\|C\|_2,$$

which concludes the proof. □

Remark 16. We can compute the product $\mathrm{fl}(C\,\mathrm{adj}(A)B)$ and the perturbations δA, δB, and δC in different ways. Furthermore, the bounds in the above lemma are not the tightest possible. Instead, the above perturbations and their bounds are such that we can invoke the lemma for the transposed and permuted triplet $(\Pi_r A^T \Pi_c, \Pi_r C^T \Pi_c, \Pi_r B^T \Pi_c)$ from the end of Section 3, rather than for the original triplet (A, B, C), without getting qualitative differences in the perturbations of A, B, and C.

We are now ready for the main result of this section: the numerical stability of Algorithm 2 in floating-point arithmetic.

Theorem 17. *Suppose that $\overline{A}'$, $\overline{B}'$, $\overline{C}'$, $\overline{H}$, $\overline{K}$, $\overline{P}$, $\overline{Q}$, $\overline{U}$, and $\overline{V}$ are computed by Algorithm 2 in floating-point arithmetic, with $\overline{M}$ computed as in Lemma 15. Furthermore, define $\overline{H}'$ and $\overline{K}'$ as $\mathrm{fl}(\overline{Q}^T\overline{H})$ and $\mathrm{fl}(\overline{KP})$, respectively, with their $(1,2)$ elements zeroed. Then the following assertions are true.*

1. *The matrices $\overline{A}'$, $\overline{B}'$, and $\overline{C}'$ are lower triangular.*
2. *The product $\overline{V}^T\overline{MU}$ is within $132\boldsymbol{\varepsilon}\|\overline{M}\|$ of being diagonal.*
3. *The rows of $\overline{C}'$ and $\mathrm{adj}(\overline{H}')$ are within $86\boldsymbol{\varepsilon}\|C\|$ and $93.5\boldsymbol{\varepsilon}\|A\|\|B\|$, respectively, of being parallel. Likewise, the columns of $\mathrm{adj}(\overline{K}')$ and $\overline{B}'$ are within $93.5\boldsymbol{\varepsilon}\|A\|\|C\|$ and $86\boldsymbol{\varepsilon}\|B\|$, respectively, of being parallel.*
4. *The matrices $\overline{B}'$, $\overline{C}'$, $\overline{H}'$, and $\overline{K}'$ are computed stably in the following sense. There exist δB, δC, δH, and δK, and orthonormal matrices $\mathcal{P}$, $\mathcal{Q}$, U, and V, such that $U^T\overline{M}V$ is an exact (unnormalized) SVD of $\overline{M}$, and*

$$\begin{aligned}
\overline{B}' &= \mathcal{P}^T(B + \delta B)U, \quad \overline{H}' = \mathcal{Q}^T(\mathrm{adj}(A)B + \delta H)U, \\
\overline{C}' &= V^T(C + \delta C)\mathcal{Q}, \quad \overline{K}' = V^T(C\,\mathrm{adj}(A) + \delta K)\mathcal{P},
\end{aligned}$$

where $\|\delta B\| \leq 493\boldsymbol{\varepsilon}\|B\|$, $\|\delta C\| \leq 493\boldsymbol{\varepsilon}\|C\|$, $\|\delta H\| \leq 486\boldsymbol{\varepsilon}\|C\|\|A\|$, and $\|\delta K\| \leq 486\boldsymbol{\varepsilon}\|A\|\|B\|$. ∎

Proof. The proofs of first three assertions of the theorem follow the proof of Bai and Demmel for the QSVD [4, Thm. 3.1], *mutatis mutandis*. The proof of the fourth assertion deviates in the choice of the ηs defined below, which is a difference that will be useful for later propositions and bounds. Due to this similarity, we also use the following facts from Bai and Demmel's proof.

Fact 1 The computed $\overline{U}$ and $\overline{V}$ from xLASV2 satisfy $\overline{U} = U + \delta U$, $\overline{V} = V + \delta V$, where $V^T\overline{M}U$ is an exact (unnormalized) SVD of $\overline{M}$, and δU and δV are small componentwise relative perturbations of U and V, respectively, bounded by 46.5ε in each component. This also implies $\|\delta U\| \le \sqrt{2} \cdot 46.5\varepsilon < 66\varepsilon$ and $\|\delta V\| < 66\varepsilon$.

Fact 2 Using simple geometry, one can show that changing f to $f + \delta f$ and g to $g + \delta g$ changes $c = f/\sqrt{f^2 + g^2}$ and $s = g/\sqrt{f^2 + g^2}$ to $c + \delta c$ and $s + \delta s$, respectively, where $\sqrt{\delta c^2 + \delta s^2} \le 2((\delta f^2 + \delta g^2)/(f^2 + g^2))^{1/2}$.

Fact 3 Subroutine xLARTG computes $c = f/\sqrt{f^2 + g^2}$ and $s = g/\sqrt{f^2 + g^2}$ with relative errors bounded by 6ε. This means that the 2×2 matrix $\mathrm{rot}(c, s)$ has an error bounded in norm by $\sqrt{2} \cdot 6\varepsilon < 9\varepsilon$.

Fact 4 If X and Y are 2×2 matrices, then $\| \mathrm{fl}(XY) - XY\| \le 4\varepsilon\|X\|\|Y\|$.

To prove the assertions of the theorem, first suppose that $c_{11} = 0$ and $b_{22} = 0$. Then the first assertion holds by construction, and the second assertion follows from the nonzero structure of the matrices. The third and fourth assertions hold since $\overline{P} = \overline{Q} = J$ are exact, and $\overline{U}$ and $\overline{V}$ are computed from B and C with high relative accuracy by Fact 3. Now assume for the rest of the proof that $c_{11} \neq 0$ or $b_{22} \neq 0$.

The lower-triangularity of $\overline{A}'$, $\overline{B}'$, and $\overline{C}'$ hold by construction. The near diagonality of $\overline{V}^T\overline{MU}$ holds by the high accuracy of $\overline{U}$ and $\overline{V}$. Specifically, it follows from Fact 1 that

$$\overline{V}^T\overline{MU} = (V + \delta V)^T\overline{M}(U + \delta U) \approx V^T\overline{M}U + \delta V^T\overline{M}U + V^T\overline{M}\delta U,$$

where $V^T\overline{M}U$ is an exact unnormalized SVD of $\overline{M}$, and

$$\|\delta V^T\overline{M}U\| + \|V^T\overline{M}\delta U\| \le (66 + 66)\varepsilon\|\overline{M}\| = 132\varepsilon\|\overline{M}\|.$$

We prove the third assertion only for $\overline{C}'$ and $\mathrm{adj}(\overline{H}')$, as the and the proof for $\mathrm{adj}(\overline{K}')$ and $\overline{B}'$ is similar. We also only have to consider the bottom rows, since the explicitly zeroed $(1, 2)$ entries make the top rows parallel by construction. Now, the bottom rows of $\overline{C}'$ and $\mathrm{adj}(\overline{H}')$ are identical to the bottom rows of $\mathrm{fl}(\overline{GQ})$ and $\mathrm{adj}(\mathrm{fl}(\overline{Q}^T\overline{H}))$, respectively, and the bottom rows of $V^T(C + \delta C_0)\mathcal{Q}$ and $\mathrm{adj}(U^T(B + \delta B_0)\,\mathrm{adj}(A + \delta A_0)\mathcal{Q})$ are parallel by construction for any orthonormal matrix $\mathcal{Q}$. Hence, it suffices to bound the distance between the former two pairs of matrices for a suitable choice of $\mathcal{Q}$, which we can do as follows. From Lemma 15 and Fact 4 it follows that for some error term F_1 with $\|F_1\| \le 4\varepsilon\|C\|$, we have that

$$\overline{G} = \mathrm{fl}(\overline{V}^T C) = \overline{V}^T C + F_1 = V^T(C + \delta C_0) \underbrace{- V^T\delta C_0 + \delta V^T C + F_1}_{F_2};$$

thus, the error in $\overline{G}$ is bounded by

$$\|F_2\| \le \|V^T\delta C_0\| + \|\delta V^T C\| + \|F_1\| \le (3 + 66 + 4)\varepsilon\|C\| = 73\varepsilon\|C\|.$$

Using Fact 3, we see that for any $\overline{Q} = \mathcal{Q} + \delta\mathcal{Q}$ computed with xLARTG, we have that

$$\begin{aligned}\mathrm{fl}(\overline{GQ}) = \overline{GQ} + F_2 &= (V^T(C + \delta C_0) + F_2)(\mathcal{Q} + \delta\mathcal{Q}) + F_3 \\ &= V^T(C + \delta C_0)\mathcal{Q} + \underbrace{F_2\mathcal{Q} + V^T C\delta\mathcal{Q} + F_3}_{F_4},\end{aligned}$$

with the error term bounded by

$$\|F_4\| \le \|F_2\mathcal{Q}\| + \|V^T C\delta\mathcal{Q}\| + \|F_3\| \le (73 + 9 + 4)\varepsilon\|C\| = 86\varepsilon\|C\|.$$

For some F_5 with $\|F_5\| \le 2 \cdot 4\boldsymbol{\varepsilon}\|A\|\|B\|$, we have that

$$\begin{aligned}\overline{H} &= \operatorname{adj}(A)B\overline{U} + F_5 \\ &= \operatorname{adj}(A + \delta A_0 - \delta A_0)(B + \delta B_0 - \delta B_0)U + \operatorname{adj}(A)B\delta U + F_5 \\ &= \operatorname{adj}(A + \delta A_0)(B + \delta B_0)U \underbrace{- \operatorname{adj}(\delta A_0)BU - \operatorname{adj}(A)\delta B_0 U + \operatorname{adj}(A)B\delta U + F_5}_{F_6},\end{aligned}$$

so that the error in $\overline{H}$ is bounded by

$$\begin{aligned}\|F_6\| &\le \|\operatorname{adj}(\delta A_0)BU\| + \|\operatorname{adj}(A)\delta B_0 U\| + \|\operatorname{adj}(A)B\delta U\| + \|F_5\| \\ &\le (3.5 + 3 + 66 + 2 \cdot 4)\boldsymbol{\varepsilon}\|A\|\|B\| = 80.5\boldsymbol{\varepsilon}\|A\|\|B\|.\end{aligned}$$

Hence, for some F_6 and F_7 with $\|F_6\| \le 80.5\boldsymbol{\varepsilon}\|A\|\|B\|$ and $\|F_7\| \le 4\boldsymbol{\varepsilon}\|A\|\|B\|$, and any rotation $\overline{Q} = \mathcal{Q} + \delta\mathcal{Q}$ computed with xLARTG, we have that

$$\begin{aligned}\operatorname{fl}(\overline{Q}^T\overline{H}) &= \overline{Q}^T\overline{H} + F_7 \\ &= (\mathcal{Q} + \delta\mathcal{Q})^T(\operatorname{adj}(A + \delta A_0)(B + \delta B_0)V + F_6) + F_7 \\ &= \mathcal{Q}^T \operatorname{adj}(A + \delta A_0)(B + \delta B_0)V + \underbrace{\mathcal{Q}^T F_6 + \delta\mathcal{Q}^T \operatorname{adj}(A)BV + F_7}_{F_8},\end{aligned} \tag{14}$$

so that the error term is bounded by $(80.5 + 9 + 4)\boldsymbol{\varepsilon}\|A\|\|B\| = 93.5\boldsymbol{\varepsilon}\|A\|\|B\|$.

For the fourth and final assertion, we only prove the bounds for $\|\delta C\|$ and $\|\delta H\|$, because bounding $\|\delta B\|$ and $\|\delta K\|$ is similar. The main challenge now is to quantify the effect of zeroing the $(1,2)$ entries at the end of the algorithm. Suppose first that $|\overline{h}_{12}| + |\overline{h}_{22}| = 0$, then the algorithm computes Q from $\overline{G}$, and $\overline{Q}$ zeros the $(1,2)$ entry of $\overline{G}$ with high relative accuracy as a result of Fact 3. Furthermore, in this case it holds for any $\overline{Q}$ that $\operatorname{fl}(\overline{Q}^T\overline{H})_{12} = 0$. Otherwise, if $|\overline{h}_{12}| + |\overline{h}_{22}| \neq 0$ but $|\overline{g}_{11}| + |\overline{g}_{12}| = 0$, then the algorithm computes $\overline{Q}$ to accurately zero out the $(1,2)$ entry of $\overline{H}$, and $\operatorname{fl}(\overline{GQ})_{12} = 0$. Now we may assume that $|\overline{h}_{12}| + |\overline{h}_{22}| \neq 0$ and $|\overline{g}_{11}| + |\overline{g}_{12}| \neq 0$ for the rest of the proof, and that $\overline{\eta}_g \le \overline{\eta}_h$ so that the algorithm computes Q from $\overline{G}$. The proof is similar when $\overline{\eta}_g > \overline{\eta}_h$ and the algorithm computes Q from $\overline{H}$, but leads to different bounds that we summarize at the end of the proof.

It follows from Fact 3 that the algorithm computes $\overline{Q}$ in such a way that the $(1,2)$ entry of $\overline{G}$ is zeroed with high relative precision. Bounding the effect of zeroing the $(1,2)$ entry of $\operatorname{fl}(\overline{Q}^T\overline{H})$ to get $\overline{H}'$ is more involved. Let $\overline{Q} = Q + \delta Q$, where Q denotes the exact rotation obtained from $V^T(C + \delta C_0)$ in exact arithmetic (which can be bigger than just the error from xLARTG due to the errors in $\overline{V}$); then

$$\begin{aligned}|\operatorname{fl}(\overline{Q}^T\overline{H})_{12}| &= |\overline{h}_{12}(1 + 2\epsilon)(q_{11} + \delta q_{11}) + \overline{h}_{22}(1 + 2\epsilon)(q_{21} + \delta q_{21})| \\ &= |(h_{12} + 82.5\epsilon\|A\|\|B\|)q_{11} + (1 + 2\epsilon)\overline{h}_{12}\delta q_{11} \\ &\quad + (h_{22} + 82.5\epsilon\|A\|\|B\|)q_{21} + (1 + 2\epsilon)\overline{h}_{22}\delta q_{21}| \\ &\le (1 + 2\boldsymbol{\varepsilon})(|\overline{h}_{12}||\delta q_{11}| + |\overline{h}_{22}||\delta q_{21}|) + \sqrt{2} \cdot 82.5\boldsymbol{\varepsilon}\|A\|\|B\|.\end{aligned}$$

Before proceeding, recall that $\overline{\widehat{G}} = \operatorname{fl}(|\overline{V}|^T|C|)$ and $\overline{\widehat{H}} = \operatorname{fl}(|\operatorname{adj}(A)|\operatorname{fl}(|B||\overline{U}|))$, so that

$$\boldsymbol{\varepsilon}\overline{\eta}_g = \boldsymbol{\varepsilon}(\overline{\widehat{g}}_{11} + \overline{\widehat{g}}_{12})/(|\overline{g}_{11}| + |\overline{g}_{12}|) \quad \text{and} \quad \boldsymbol{\varepsilon}\overline{\eta}_h = \boldsymbol{\varepsilon}(\overline{\widehat{h}}_{12} + \overline{\widehat{h}}_{22})/(|\overline{h}_{12}| + |\overline{h}_{22}|).$$

Furthermore, it can be verified that the entries of $\overline{G}$ may have a perturbation of up to $51.5\boldsymbol{\varepsilon}\overline{\widehat{g}}_{ij}$, where $46.5\boldsymbol{\varepsilon}$ comes from the perturbations in δU, and $2\boldsymbol{\varepsilon}$ from the roundoff errors in the matrix-matrix multiplication, and $3\boldsymbol{\varepsilon}$ from δC_0. Hence, using

$$(51.5\boldsymbol{\varepsilon}\overline{\widehat{g}}_{11})^2 + (51.5\boldsymbol{\varepsilon}\overline{\widehat{g}}_{12})^2 \le (51.5\boldsymbol{\varepsilon})^2(|\overline{\widehat{g}}_{11}| + |\overline{\widehat{g}}_{12}|)^2 = (51.5\boldsymbol{\varepsilon}\overline{\eta}_g)^2(|\overline{g}_{11}| + |\overline{g}_{12}|)^2$$

and Facts 2 and 3, we can bound $(|\delta q_{11}|^2 + |\delta q_{21}|^2)^{1/2}$ by

$$9\varepsilon + 2 \cdot 51.5\varepsilon\overline{\eta}_g \frac{|\overline{g}_{11}| + |\overline{g}_{12}|}{\sqrt{\overline{g}_{11}^2 + \overline{g}_{12}^2}} \leq 9\varepsilon + 2\sqrt{2} \cdot 51.5\varepsilon\overline{\eta}_g \leq 155\varepsilon \max\{1, \overline{\eta}_g\}.$$

If $\overline{\eta}_g \leq 1$, then $|\mathrm{fl}(\overline{Q}^T\overline{H})_{12}| \leq \sqrt{2}(155 + 82.5)\varepsilon\|A\|\|B\| \leq 336\varepsilon\|A\|\|B\|$; otherwise, we can use Lemma 14 to show that

$$(|\overline{h}_{11}| + |\overline{h}_{12}|)\varepsilon = (\overline{\widehat{h}}_{12} + \overline{\widehat{h}}_{22})\varepsilon\overline{\eta}_h^{-1} \leq \sqrt{3}\|A\|\|B\|\varepsilon\overline{\eta}_h^{-1},$$

which in turn implies the bound

$$\begin{aligned}
|\,\mathrm{fl}((\overline{Q}^T\overline{H})_{12})| &\leq (1 + 2\varepsilon)(|\overline{h}_{11}| + |\overline{h}_{12}|)155\varepsilon\overline{\eta}_g + 117\varepsilon\|A\|\|B\| \\
&\leq (\sqrt{3} \cdot 155\frac{\overline{\eta}_g}{\overline{\eta}_h} + 117)\varepsilon\|A\|\|B\|.
\end{aligned}$$

Since $\overline{\eta}_g \leq \overline{\eta}_h$ by assumption, it follows that $|\mathrm{fl}(\overline{Q}^T\overline{H})_{12}| \leq 386\varepsilon\|A\|\|B\|$. By writing the error term in (14) as F_8 and the explicit zeroing of the $(1,2)$ entry as $F_9 = -\mathrm{fl}(\overline{Q}^T\overline{H})_{12}e_1e_2^T$, everything can be put together to yield

$$\begin{aligned}
\overline{H}' &= \mathcal{Q}^T \,\mathrm{adj}(A + \delta A_0)(B + \delta B_0)U + F_8 + F_9 \\
&= \mathcal{Q}^T(\mathrm{adj}(A)B + \mathrm{adj}(\delta A_0)B + \mathrm{adj}(A)\delta B_0 + QF_8U^T + QF_9U^T)U \\
&= \mathcal{Q}^T(\mathrm{adj}(A)B + \delta H)U,
\end{aligned}$$

where $\|\delta H\| \leq (3.5 + 3 + 93.5 + 386)\varepsilon\|A\|\|B\| = 486\varepsilon\|A\|\|B\|$.

The proof is similar when $\overline{\eta}_g > \overline{\eta}_h$ and $\overline{Q}$ is computed from $\overline{H}$, except for the following differences. We get $\sqrt{2} \cdot (73 + 2) \leq 107$ instead of $\sqrt{2} \cdot (80.5 + 2) \leq 117$, elements in $\overline{H}$ may be perturbed by up to $(46.5 + 3.5 + 3 + 2 \cdot 2)\varepsilon\overline{\widehat{h}}_{ij} = 57\varepsilon\overline{\widehat{h}}_{ij}$, the quantity $(|\delta q_{11}|^2 + |\delta q_{21}|^2)^{1/2}$ is bounded by $9\varepsilon + 2\sqrt{2} \cdot 57\varepsilon\overline{\eta}_h \leq 171\varepsilon \max\{1, \overline{\eta}_h\}$, the perturbation δA_0 is not part of δC, and 93.5ε should be replaced by $\|F_4\| \leq 86\varepsilon$ in the final bound. Hence, the factor in the resulting bound is $(3 + 86 + 107 + \sqrt{3} \cdot 171)\varepsilon \leq 493\varepsilon$. □

Although the theorem above shows that Algorithm 2 has favorable numerical properties, it lacks a bound on the backward error of $\overline{A}'$. Moreover, we have to content ourselves with $\overline{H}'$ and $\overline{K}'$ instead of $\mathrm{adj}(\overline{A}')\overline{B}'$ and $\overline{C}'\,\mathrm{adj}(\overline{A}')$. However, the proof shows that we can bound the errors in P and Q in terms of the ηs, which in turn allows us to express the error in $\overline{A}'$ in terms of the ηs. We can then try to ensure that the ηs remain small, so that the error in $\overline{A}'$ is small. These things are the focus of the next section.

5 The backward error of the computed A′. Although the numerical results in Section 9 suggest that the relative magnitude of $\mathrm{fl}(\overline{P}^T A\overline{Q})_{12}$ is always small in practice, it is unclear if we can prove that $\|\delta A\|$ is $\mathcal{O}(\varepsilon\|A\|)$ in the worst case. An alternative is to bound the backward error of $\overline{A}'$ in terms of the ηs, and then to analyze the behavior of the ηs. We can simplify this analysis with the following two definitions.

Definition 18. Define η_g from Algorithm 2 as

$$\eta_g = \begin{cases} (\widehat{g}_{11} + \widehat{g}_{12})/(|g_{11}| + |g_{12}|) & \text{if } |g_{11}| + |g_{12}| \neq 0 \\ \infty & \text{if } |g_{11}| + |g_{12}| = 0, \end{cases}$$

and define the remaining ηs and $\overline{\eta}$s analogously.

Definition 19. Define $\eta_{\max}$ as $\eta_{\max} = \max\{1, \min\{\eta_g, \eta_h\}, \min\{\eta_k, \eta_l\}\}$ and define $\overline{\eta}_{\max}$ analogously.

We will later see that $\eta_{\max}, \overline{\eta}_{\max} < \infty$, which is important for two reasons. First, it allows us to simplify the conditions in Algorithm 2 that determine whether to compute Q and P from G or H and L or K, respectively, by dropping the zero checks and keeping just $\eta_g \leq \eta_h$ and $\eta_l \leq \eta_k$. Second, we can now bound the backward error of $\overline{A}'$ in terms of $\overline{\eta}_{\max}$ instead of having to consider separate cases with separate $\overline{\eta}$s.

Theorem 20. *Suppose $\overline{P}$ is obtained in a similar way as $\overline{Q}$, and $171\varepsilon\overline{\eta}_{\max} \ll 1$; then there exists δA, $\mathcal{P}$, and $\mathcal{Q}$ such that $\overline{A}' = \mathcal{P}(A + \delta A)\mathcal{Q}$ and $\|\delta A\| \leq (44.5 + 342\overline{\eta}_{\max})\varepsilon\|A\|$.*

Proof. Since $\overline{P}$ is computed in a similar way as $\overline{Q}$, it follows from the proof of Theorem 17 that $\overline{P}$ can be decomposed as both $\overline{P} = \mathcal{P} + \delta\mathcal{P}$ and $\overline{P} = P + \delta P$. Here, $\mathcal{P}$ and P are both exactly orthonormal matrices, and $\|\delta\mathcal{P}\| \leq 9\varepsilon$ is the error incurred by computing any $\overline{P}$ with xLARTG in floating-point arithmetic, and $\|\delta P\| \leq 171\varepsilon\eta_{\max}$ is the error incurred by computing the rotation from an approximation of $V^T(C+C_0)\,\mathrm{adj}(A+A_0)$ or $(B+B_0)U$ in floating-point arithmetic. It follows that for some F_1 with $\|F_1\| \leq 2 \cdot 4\varepsilon$,

$$\begin{aligned}\mathrm{fl}(\overline{P}^T A\overline{Q}) &= (\mathcal{P} + \delta\mathcal{P})^T A(\mathcal{Q} + \delta\mathcal{Q}) + F_1 \\ &= \mathcal{P}^T(A + \delta A_0)\mathcal{Q} \underbrace{- \mathcal{P}^T\delta A_0\mathcal{Q} + \delta\mathcal{P}^T A\mathcal{Q} + \mathcal{P}^T A\delta\mathcal{Q} + \delta\mathcal{P}^T A\delta\mathcal{Q} + F_1}_{F_2},\end{aligned}$$

where the error is bounded by

$$\|F_2\| \leq \|\mathcal{P}^T\delta A_0\mathcal{Q}\| + \|\delta\mathcal{P}^T A\mathcal{Q}\| + \|\mathcal{P}^T A\delta\mathcal{Q}\| + \|F\| \leq (3.5 + 9 + 9 + 2 \cdot 4)\varepsilon\|A\| = 29.5\varepsilon\|A\|.$$

Furthermore, for some f_3 with

$$|f_3| \leq 8\varepsilon\|\boldsymbol{e}_1^T(P + \delta P)\|\|A\|\|(Q + \delta Q)\boldsymbol{e}_2\| \leq 8\varepsilon(1 + 171\varepsilon\overline{\eta}_{\max})^2\|A\| \approx 8\varepsilon\|A\|$$

we have that

$$\begin{aligned}|\,\mathrm{fl}(\overline{P}^T A\overline{Q})_{12}| &\leq |(P + \delta P)^T A(Q + \delta Q)|_{12} + |f_3| \\ &= |P^T(A + \delta A_0)Q - P^T\delta A_0 Q + \delta P^T AQ + P^T A\delta Q + \delta P^T A\delta Q|_{12} + |f_3| \\ &\leq |P^T\delta A_0 Q|_{12} + |\delta P^T AQ|_{12} + |P^T A\delta Q|_{12} + |\delta P^T A\delta Q|_{12} + |f_3| \\ &\leq (3.5\varepsilon + 2 \cdot 171\varepsilon\overline{\eta}_{\max} + 8\varepsilon)\|A\|.\end{aligned}$$

Here, we used that $P^T(A + \delta A_0)Q = 0$, and that the assumption $171\varepsilon\overline{\eta}_{\max} \ll 1$ makes $\varepsilon^2\overline{\eta}_{\max}$ and $(\varepsilon\overline{\eta}_{\max})^2$ higher-order terms. Now the explicit zeroing of the $(1,2)$ entry of A' is the same as adding the error term $F_4 = -\,\mathrm{fl}(\overline{P}^T A\overline{Q})_{12}\boldsymbol{e}_1\boldsymbol{e}_2^T$, so that

$$A' = \mathcal{P}^T(A + \delta A_0)\mathcal{Q} + F_2 + F_4 = \mathcal{P}^T(A + \delta A_0 + \mathcal{P}(F_2 + F_4)\mathcal{Q}^T)\mathcal{Q} = \mathcal{P}^T(A + \delta A)\mathcal{Q}.$$

Thus, by combining the relevant error terms and their bounds, we get

$$\|\delta A\| \leq (3.5 + 29.5 + 11.5 + 342\overline{\eta}_{\max})\varepsilon\|A\| = (44.5 + 342\overline{\eta}_{\max})\varepsilon\|A\|,$$

which is the desired result. □

In essence, if $\overline{\eta}_{\max}$ is sufficiently small, then the errors δP and δQ stay small, and Algorithm 2 computes $\overline{A}'$ stably. Hence, the goal is now to bound $\overline{\eta}_{\max}$. We start by showing that $\overline{\eta}_{\max}$ is always finite, but before we can start with the proof, we need the following properties of Algorithm 2 and the routine xLASV2.

Lemma 21. *Consider Algorithm 2 and assume the following:* $c_{11} \neq 0$ *or* $b_{22} \neq 0$*, the SVD of* $\overline{M}$ *is computed with* xLASV2 *as given in Bai and Demmel [4, App.], and the columns of* $\overline{U}$ *and* $\overline{V}$ *are postmultiplied by* J *if the relevant conditions in the algorithm are met. Then* $\overline{U} = \overline{V} = I$ *if* $\overline{M} = 0$*,* $|\overline{U}| = |J|$ *and* $|\overline{V}| = I$ *if* $\overline{m}_{11} = \overline{m}_{22} = 0$ *and* $\overline{m}_{12} \neq 0$*,* $|\overline{V}| \approx I$ *if* $b_{22} = 0$*, and* $|\overline{U}| \approx |J|$ *if* $c_{11} = 0$ *but* $C \neq 0$*, where the zeros are exact even for the latter* $\overline{U}$ *and* $\overline{V}$*.*

Proof. The desired results follow from the implementation and high relative accuracy of xLASV2, combined with the postmultiplication of $\overline{U}$ and $\overline{V}$ by J when the conditions on Line 9 of Algorithm 2 are met. □

The preceding lemma implies that the $\overline{U}$ and $\overline{V}$ computed with Algorithm 2 in floating-point arithmetic correspond to the exact U and V from Lemma 11 with high relative accuracy, at least for those cases of Lemma 11 that correspond to the assumption of the lemma above. With this result, we can now prove the following proposition and corollary, which show that $\overline{\eta}_{\max}$ is finite.

Proposition 22. *If* $c_{11} \neq 0$ *or* $b_{22} \neq 0$*, then* $|\overline{g}_{11}| + |\overline{g}_{12}| = 0$ *and* $|\overline{h}_{12}| + |\overline{h}_{22}| = 0$ *cannot hold simultaneously. Likewise and under the same assumptions, neither* $|\overline{k}_{11}| + |\overline{k}_{12}| = 0$ *and* $|\overline{l}_{12}| + |\overline{l}_{22}| = 0$*, nor* $|\overline{g}_{11}| + |\overline{g}_{12}| = 0$ *and* $|\overline{l}_{12}| + |\overline{l}_{22}| = 0$ *can hold simultaneously.*

Proof. If $b_{22} = 0$, then by Lemma 21 $|\overline{V}| \approx I$, so that $g_{11} = \overline{v}_{11} c_{11}(1+\epsilon) \neq 0$. Conversely, if $C = 0$, then $\overline{U} = I$ and $\overline{h}_{22} = a_{11} b_{22}(1+\epsilon) \neq 0$. If $c_{11} = 0$ but $C \neq 0$, then

$$\overline{M} = \begin{bmatrix} 0 & c_{12}(a_{11}b_{22})(1+\epsilon_1)(1+\epsilon_2) \\ 0 & c_{22}(a_{11}b_{22})(1+\epsilon_1)(1+\epsilon_3) \end{bmatrix}$$

for some $|\epsilon_1|, |\epsilon_2|, |\epsilon_3| \leq \boldsymbol{\varepsilon}$. Using the entries of $\overline{M}$ and Facts 1 and 2 from the proof of Theorem 17, we see that $|\overline{g}_{12}|$ equals

$$|v_{11}c_{12}(1+48.5\epsilon) + v_{21}c_{22}(1+48.5\epsilon)| = \left| \frac{c_{12}^2(1+50.5\epsilon)}{(c_{12}^2+c_{22}^2)^{-1/2}} + \frac{c_{22}^2(1+50.5\epsilon)}{(c_{12}^2+c_{22}^2)^{-1/2}} \right|,$$

which is within $50.5\|C\|\boldsymbol{\varepsilon}$ of $\|C\|$. Hence, we conclude that $|\overline{g}_{12}|$ is nonzero.

If both $c_{11} \neq 0$ and $b_{22} \neq 0$ but $|\overline{g}_{11}| + |\overline{g}_{12}| = 0$ and $|\overline{h}_{12}| + |\overline{h}_{22}| = 0$, then

$$\begin{aligned} 0 &= |v_{11}c_{11}(1+47.5\epsilon)| + |v_{11}c_{12}(1+48.5\epsilon) + v_{21}c_{22}(1+48.5\epsilon)|, \\ 0 &= |u_{22}b_{22}a_{11}(1+48.5\epsilon)| \\ &\quad + |u_{22}(a_{22}b_{12}(1+50.5\epsilon) - a_{12}b_{22}(1+50.5\epsilon)) + u_{12}a_{22}b_{11}(1+50.5\epsilon)|. \end{aligned}$$

The former implies that $v_{11} = 0$ and thus also that $c_{22} = 0$, and the latter implies that $u_{22} = 0$ and thus also that $b_{11} = 0$. Yet, by Lemma 21 we cannot simultaneously have $|\overline{U}| = |J|$ and $|\overline{V}| = |J|$ when $c_{22} = b_{11} = 0$ so that we have a contradiction.

The proof for the second claim in the proposition is similar, and the third claim holds because $|\overline{k}_{11}| + |\overline{k}_{12}| = 0$ if $|\overline{g}_{11}| + |\overline{g}_{12}| = 0$ and $|\overline{h}_{12}| + |\overline{h}_{22}| = 0$ if $|\overline{l}_{12}| + |\overline{l}_{22}| = 0$. □

Corollary 23. *Since neither* $\overline{\eta}_g$ *and* $\overline{\eta}_h$*, nor* $\overline{\eta}_k$ *and* $\overline{\eta}_l$ *are infinite simultaneously,* $\overline{\eta}_{\max}$ *is finite.*

In exact arithmetic, we can prove an even stronger result, namely that $\eta_g = \infty$ and $\eta_l = \infty$ if and only if $c_{11} = 0$ and $b_{22} = 0$, respectively. This is not the case in floating-point arithmetic, and the $\overline{\eta}$s may be finite or infinite in unexpected situations. For example, if $B = \left[\begin{smallmatrix} b_{11} & b_{12} \\ 0 & 0 \end{smallmatrix}\right]$, then the exact U should be such that $L = BU = \left[\begin{smallmatrix} l_{11} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ so that $\eta_l = \infty$; but the computed $\overline{U}$ is typically such that $\overline{L} = \mathrm{fl}(B\overline{U}) = \left[\begin{smallmatrix} \bar{l}_{11} & \mathcal{O}(\epsilon\|B\|) \\ 0 & 0 \end{smallmatrix}\right]$ so that $\overline{\eta}_l < \infty$. This can be a problem when, for example, $C = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}\right]$ and $\mathrm{adj}(A) = \left[\begin{smallmatrix} \mu & 1-\mu \\ 0 & -1 \end{smallmatrix}\right]$ for some $0 < \mu \ll 1$, so that $|\overline{V}| \approx I$ and $\overline{\eta}_k \approx (1 + |1-\mu|)/|2\mu|$. Hence, if

$\mu \to 0$ and we do not explicitly set $\bar{l}_{12}$ to zero, then $\overline{\eta}_k$ can become larger than $\overline{\eta}_l$ and Algorithm 2 will compute P from BU rather than from $V^TC\,\mathrm{adj}(A)$. The results from Theorem 17 still hold if this happens, but $\overline{\eta}_{\max}$ will be large and we can no longer expect $|\mathrm{fl}(\overline{P}^T A\overline{Q})_{12}|$ to be small. In this example, the condition number of A is of order $\mathcal{O}(\mu^{-1})$ too, and we will later see that $\kappa(A)$ plays an important role in bounding $\eta_{\max}$.

The next proposition implies that $\overline{\eta}_g < \infty$ if $\eta_g < \infty$ and that $\overline{\eta}_l < \infty$ if $\eta_l < \infty$, which means that the discrepancy between the ηs and the $\overline{\eta}$s for ∞ exists only in one direction. Furthermore, the proposition makes it easier to compute the bounds that we want, because we have formulae for the exact ηs while the computed $\overline{\eta}$s are perturbed by unknown roundoff errors.

Proposition 24. *Suppose $\eta_g, \overline{\eta}_g < \infty$ and $\eta_h, \overline{\eta}_h < \infty$ are small enough; then we have the first-order approximations*

$$\overline{\eta}_g = \eta_g \frac{1 + 50.5\epsilon_1}{1 + 49.5\epsilon_2\eta_g} \quad \textit{and} \quad \overline{\eta}_h = \eta_h \frac{1 + 52.5\epsilon_3}{1 + 51.5\epsilon_4\eta_h}, \tag{15}$$

respectively, where $|\epsilon_i| \leq \boldsymbol{\varepsilon}$ for $i = 1, \ldots 4$. A similar statement holds for $\eta_k, \overline{\eta}_k$ and $\eta_l, \overline{\eta}_l$.

Proof. Since $\overline{v}_{11} = v_{11}(1 + 46.5\epsilon)$ and $\overline{v}_{21} = v_{21}(1 + 46.5\epsilon)$, it follows that

$$\begin{aligned}
\mathrm{fl}(\overline{\widehat{g}}_{11} + \overline{\widehat{g}}_{12}) &= (|\,\mathrm{fl}(\overline{v}_{11}c_{11})| + \mathrm{fl}(|\,\mathrm{fl}(\overline{v}_{11}c_{12})| + |\,\mathrm{fl}(\overline{v}_{21}c_{22})|))(1 + \epsilon) \\
&= |v_{11}c_{11}|(1 + 48.5\epsilon) + |v_{11}c_{12}|(1 + 49.5\epsilon) + |v_{21}c_{22}|(1 + 49.5\epsilon) \\
&= (\widehat{g}_{11} + \widehat{g}_{12})(1 + 49.5\epsilon)
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{fl}(|\overline{g}_{11}| + |\overline{g}_{12}|) &= (|\,\mathrm{fl}(\overline{v}_{11}c_{11})| + |\,\mathrm{fl}(\mathrm{fl}(\overline{v}_{11}c_{12}) + \mathrm{fl}(\overline{v}_{21}c_{22}))|)(1 + \epsilon) \\
&= |v_{11}c_{11}(1 + 48.5\epsilon)| + |v_{11}c_{12}(1 + 49.5\epsilon) + v_{21}c_{22}(1 + 49.5\epsilon)| \\
&= |g_{11} + 48.5\epsilon\widehat{g}_{11}| + |g_{12} + 49.5\epsilon\widehat{g}_{12}| \\
&= (|g_{11}| + |g_{12}|)(1 + 49.5\epsilon\eta_g),
\end{aligned}$$

so that

$$\overline{\eta}_g = \mathrm{fl}\left(\frac{(\widehat{g}_{11} + \widehat{g}_{12})(1 + 49.5\epsilon)}{(|g_{11}| + |g_{12}|)(1 + 49.5\epsilon\eta_g)}\right) = \eta_g \frac{1 + 50.5\epsilon}{1 + 49.5\epsilon\eta_g}.$$

The derivation of the relation between $\overline{\eta}_h$ and η_h is analogous. The proof for $\eta_k, \overline{\eta}_k$ and $\eta_l, \overline{\eta}_l$ is similar. □

Corollary 25. *Under the same assumptions as in Proposition 24, solving* (15) *for η_g and η_h yields*

$$\eta_g = \overline{\eta}_g(1 + 50.5\epsilon_1 - 49.5\epsilon_2\overline{\eta}_g)^{-1} \quad \textit{and} \quad \eta_h = \overline{\eta}_h(1 + 52.5\epsilon_3 - 51.5\epsilon_4\overline{\eta}_h)^{-1}.$$

Proposition 24 shows that the computed $\overline{\eta}$s approximate their exact counterparts if $\boldsymbol{\varepsilon}\eta_{\max} \ll 1$. Although we generally do not know the exact ηs in practice, we still expect this result to hold if $\boldsymbol{\varepsilon}\overline{\eta}_{\max} \ll 1$.

Now that we know the relation between the ηs and $\overline{\eta}$s, we can use bounds for the former to inform us of the behavior of the latter. The next two propositions and the corollary show that bounding the ηs from below and in terms of each other is straightforward.

Proposition 26. *It holds that $\eta_g, \eta_h, \eta_k, \eta_l \geq 1$.*

Proof. Using the triangle inequality we see that

$$\eta_g = \frac{|v_{11}||c_{11}| + |v_{11}||c_{12}| + |v_{21}||c_{22}|}{|v_{11}||c_{11}| + |v_{11}c_{12} + v_{21}c_{22}|} \geq \frac{|v_{11}||c_{11}| + |v_{11}||c_{12}| + |v_{21}||c_{22}|}{|v_{11}||c_{11}| + |v_{11}||c_{12}| + |v_{21}||c_{22}|} = 1.$$

The proof for the remaining ηs is similar. □

Lemma 27. *For any 2×2 upper-triangular matrix A, the singular values of A equal the singular values of $|A|$.*

Proof. Compare the eigenvalues of A^TA and $|A|^T|A|$. □

Proposition 28. *If $\eta_l, \eta_h < \infty$ and $\eta_g, \eta_k < \infty$, then*

$$\frac{1}{2}\kappa(A)^{-1}\eta_l \leq \eta_h \leq 2\kappa(A)\eta_l \quad \text{and} \quad \frac{1}{2}\kappa(A)^{-1}\eta_g \leq \eta_k \leq 2\kappa(A)\eta_g,$$

respectively.

Proof. From $\eta_h = \|\widehat{H}\boldsymbol{e}_2\|_1/\|H\boldsymbol{e}_2\|_1$ and $\eta_l = \|\widehat{L}\boldsymbol{e}_2\|_1/\|L\boldsymbol{e}_2\|_1$ we get

$$\eta_h \leq \sqrt{2}\frac{\|\widehat{H}\boldsymbol{e}_2\|_2}{\|H\boldsymbol{e}_2\|_2} \leq \sqrt{2}\frac{\sigma_{\max}(|\operatorname{adj}(A)|)}{\sigma_{\min}(\operatorname{adj}(A))}\frac{\|\widehat{L}\boldsymbol{e}_2\|_2}{\|L\boldsymbol{e}_2\|_2} \leq 2\kappa(A)\eta_l$$

and

$$\eta_h \geq \frac{1}{\sqrt{2}}\frac{\|\widehat{H}\boldsymbol{e}_2\|_2}{\|H\boldsymbol{e}_2\|_2} \geq \frac{1}{\sqrt{2}}\frac{\sigma_{\min}(|\operatorname{adj}(A)|)}{\sigma_{\max}(\operatorname{adj}(A))}\frac{\|\widehat{L}\boldsymbol{e}_2\|_2}{\|L\boldsymbol{e}_2\|_2} \geq \frac{1}{2}\kappa(A)^{-1}\eta_l.$$

The proof for the bounds with η_g and η_k is analogous. □

Corollary 29. *It follows from the above proposition that*

$$\frac{1}{2}\kappa(A)^{-1}\eta_h \leq \eta_l \leq 2\kappa(A)\eta_h \quad \text{and} \quad \frac{1}{2}\kappa(A)^{-1}\eta_k \leq \eta_g \leq 2\kappa(A)\eta_k.$$

Despite the above bounds, we have no upper bound for $\eta_{\max}$ yet. For example, consider $A = I$, $B = \begin{bmatrix} 1 & \mu \\ 0 & -\mu^5 \end{bmatrix}$, and $C = \begin{bmatrix} 1 & \mu^{-2} \\ 0 & -\mu^{-4} \end{bmatrix}$ for some $0 < \mu \ll 1$; then $\eta_g \approx \mu^{-2}$, $\eta_h \approx \mu^{-4}$, $\eta_k \approx \mu^{-2}$, $\eta_l \approx \mu^{-4}$, and $\eta_{\max} \approx \mu^{-2}$. The key to bounding $\eta_{\max}$, is to permute the columns of U and the columns of V if necessary.

Lemma 30. *If $\mu^{-1} \leq \eta_g < \infty$ for some $0 < \mu < 1/2$, then $\|\boldsymbol{e}_2^T\widehat{G}\|_1/\|\boldsymbol{e}_2^TG\|_1 \leq 2/(1-2\mu)$. A similar statement holds for η_l.*

Proof. Define $\|\operatorname{vec}(G)\|_1 = |g_{11}| + |g_{12}| + |g_{21}| + |g_{22}|$, then

$$\|\widehat{G}\|_1 = \||V^T|VG\| \leq \||V^T|\|_1\|V\|_1\|G\|_1 \leq 2\|\operatorname{vec}(G)\|_1.$$

Now the bound $\eta_g = \|\boldsymbol{e}_1^T\widehat{G}\|_1/\|\boldsymbol{e}_1^TG\|_1 \geq \mu^{-1}$ implies that

$$\|\boldsymbol{e}_1^TG\|_1 \leq \mu\|\boldsymbol{e}_1^T\widehat{G}\|_1 \leq 2\mu\|\operatorname{vec}(G)\|_1.$$

Hence,

$$\|\boldsymbol{e}_2^TG\|_1 = \|\operatorname{vec}(G)\|_1 - \|\boldsymbol{e}_1^TG\|_1 \geq (1-2\mu)\|\operatorname{vec}(G)\|_1$$

so that $\|\boldsymbol{e}_2^T\widehat{G}\|_1/\|\boldsymbol{e}_2^TG\|_1 \leq 2\|\operatorname{vec}(G)\|_1/\|\boldsymbol{e}_2^TG\|_1 = 2/(1-2\mu)$. □

The result of the lemma above implies that working with UJ and VJ instead of U and V, respectively, decreases the value of η_g (η_l) when $\eta_g > 4$ ($\eta_l > 4$). However, this postmultiplication with J may interfere with our attempt to minimize the angles of the rotations as described in Section 4, and may thus lead to slower convergence of the implicit Kogbetliantz iteration. Hence, we should not try to minimize $\eta_{\max}$ thoughtlessly. A possible solution is to check whether $\eta_{\max}$ is larger than some tolerance $\tau_\eta \geq 1$, and whether working with UJ and VJ reduces $\eta_{\max}$. Otherwise, we should keep the original U and V. This idea leads to the following algorithm.

Algorithm 3 (2×2 upper-triangular RSVD (RSVD22-τ_η))**.**
Input: 2×2 upper-triangular matrices A, B, and C, with A nonsingular, and tolerance $\tau_\eta \geq 1$.
Output: Orthonormal matrices P, Q, U, and V, such that P^TAQ, P^TBU, and V^TCQ are lower triangular, and $(V^TCQ)\,\mathrm{adj}(P^TAQ)(P^TBU) = \Sigma$ is diagonal.

1. Follow Lines 1 through 15 of Algorithm 2.
2. Define $\eta_g^{(1)} = (\widehat{g}_{11} + \widehat{g}_{12})/(|g_{11}| + |g_{12}|)$ and $\eta_h^{(1)} = (\widehat{h}_{12} + \widehat{h}_{22})/(|h_{12}| + |h_{22}|)$.
3. Define $\eta_k^{(1)} = (\widehat{k}_{11} + \widehat{k}_{12})/(|k_{11}| + |k_{12}|)$ and $\eta_l^{(1)} = (\widehat{l}_{12} + \widehat{l}_{22})/(|l_{12}| + |l_{22}|)$.
4. Define $\eta_g^{(2)} = (\widehat{g}_{21} + \widehat{g}_{22})/(|g_{21}| + |g_{22}|)$ and $\eta_h^{(2)} = (\widehat{h}_{11} + \widehat{h}_{21})/(|h_{11}| + |h_{21}|)$.
5. Define $\eta_k^{(2)} = (\widehat{k}_{21} + \widehat{k}_{22})/(|k_{21}| + |k_{22}|)$ and $\eta_l^{(2)} = (\widehat{l}_{11} + \widehat{l}_{21})/(|l_{11}| + |l_{21}|)$.
6. Define $\eta_{\max}^{(i)} = \max\{\eta_g^{(i)}, \eta_h^{(i)}\}$ for $i = 1, 2$.
7. **if** $\eta_{\max}^{(1)} \leq \tau_\eta$ **and** $\eta_{\max}^{(1)} \leq \eta_{\max}^{(2)}$ **then**
8. Let $\eta_g = \eta_g^{(1)}$, $\eta_h = \eta_h^{(1)}$, $\eta_k = \eta_k^{(1)}$, and $\eta_l = \eta_l^{(1)}$.
9. **else**
10. Let $\eta_g = \eta_g^{(2)}$, $\eta_h = \eta_h^{(2)}$, $\eta_k = \eta_k^{(2)}$, and $\eta_l = \eta_l^{(2)}$.
11. Set $U = UJ$, $V = VJ$, $G = J^TG$, $H = HJ$, $K = J^TK$, and $L = LJ$.
12. **endif**
13. Follow Lines 18 through 28 of Algorithm 2.

Numerical tests in Section 9 show the trade-off between accuracy and performance for different values of τ_η. For now, the following upper bound on the smallest $\eta_{\max}$ that we get is more important.

Proposition 31. *Define $\eta_g^{(i)}$, $\eta_h^{(i)}$, $\eta_k^{(i)}$, and $\eta_l^{(i)}$ as in Algorithm 3 for $i = 1, 2$, and define the corresponding $\eta_{\max}^{(i)}$ as in Definition 19. Then*

$$\eta_{\max}^{\min} = \min\{\eta_{\max}^{(1)}, \eta_{\max}^{(2)}\} \leq \begin{cases} 4\kappa(A) + 2 & \text{if } \max\{\eta_g^{(1)}, \eta_l^{(1)}\} < \infty, \\ 8\kappa(A) & \text{otherwise.} \end{cases}$$

Proof. Suppose that both $\eta_g^{(1)}$ and $\eta_l^{(1)}$ are finite and that $\eta_{\max}^{(1)} > 4\kappa(A) + 2$. Then $\eta_g^{(1)} > 4\kappa(A) + 2$ or $\eta_l^{(1)} > 4\kappa(A) + 2$, and we can assume without loss of generality that the first of the two bounds holds. By applying Lemma 30 we get the bound

$$\eta_g^{(2)} \leq 2(1 - 2(4\kappa(A) + 2)^{-1})^{-1} = \frac{4\kappa(A) + 2}{2\kappa(A)},$$

and then from Proposition 28 the bound $\eta_k^{(2)} \leq 4\kappa(A) + 2$. Hence, we can conclude that $\eta_{\max}^{(2)} \leq 4\kappa(A) + 2$.

Now suppose $\eta_g^{(1)} = \infty$ or $\eta_l^{(1)} = \infty$ and $\eta_{\max}^{(1)} > 8\kappa(A)$, and assume without loss of generality that $\eta_g^{(1)} = \infty$; then by Proposition 22 we have that $\eta_h^{(1)} < \infty$ and $\eta_l^{(1)} < \infty$. Hence, by Definition 19 we must have $\eta_h^{(1)} > 8\kappa(A)$ or $\eta_l^{(1)} > 8\kappa(A)$, so that it follows from Corollary 29 and Proposition 28, respectively, that $\eta_l^{(1)} > 4$. The result is that we can invoke Lemma 30 to see that $\eta_l^{(2)} \leq 4$, followed by Proposition 28 to see that $\eta_h^{(2)} \leq 8\kappa(A)$. Thus, we can conclude that $\eta_{\max}^{(2)} \leq 8\kappa(A)$. □

By combining Theorem 20 with the proposition above, we get the following result.

Theorem 32. *Suppose that we compute all floating-point operations in Algorithm 3 with a precision of at least $\mathcal{O}(\boldsymbol{\varepsilon}\kappa(A)^{-1})$ and use the tolerance $\tau_\eta = 8\kappa(A)$. Then the algorithm computes $\overline{A}'$ stably with respect to the precision $\boldsymbol{\varepsilon}$.*

Theorem 32 shows how to pick the working precision to guarantee an accurate result. But tying the working precision of the algorithm to the condition number of A is impractical and mathematically inelegant. An alternative without a strong a priori guarantee, is to pick a fixed working precision independent of $\kappa(A)$, with two obvious choices. The first choice is to double the precision, which we can motivate as follows. If $A^{(1)}$ is the $p \times q$ input matrix that we have at the beginning of the preprocessing phase; then, with typical bounds, the first compression sets all singular values smaller than $\boldsymbol{\varepsilon} \max\{p, q\}\sigma_{\max}(A^{(1)})$ to zero. Hence, the resulting $A_{12}^{(2)}$ has a condition number bounded by $(\max\{p, q\}\boldsymbol{\varepsilon})^{-1}$. Note, however, that this does not guarantee that the condition number of the upper-triangular 2×2 matrices A_{ij} from the Kogbetliantz phase have the same bound. The second choice is to not increase the working precision. As we will see in Section 9, a large $\eta_{\max}$ is rare, even for highly ill-conditioned A, and a large $\eta_{\max}^{\min}$ even rarer. Furthermore, the numerical results show that the bound from Theorem 20 is pessimistic, and that the relative errors do not scale in proportion to $\eta_{\max}$. In any case, if we fix the working precision, then we can cheaply estimate a posteriori whether the computed $\overline{A}'$ is accurate in two ways. Either by checking if $\eta_{\max}$ or $\max\{\tau_\eta, \eta_{\max}^{\min}\}$ is sufficiently small, or by checking if $|\operatorname{fl}(\overline{P}^T A \overline{Q})_{12}|$ is sufficiently small.

This section ends with the following remarkable result for a final bit of insight into the behavior of the ηs. Although the proof is not obvious, it requires only elementary arithmetic and is omitted for brevity.

Proposition 33. *In exact arithmetic $\eta_g > \eta_h$ and $\eta_l > \eta_k$ cannot hold simultaneously. Furthermore, if $g_{11} \neq 0$ and $l_{22} \neq 0$, then $\eta_g = \eta_h$ and $\eta_l = \eta_k$ can only hold at the same time if $a_{12} = 0$.*

The consequence of the proposition above is that (in exact arithmetic) Algorithm 2 computes either P from L or Q from G. In other words, P and Q are never computed from $K = G\operatorname{adj}(A)$ and $H = \operatorname{adj}(A)L$ at the same time.

6 The extraction phase. In this section we consider the problem of extracting the singular triplets (α, β, γ) from the upper-triangular matrices $(A, B, C) = (SD_\alpha T, SD_\beta, D_\gamma T)$. Without loss of generality, we can focus on the diagonal entries and drop the indices, and consider $a = st\alpha$, $b = s\beta$, $c = t\gamma$, and $\sigma = a/(bc) = \alpha/(\beta\gamma)$ for unknown s, t, α, β, and γ. Typical treatment of the RSVD imposes the condition $\alpha^2 + \beta^2 + \gamma^2 = 1$, but this condition alone generally does not define the singular triplet uniquely. For example, for nonzero b and c we can swap the values of β and γ and adjust s and t accordingly. Another example is when $a = b = c = 1$; then we can pick any $\gamma^2 \in (0, 1)$ and let

$$\alpha^2 = \gamma^2 \frac{1 - \gamma^2}{1 + \gamma^2}, \qquad \beta^2 = \frac{1 - \gamma^2}{1 + \gamma^2}, \qquad s = \beta^{-1}, \quad \text{and} \quad t = \gamma^{-1}.$$

Which further conditions we should impose to make the triplet (α, β, γ) well defined, are unclear. It is also unclear how to compute the triplets in a numerically sound way.

As an alternative, we propose to impose the condition

$$\alpha^2 + \beta^2\gamma^2 = 1 \tag{16}$$

for the normalization of the triplets for the following reasons.

- This condition is the correct homogeneous formulation corresponding to the fraction $\sigma = \alpha/(\beta\gamma)$, and uniquely defines the pair $(\alpha, \beta\gamma)$.
- We know that the RSVs correspond to the nonnegative eigenvalues of the pencil $\left[\begin{smallmatrix} 0 & A \\ A^* & 0 \end{smallmatrix}\right] - \lambda \left[\begin{smallmatrix} BB^* & 0 \\ 0 & C^*C \end{smallmatrix}\right]$; see, e.g., [28, p. 193]. Solving this generalized eigenvalue problem yields the eigenpairs $(\alpha, \pm\beta\gamma)$.
- This condition allows us to express the generalized singular pairs of a QSVD (i.e., an RSVD with $B = I$) in terms of restricted singular triplets with $\beta = 1$ (and $s = 1$).
- Triplets corresponding to zero and infinite singular values can be written as $(0, 1, 1)$, $(1, 0, 0)$, $(1, 0, 1)$, and $(1, 1, 0)$, and all satisfy (16).
- As shown below, we can impose a simple condition to make computing (α, β, γ), s, and t with (16) elegant and straightforward.
- With (16), the pair $(\alpha, \beta\gamma)$ is invariant under the scaling $(\lambda A, \lambda^p B, \lambda^q C)$ of the matrix triplet (A, B, C), where $\lambda > 0$ and $p + q = 1$.

Some flexibility is still left when it comes to computing β, γ, s, and t. One option is to take $|s| = |t|$, or more generally $|s|^q = |t|^p$ with $p + q = 1$, so that

$$a^2 + b^2c^2 = (\alpha^2 + \beta^2\gamma^2)(st)^2 = |s|^{2/p} = |t|^{2/q},$$

and

$$\alpha = |a|(a^2 + b^2c^2)^{-1/2}, \qquad \beta = |b|(a^2 + b^2c^2)^{-p/2}, \quad \text{and} \quad \gamma = |c|(a^2 + b^2c^2)^{-q/2}.$$

Although we have some flexibility when picking p and q, the choice $p = q = 1/2$ is the most natural in absence of an application specific preference. This choice also allows us to reliably compute the triplets (α, β, γ) in floating-point arithmetic for a wide range of triplets (a, b, c) with the algorithm below. A key part of the algorithm is the function hypot(x, y), which computes $(x^2+y^2)^{-1/2}$ without unnecessary overflow or underflow for $x, y \in \mathbb{R}$. The problem that the algorithm addresses, is that we cannot use hypot$(|a|, |b||c|)$ directly if the product $|b||c|$ overflows or underflows. Hence, the algorithm only applies hypot to $|a|$ and $|b||c|$ directly if the latter product is finite and nonzero in floating-point arithmetic. Otherwise, the algorithm first rescales the input triplet by exploiting the scaling invariance.

Algorithm 4 (Extracting restricted singular triplets.)**.**
Input: A triplet (a, b, c), where $\max\{|a|, |b|, |c|\} < \infty$ and $a \neq 0$.
Output: A triplet (α, β, γ) satisfying $\beta\gamma/\alpha = bc/a$ and $\alpha^2 + \beta^2\gamma^2 = 1$.

1. **if** $0 < \text{fl}(|b||c|) < \infty$ **then**
2. Let $|st| = \text{hypot}(|a|, |b||c|)$.
3. Let $\alpha = |a|/|st|$, $\beta = |b|/|st|^{1/2}$, and $\gamma = |c|/|st|^{1/2}$.
4. **else if** $|a|^{1/2} \geq \max\{|b|, |c|\}$ **then**
5. Let $b' = |b|/|a|^{1/2}$, $c' = |c|/|a|^{1/2}$, and $|st| = (1 + (b'c')^2)^{1/2}$.
6. Let $\alpha = 1/|st|$, $\beta = b'/|st|^{1/2}$, and $\gamma = c'/|st|^{1/2}$.
7. **else if** $|b| \geq \max\{|a|^{1/2}, |c|\}$ **then**
8. Let $a' = (|a|/|b|)/|b|$, $c' = |c|/|b|$, and $|st| = \text{hypot}(a', c')$.
9. Let $\alpha = a'/|st|$, $\beta = 1/|st|^{1/2}$, and $\gamma = c'/|st|^{1/2}$.
10. **else if** $|c| \geq \max\{|a|^{1/2}, |b|\}$ **then**
11. Let $a' = (|a|/|c|)/|c|$, $b' = |b|/|c|$, and $|st| = \text{hypot}(a', b')$.
12. Let $\alpha = a'/|st|$, $\beta = b'/|st|^{1/2}$, and $\gamma = 1/|st|^{1/2}$.
13. **end**

7 The postprocessing phase. If the implicit Kogbetliantz iteration from Section 4 converges, then we get the Schur-form RSVD from Theorem 7. Combined with the extraction from Section 6, this form is already useful in its own right, as explained in Section 2. However, if we want the full decomposition from Theorem 1 or Corollary 4, or any of the individual factors Σ_α, Σ_β, Σ_γ, X, Y, S, or T, then further postprocessing is necessary. This necessary postprocessing is nontrivial in the most general case, and requires that the output of the implicit Kogbetliantz iteration is of the form described by Proposition 13. Moreover, some of the postprocessing steps are troublesome in floating-point arithmetic, e.g., due to sensitivity to perturbations, which may affect their reliability. Hence, we consider the postprocessing steps in exact arithmetic in Section 7.1, and discuss some of the numerical challenges in floating-point arithmetic in Section 7.2.

7.1 Postprocessing in exact arithmetic. Suppose that A, B, and C are as in (11), then the first step of the preprocessing phase is to use the transformations from the proof of Proposition 13 to get the structure from (12). The next step is to extract the nonzero restricted singular triplets D_α, D_β, and D_γ with Algorithm 4, and to let

$$\widetilde{\Sigma}_\alpha = \begin{bmatrix} D_\alpha & & & \\ & I & & \\ & & I & \\ & & & I \end{bmatrix}, \qquad \widetilde{\Sigma}_\beta = \begin{bmatrix} D_\beta & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \text{and} \quad \widetilde{\Sigma}_\gamma = \begin{bmatrix} D_\gamma & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix}.$$

Using these Σs, we can decompose our matrix triplet as

$$A = S\widetilde{\Sigma}_\alpha T, \qquad B = S\widetilde{\Sigma}_\beta, \quad \text{and} \quad C = \widetilde{\Sigma}_\gamma T, \tag{17}$$

where S and T are upper triangular. In particular, let $S_{11} = B_{11}D_\beta^{-1}$ and $T_{11} = D_\gamma^{-1}C_{11}$, so that

$$A_{11}C_{11}^{-1} = (S_{11}D_\alpha T_{11})(T_{11}^{-1}D_\gamma^{-1}) = S_{11}D_\alpha D_\gamma^{-1} = B_{11}D_\beta^{-1}D_\alpha D_\gamma^{-1},$$

and define the Schur complement $Z_{1j} = A_{1j} - A_{11}C_{11}^{-1}C_{1j}$; then

$$S = \begin{bmatrix} S_{11} & B_{12} & Z_{13} & Z_{14}C_{44}^{-1} \\ & B_{22} & A_{23} & A_{24}C_{44}^{-1} \\ & & A_{33} & A_{34}C_{44}^{-1} \\ & & & A_{44}C_{44}^{-1} \end{bmatrix}, \quad \text{and} \quad T = \begin{bmatrix} T_{11} & D_\gamma^{-1}C_{12} & D_\gamma^{-1}C_{13} & D_\gamma^{-1}C_{14} \\ & B_{22}^{-1}A_{22} & 0 & 0 \\ & & I & 0 \\ & & & C_{44} \end{bmatrix}. \tag{18}$$

To see that these S and T are correct, consider the leading principal 2×2 blocks of $CA^{-1}B$, given by

$$\begin{aligned} \begin{bmatrix} C_{11}A_{11}^{-1}B_{11} & 0 \\ & 0 \end{bmatrix} &= \begin{bmatrix} C_{11} & C_{12} \\ & 0 \end{bmatrix} \begin{bmatrix} A_{11}^{-1} & -A_{11}^{-1}A_{12}A_{22}^{-1} \\ & A_{22}^{-1} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix} \\ &= \begin{bmatrix} C_{11}A_{11}^{-1}B_{11} & C_{11}A_{11}^{-1}B_{12} - C_{11}A_{11}^{-1}A_{12}A_{22}^{-1}B_{22} + C_{12}A_{22}^{-1}B_{22} \\ & 0 \end{bmatrix}. \end{aligned}$$

Since the $(1,2)$ block must be zero, we have that $A_{12} = A_{11}C_{11}^{-1}C_{12} + B_{12}B_{22}^{-1}A_{22}$, which we can use to verify that

$$\begin{bmatrix} S_{11} & B_{12} \\ & B_{22} \end{bmatrix} \begin{bmatrix} D_\alpha & \\ & I \end{bmatrix} \begin{bmatrix} T_{11} & D_\gamma^{-1}C_{12} \\ & B_{22}^{-1}A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{11}C_{11}^{-1}C_{12} + B_{12}B_{22}^{-1}A_{22} \\ & A_{22} \end{bmatrix}$$

is equal to the leading principal 2×2 blocks of A. The rest of the proof that S and T are of the form in (18) is by direct verification.

Now that we have the decomposition (17), we can plug it back into the Schur-form RSVD from (3)–(4) to get the triplet $(A^{(\ell)}, B^{(\ell)}, C^{(\ell)})$. Here, we use a similar notation as in the preprocessing phase, with a similar numbering of the blocks. Define

$$(X^{(\ell)})^{-T} = \begin{bmatrix} I & & \frac{1}{2}A_{15} & B_{14} & \\ & S & A_{25} & B_{24} & \\ & & A_{35} & B_{34} & \\ & & & B_{44} & \\ & & & & I \end{bmatrix} \quad \text{and} \quad (Y^{(\ell)})^{-1} = \begin{bmatrix} I & & & & \\ & C_{12} & C_{13} & C_{14} & C_{15} \\ & & A_{13} & A_{14} & \frac{1}{2}A_{15} \\ & & & T & \\ & & & & I \end{bmatrix}$$

cf. (7), where S and T are as in (18), and let

$$A^{(\ell+1)} = (X^{(\ell)})^T A^{(\ell)} Y^{(\ell)}, \qquad B^{(\ell+1)} = (X^{(\ell)})^T B^{(\ell)}, \quad \text{and} \quad C^{(\ell+1)} = C^{(\ell)} Y^{(\ell)}.$$

Then the above three matrices are as in (6), but with A_{24}, B_{23}, and C_{24} replaced by $\widetilde{\Sigma}_\alpha$, $\widetilde{\Sigma}_\beta$, and $\widetilde{\Sigma}_\gamma$, respectively. Hence, with the appropriate block permutations we get the matrices

$$A^{(\ell+2)} = \begin{bmatrix} 0 & 0 & D_\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$B^{(\ell+2)} = \begin{bmatrix} D_\beta & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 \\ B_{31}^{(\ell+2)} & B_{32}^{(\ell+2)} & B_{33}^{(\ell+2)} & B_{34}^{(\ell+2)} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$C^{(\ell+2)} = \begin{bmatrix} 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & D_\gamma & 0 & 0 & 0 & 0 & C_{28}^{(\ell+2)} \\ 0 & 0 & 0 & 0 & 0 & 0 & I & C_{38}^{(\ell+2)} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & C_{48}^{(\ell+2)} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & C_{58}^{(\ell+2)} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Next we need to compress $[B_{33}^{(\ell+2)} \; B_{34}^{(\ell+2)}]$ and $[C_{48}^{(\ell+2)}; \; C_{58}^{(\ell+2)}]$ (and transfer the transformations

"through" $A^{(\ell+2)}$) to get $A^{(\ell+3)} = A^{(\ell+2)}$, and $B^{(\ell+3)}$ and $C^{(\ell+3)}$ given by

$$\begin{bmatrix} D_\beta & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 \\ B_{31}^{(\ell+3)} & B_{32}^{(\ell+3)} & R_B & 0 & 0 & 0 \\ B_{41}^{(\ell+3)} & B_{42}^{(\ell+3)} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & I & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & D_\gamma & 0 & \cdots & 0 & 0 & C_{29}^{(\ell+3)} & C_{2,10}^{(\ell+3)} \\ 0 & 0 & 0 & 0 & \cdots & 0 & I & C_{39}^{(\ell+3)} & C_{3,10}^{(\ell+3)} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & R_C & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \end{bmatrix},$$

respectively. Next, take $(X^{(\ell+3)})^{-T}$ as

$$\begin{bmatrix} I & 0 & 0 & 0 & 0 & 0 & -D_\alpha D_\gamma^{-1} C_{29} R_C^{-1} & -D_\alpha D_\gamma^{-1} C_{2,10} & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B_{31} D_\beta^{-1} & B_{32} & R_B & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B_{41} D_\beta^{-1} & B_{42} & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I & -C_{39} R_C^{-1} & -C_{3,10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & R_C^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I \end{bmatrix}$$

and $(Y^{(\ell+3)})^{-1}$ as

$$\begin{bmatrix} I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & D_\gamma^{-1} C_{29} & D_\gamma^{-1} C_{2,10} \\ 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -R_B^{-1} B_{31} D_\beta^{-1} D_\alpha & -R_B^{-1} B_{32} & R_B^{-1} & 0 & 0 & 0 & -R_B^{-1} B_{31} D_\sigma^{-1} C_{29} & -R_B^{-1} B_{31} D_\sigma^{-1} C_{2,10} \\ 0 & 0 & -B_{41} D_\beta^{-1} D_\alpha & -B_{42} & 0 & I & 0 & 0 & -B_{41} D_\sigma^{-1} C_{29} & -B_{41} D_\sigma^{-1} C_{2,10} \\ 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & C_{39} & C_{3,10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_C & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I \end{bmatrix},$$

where we dropped the superscript indices to save horizontal whitespace and $D_\sigma^{-1} = D_\beta^{-1} D_\alpha D_\gamma^{-1}$, and compute $A^{(\ell+4)} = \Sigma_\alpha$, $B^{(\ell+4)} = \Sigma_\beta$, and $C^{(\ell+4)} = \Sigma_\gamma$ with a transformation like in (17). Here, $X^{(\ell+3)}$ is upper triangular if $p_1 = 0$ in Theorem 7, and $Y^{(\ell+3)}$ is lower triangular if $q_5 = 0$ in Theorem 7. We can always assume that we have the former case if we wish, by transforming the input triplet as discussed at the end of the preprocessing phase.

7.2 Challenges in floating-point arithmetic. If B or C is singular before the application of the implicit Kogbetliantz iteration, then we may have $\overline{m}_{ii}$ after convergence that should have been zero in exact arithmetic, but are nonzero due to roundoff errors. It follows that in floating-point arithmetic $\overline{M}$ lacks the desired form of Proposition 12, or at least, has more nonzeros than it should. Consider a 3×3 example, where $A = I$ and C, B, and M are

$$\begin{bmatrix} c_{11}^{(1)} & c_{12}^{(1)} & c_{13}^{(1)} \\ & c_{22}^{(1)} & c_{23}^{(1)} \\ & & c_{33}^{(1)} \end{bmatrix} \begin{bmatrix} b_{11}^{(1)} & b_{12}^{(1)} & b_{13}^{(1)} \\ & 0 & 0 \\ & & 0 \end{bmatrix} = \begin{bmatrix} m_{11}^{(1)} & m_{12}^{(1)} & m_{13}^{(1)} \\ & 0 & 0 \\ & & 0 \end{bmatrix}.$$

After eliminating the $(1, 2)$ element of M in exact arithmetic we get

$$\begin{bmatrix} c_{11}^{(2)} & 0 & c_{13}^{(2)} \\ c_{21}^{(2)} & c_{22}^{(2)} & c_{23}^{(2)} \\ & & c_{33}^{(2)} \end{bmatrix} \begin{bmatrix} b_{11}^{(2)} & 0 & b_{13}^{(2)} \\ b_{21}^{(2)} & 0 & b_{23}^{(2)} \\ & & 0 \end{bmatrix} = \begin{bmatrix} m_{11}^{(2)} & 0 & m_{13}^{(2)} \\ & 0 & 0 \\ & & 0 \end{bmatrix}.$$

Eliminating the $(1, 3)$ element gives us

$$\begin{bmatrix} c_{11}^{(3)} & 0 & 0 \\ c_{21}^{(3)} & c_{22}^{(3)} & c_{23}^{(3)} \\ c_{31}^{(3)} & & c_{33}^{(3)} \end{bmatrix} \begin{bmatrix} b_{11}^{(3)} & 0 & 0 \\ b_{21}^{(3)} & 0 & 0 \\ b_{31}^{(3)} & & 0 \end{bmatrix} = \begin{bmatrix} m_{11}^{(3)} & 0 & 0 \\ & 0 & 0 \\ & & 0 \end{bmatrix},$$

where $b_{23}^{(2)}$ is eliminated at the same time as $b_{13}^{(2)}$ because the vectors $[b_{11}^{(2)}\ b_{21}^{(2)}]$ and $[b_{13}^{(2)}\ b_{23}^{(2)}]$ are parallel. But in floating-point arithmetic we suffer from roundoff errors and can expect to end up with

$$\begin{bmatrix} c_{11}^{(3)} & 0 & 0 \\ c_{21}^{(3)} & c_{22}^{(3)} & c_{23}^{(3)} \\ c_{31}^{(3)} & & c_{33}^{(3)} \end{bmatrix} \begin{bmatrix} b_{11}^{(3)} & 0 & 0 \\ b_{21}^{(3)} & \epsilon & \epsilon \\ b_{31}^{(3)} & & \epsilon \end{bmatrix} = \begin{bmatrix} m_{11}^{(3)} & 0 & 0 \\ & \epsilon & \epsilon \\ & & \epsilon \end{bmatrix},$$

where the $(1, 2)$ and $(1, 3)$ elements are explicitly set to zero. Now, we can ensure that $b_{22}^{(3)} = b_{33}^{(3)} = 0$ by ensuring that Algorithm 2 produces the output of Lemma 11 and by copying the elements of the 2×2 results back to the larger matrices. The necessary changes to Algorithm 2 are to explicitly set $g_{22} = 0$ whenever $c_{11} = 0$, and to set $l_{12} = 0$ whenever $b_{22} = 0$. Still, this does not take care of the nonzero element $b_{23}^{(3)}$, and at the end of the cycle we end up with a second nonzero on the diagonal of B and M. Moreover, explicitly zeroing g_{22} and l_{12} is not automatically better than not doing so if we consider cases with underflow.

For A we face the opposite problem. If A is severely ill-conditioned, then one of its 2×2 submatrices may become singular by applying the rotations. Both the order of evaluation and how the rotations are applied may affect the outcome in these cases. For more information on the latter, see [14, Sec. 3]. This again shows that the condition numbers of the 2-by-2 matrices A_{ij} are important, as we already know from Section 5.

A naive way to get rid of the unwanted nonzeros is the following: sort the diagonal entries of $\overline{M}$ by magnitude after convergence by picking $U = V = I$ or $U = V = J$ in the 2-by-2 RSVD algorithm. Then set diagonal entries, and their corresponding rows, to zero if they are below some threshold, and follow the steps at the end of Section 4.2 with similar thresholding. This strategy appears reasonable at first sight since the $|\overline{m}_{ii}|$ are computed to approximate the singular values of $CA^{-1}B$. But $\overline{M}$ is a product of matrices and a rank decision based on a simple threshold is even less reliable than usual. For example, suppose that $A = I$ and $B = C = \mathrm{diag}(1, 10^{-10})$; then

$M = CA^{-1}B = \mathrm{diag}(1, 10^{-20})$ is numerically singular (in IEEE 754 double precision) for a typical threshold like 2ε, even though B and C are numerically nonsingular. Another example is with $A = I$, $B = \mathrm{diag}(1, 10^{-20})$, and $C = \mathrm{diag}(1, 10^{20})$; now $M = I$ looks nonsingular, while B and C are both numerically singular.

The latter of the two examples above is an example of ill-conditioned restricted singular values, e.g., a relative perturbation of 10^{-15} in either B or C may result in a relative perturbation of 10^5 in M. Still, declaring the small relative entries of B and C to be zero is not automatically reasonable, despite the unreliable entries of M.

Another issue is that the implicit Kogbetliantz iteration is not rank revealing for B and C in general. For example, if $C^{(0)} = \left[\begin{smallmatrix} 1 & 10^{10} \\ 0 & 1 \end{smallmatrix}\right]$, $A^{(0)} = I \in R^{2\times 2}$, and $B^{(0)} = 0 \in R^{2\times 2}$. Then in the odd cycle $Q^{(0)}$ zeros the $(1,2)$ entry of $C^{(0)}$. Hence, $B^{(1)} = (C^{(0)}Q^{(0)})^T$ so that $(P^{(1)})^T = Q^{(0)}$, and thus $C^{(2)} = ((P^{(1)})^T B^{(1)})^T = C^{(0)}$. This example also demonstrates that it does not suffice to just look at the diagonal entries. The diagonal entries of $C^{(0)}$ are both 1, but the condition number of $C^{(0)}$ is approximately 10^{20}.

Now suppose that we have decided that both $\overline{m}_{ii}$ and $\overline{m}_{jj}$ should be zero. If we set diagonal elements of B or C to zero without doing anything else, then we do not automatically get $\overline{m}_{ij} = 0$. That is, we need to be careful not to introduce new nonzeros while zeroing elements.

Although the examples above are not exhaustive, it should be clear by now that determining which entries of B, C, and M should be zero is a nontrivial problem. Similar problems exist for the generalized eigenvalue problem (see, for example, Stewart and Sun [27, Ch. 6]), and their solutions are outside the scope of this work. Although it may appear that Zha's algorithm and the algorithm of Chu et al. do not suffer from these issues, similar problems hide inside the rank decisions in their preprocessing phases. The difference is that we move part of these rank-decision woes to the postprocessing that comes after the Kogbetliantz phase, and which we can omit if the Schur-form RSVD is sufficient for our needs. Furthermore, we are in a better position to spot sensitivity issues after the Kogbetliantz phase and with the help of Algorithm 4.

8 Nonorthogonal transformations. In the first step of the preprocessing phase we compress A using orthonormal transformations that we compute with some URV decomposition. We can sometimes do better, for example for graded matrices, if we allow arbitrary nonsingular transformations. Then we can replace the URV decomposition by a rank-reveal LU or LDU decomposition, or some other rank-revealing decomposition. This is also what Drmač's algorithm for the RSVD [15] uses. The algorithm below summarizes the modified algorithm for a simplified input.

Algorithm 5 (Nonorthogonal RSVD)**.**
Input: Square and upper-triangular $k \times k$ matrices A, B, C, and A nonsingular.
Output: Nonsingular matrices X and Y, and orthonormal matrices U and V, such that X^TAY, X^TBU, and V^TCY are upper-triangular, and $V^TCA^{-1}BU$ is diagonal.

1. Compute $D_B = \mathrm{diag}(\|\boldsymbol{e}_i^T B\|)$ and $D_C = \mathrm{diag}(\|C\boldsymbol{e}_i\|)$.
2. Set $A_1 = D_B^{-1}AD_C^{-1}$, $B_1 = D_B^{-1}B$, and $C_1 = CD_C^{-1}$.
3. Compute the LDU decomposition $\Pi_r A_1 \Pi_c = L_A D_A U_A$ with full pivoting.
4. Compute the RQ decomposition $L_A^{-1}\Pi_r B_1 = R_B Q_B^T$.
5. Compute the QR decomposition $C_1\Pi_c U_A^{-1} = Q_C R_C$.
6. Use Algorithm 1 to compute U, V, P, and Q such that $P^T D_A Q$, $P^T R_B U$, and $V^T R_C Q$ are upper triangular, and $V^T R_C D_A^{-1} R_B U$ is diagonal.
7. Accumulate $U = Q_B U$, $V = Q_C V$, $X^T = P^T L_A^{-1}\Pi_r D_B^{-1}$, and $Y = D_C^{-1}\Pi_c U_A^{-1} Q$.

The benefit of Algorithm 5 is that it does not just produce orthonormal U and V such that $V^TCA^{-1}BU$ is diagonal. It also produces nonsingular X and Y such that X^TAY, X^TBU, and V^TCY are upper triangular. Although Drmač does not discuss it, we can compute such X and Y a posteriori

when using his algorithm. The problem then is that the necessary computations are nontrivial when B and C are nonsingular, and we run into some of the challenges from Section 7.2.

9 Numerical experiments. Our numerical testing consists of three parts. In the first part, we test Algorithm 3 and plot the distribution of the largest magnitudes of the computed $(1,2)$ entries, the largest relative errors, and the values of the $\eta_{\max}$s. We do this for both $\tau_\eta = 1$ and $\tau_\eta = \infty$. That is, both when we always change the columns of U and V if it improves $\eta_{\max}$, and when we never change the columns. We compare the results with similar results from Zha's method. In the second part we consider nonsingular and upper-triangular $n \times n$ matrix triplets, and again plot the distribution of the largest magnitudes of $(1,2)$ elements and the $\eta_{\max}$s, but not the relative errors. This time, we test more tolerances than just $\tau_\eta = 1$ and $\tau_\eta = \infty$, and try to see how the value of τ_η affects the accuracy and the rate of convergence of the implicit Kogbetliantz iteration. In the third and final part, we compare the difference in accuracy between Algorithm 3, Zha's method [29], Drmač's method [15], and the method from Chu, De Lathauwer, and De Moor [8]. We do this just for a small class of matrices for brevity.

For the standard linear algebra routines, such as matrix multiplication, and matrix decompositions, such as QR with column pivoting and the Jacobi SVD, we use Eigen [20]. For the high-precision arithmetic we use Boost Multiprecision [24], which we can use in combination with Eigen in a straightforward manner.

9.1 Testing 2-by-2 RSVDs. We can test Algorithm 3 for a given matrix triplet by computing its result both with double-precision and with high-precision floating-point arithmetic. Then we compare the results and use the high-precision result in place of the exact result. With this approach, we need to be careful when dealing with M, because we need to ensure that the double-precision and high-precision results approximate the same quantities. Hence, we proceed as follows. First, we generate 2×2 upper-triangular matrices A, B, and C in double precision. Each entry has the form $s \cdot 2^p$, where the sign s a Rademacher distributed random variable and the exponent p a uniform random variable in $[-333, 333)$. The range of p is such that the product $M = C\,\mathrm{adj}(A)B$, when computed in higher precision, can still be represented in double precision without overflow or underflow. Next, we take $\overline{M} = \mathrm{fl}(M)$ and use $\overline{M}$ as the input for both the double and the high precision RSVD computation. This ensures that we compute the SVD of the same matrix in both cases, and that the high-precision product $C\,\mathrm{adj}(A)B$ approximates the double precision $\overline{M}$ as well as possible. Since we generate matrices that may have extremely large or small values, we need to ensure that no overflow, underflow, or other numerical difficulties occur by rejecting samples that satisfy one or more of the following conditions.

1. The entry $\overline{m}_{12}$ is nonzero and $\min\{|\overline{u}_{11}|, |\overline{u}_{12}|\} = 0$ or $\min\{|\overline{v}_{11}|, |\overline{v}_{12}|\} = 0$. In this case the computation of the SVD underflows and the zero entries lack a high-relative precision, which means that Fact 1 in Theorem 17 does not hold.

2. The bound for the off-diagonal error

$$\mathrm{fl}\left(\frac{|\overline{\sigma}_{12}| + |\overline{\sigma}_{21}|}{|\overline{m}_{11}| + |\overline{m}_{12}| + |\overline{m}_{22}|}\right) \leq \frac{\sqrt{2}\|E\|_F}{\|\overline{M}\|_F}(1+4\boldsymbol{\varepsilon}) \leq \frac{2\|E\|_2}{\|\overline{M}\|_2}(1+4\boldsymbol{\varepsilon}) < 281\boldsymbol{\varepsilon}$$

   does not hold, where

$$E = \overline{\Sigma} - \Sigma = \mathrm{fl}(\overline{V}^T\overline{MU}) - V^T\overline{M}U = \delta V^T\overline{M}U + V^T\overline{M}\delta U + F$$

   for some F containing roundoff errors. This is because we require $\overline{V}$ and $\overline{U}$ to diagonalize $\overline{M}$ properly, and because we know the facts from the proof of Theorem 17 imply that $\|E\| \leq$

$(2 \cdot 66 + 2 \cdot 4)\varepsilon\|\overline{M}\|$ in the absence of underflow and overflow. [4].

3. The singular values of $\overline{M}$ are too close to each other, say within a relative distance of 10^{-14}, in which case the columns of the double and high-precision U and V may be in a different order and the results hard to compare.

Rejecting triplets makes the sampling nonuniform, but also allows us to test a larger range of floating-point numbers as entries of A, B, and C.

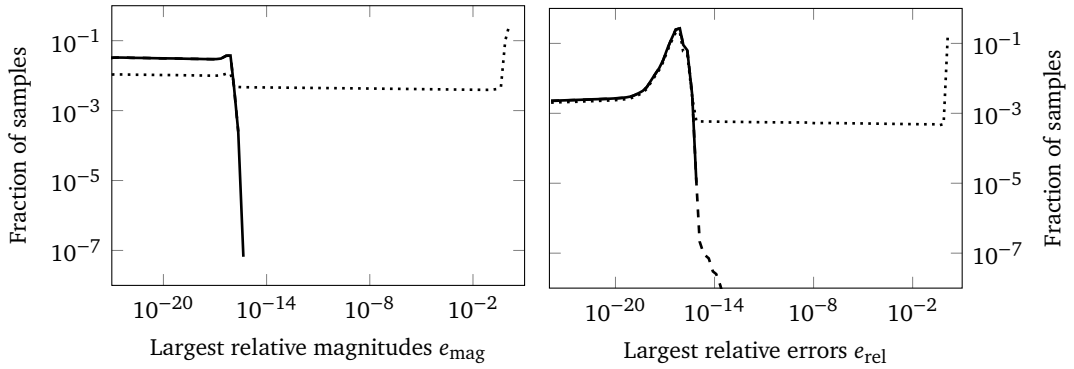

Figure 1: Normalized histograms of the $\log_{10}(e_{\text{mag}})$ (left) and $\log_{10}(e_{\text{rel}})$ (right) of the results from Algorithm 3 with $\tau_\eta = 1$ (solid) and $\tau_\eta = \infty$ (dashed), as well as Zha's method (dotted). The figures do not show the individual bars for the histograms to avoid clutter and because each figure shows the results for three different algorithms. Moreover, the histograms only include samples $e_{\text{mag}}, e_{\text{rel}} \geq 10^{-24}$, which means they only show the tail ends of the true distribution of the samples.

Given the inputs A, B, C, and $\overline{M}$, we compute $\overline{A}'$, $\overline{B}'$, $\overline{C}'$, $\overline{H}'$, and $\overline{K}'$ with Algorithm 3 in double precision, both with $\tau_\eta = 1$ and with $\tau_\eta = \infty$. Then we compute A', B', C', H', and K' with 100 decimals of precision, while making sure the high-precision computations take the same conditional branches in the algorithm as the double-precision computations. Given these results, we can compute the maximum of the relative magnitudes as

$$e_{\text{mag}} = \max\left\{\frac{|\overline{A}'_{12}|}{\|A\|_F}, \frac{|\overline{B}'_{12}|}{\|B\|_F}, \frac{|\overline{C}'_{12}|}{\|C\|_F}, \frac{|\overline{H}'_{12}|}{\|A\|_F\|B\|_F}, \frac{|\overline{K}'_{12}|}{\|A\|_F\|C\|_F}\right\}.$$

We can also compute the maximum of the relative errors as

$$e_{\text{rel}} = \max\left\{\frac{\|\overline{A}' - A'\|_F}{\|A\|_F}, \frac{\|\overline{B}' - B'\|_F}{\|B\|_F}, \frac{\|\overline{C}' - C'\|_F}{\|C\|_F}, \frac{\|\overline{H}' - H'\|_F}{\|A\|_F\|B\|_F}, \frac{\|\overline{K}' - K'\|_F}{\|A\|_F\|C\|_F}\right\}.$$

Next, we run Zha's algorithm — which, for nonsingular input, can be thought of as always computing $\overline{P}$ from $\overline{G}$ and $\overline{Q}$ from $\overline{L}$ in Algorithm 2 — for comparison and compute the same quantities. For each set of inputs and outputs we pick the corresponding maxima of the e_{mag}s and the e_{rel}s, for a total of 10^9 generated sets, and plot their distributions in Figure 1. We also keep track of the $\overline{\eta}_{\max}$; see Figure 2.

Figure 1 shows that Algorithm 3 with $\tau_\eta = \infty$ (which is identical to Algorithm 2) zeroes the $(1, 2)$ entries with high precision. In fact, even though $\overline{\eta}_{\max}$ can become large, as shown in Figure 2, the largest $(1, 2)$ entry is still $\mathcal{O}(\varepsilon)$. For the relative errors we do see a difference between $\tau_\eta = 1$ and $\tau_\eta = \infty$. In particular, the maximum relative errors remain close to ε when we always let the algorithm change the columns of U and V when it improves $\overline{\eta}_{\max}$, while the maximum relative error

[4]The goal of this section is not to verify this claim. Moreover, the proof of this part of Theorem 17 is straightforward and identical to the proof from Bai and Demmel [4, Thm. 3.1].

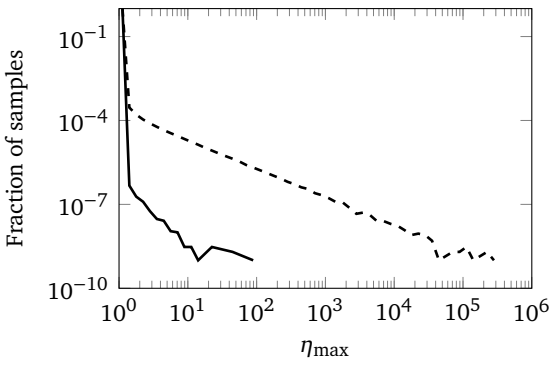

Figure 2: The distribution of $\overline{\eta}_{\max}$ in Algorithm 3 with $\tau_\eta = 1$ (solid) and $\tau_\eta = \infty$ (dashed).

can become two or three orders of magnitude larger when we never let the algorithm improve $\overline{\eta}_{\max}$. Still, these larger errors stay small and they are also rare.

The numerical results are an upper bound for the errors that we see in the $\overline{A}'$, which is encouraging since the error analysis for $\overline{A}'$ depends on $\overline{\eta}_{\max}$, and none of our bounds suggest that $\overline{\eta}_{\max}$ has to be small. All we know is that $\eta_{\max} \lesssim 4\kappa(A)$ when $\tau_\eta = 1$, but $\kappa(A)$ may be over 10^{400} for the generated As due to the numerical range of their entries. Yet, Figure 2 shows that we can expect $\overline{\eta}_{\max} \ll 4\kappa(A)$ in practice when $\tau_\eta = 1$. Even when $\tau_\eta = \infty$, $\overline{\eta}_{\max}$ with values larger than $\mathcal{O}(1)$ are rare. Furthermore, even though the largest $\overline{\eta}_{\max}$ we get exceeds 10^5, the largest error in Figure 1 is considerably smaller than the bound $\sqrt{2}(44.5 + 342 \cdot 10^5)\boldsymbol{\varepsilon} \approx 4.8 \cdot 10^7 \boldsymbol{\varepsilon}$ from Theorem 20 (where the factor $\sqrt{2}$ comes from using the Frobenius norm instead of using the 2-norm).

Zha's algorithm does not take the $\overline{\eta}$s into account, and we see that it may have large relative errors or fail to zero $(1, 2)$ entries as a result. The reason for this is that by ignoring the $\overline{\eta}$s, the algorithm effectively always computes P from BU and Q from V^TC. This in turn means that the rotations may be computed from numerical noise, because the elements of the input matrices may be extremely small or extremely large. Hence, ignoring the $\overline{\eta}$s should be less of an issue if the input matrices are well conditioned. Moreover, Figure 1 just shows the tail ends of the error distributions; hence, the results do not imply that Zha's algorithm fails to find an accurate solution in, e.g., a third of the cases.

9.2 Testing the implicit Kogbetliantz iteration. We can test the implicit Kogbetliantz iteration by computing the RSVs of generated matrices with known restricted singular values and prescribed condition numbers. Specifically, we wish to generate upper-triangular $n \times n$ matrices

$$A = PS\Sigma_\alpha TQ^T, \qquad B = PS\Sigma_\beta U^T \quad \text{and} \quad C = V\Sigma_\gamma TQ^T;$$

where P, Q, U, and V are orthonormal; S and T are upper triangular, nonsingular, and such that $s_{ii} = t_{ii}$ for $i = 1, \ldots, n$; and $\Sigma_\alpha = \operatorname{diag}(\alpha_1, \ldots, \alpha_n)$, $\Sigma_\beta = \operatorname{diag}(\beta_1, \ldots, \beta_n)$, and $\Sigma_\gamma = \operatorname{diag}(\gamma_1, \ldots, \gamma_n)$ with $\alpha_i, \beta_i, \gamma_i \geq 0$ and $\alpha_i^2 + \beta_i^2\gamma_i^2 = 1$ for $i = 1, \ldots, n$. Furthermore, we wish to control the condition numbers of S and T, and control the ratios between the largest and smallest RSVs, αs, βs, and γs. There exists no unique way to generate such matrices, and we limit the discussion to matrices randomly generated by the procedure described below.

The first step is to generate the desired RSVs σ_i for $i = 1, \ldots, n$, and compute $\alpha_i^2 = \sigma_i^2(\sigma_i^2 + 1)^{-1}$ and $\beta_i^2\gamma_i^2 = (\sigma_i^2 + 1)^{-1}$. We need to be careful when picking the σ_i, because if all $\sigma_i \geq 1$, then all α_i are $\mathcal{O}(1)$. Likewise, if all $\sigma_i \leq 1$, then all $\beta_i\gamma_i$ are $\mathcal{O}(1)$. These situations are undesirable if we want a large variation in the range of the α_i and $\beta_i\gamma_i$. We can avoid this problem by using a scaled version of the diagonals generated by LAPACKs xLATME. In particular, given a condition number κ, we randomly pick one of the following sets of σ_i:

1. $\sigma_1 = \sqrt{\kappa}$ and $\sigma_2 = \ldots \sigma_n = 1/\sqrt{\kappa}$;

2. $\sigma_1 = \cdots = \sigma_{n-1} = \sqrt{\kappa}$ and $\sigma_n = 1/\sqrt{\kappa}$;
3. $\sigma_i = \kappa^{1/2-(i-1)/(n-1)}$;
4. $\sigma_i = \kappa^{1/2}(1 - (i-1)/(n-1) \cdot (1 - 1/\kappa))$;
5. set the σ_i to random numbers in the interval $(\kappa^{-1/2}, \kappa^{1/2})$ such that the $\log(\sigma_i)$ are uniformly distributed in the interval $(-1/2\log(\kappa), 1/2\log(\kappa))$.

While these σ_i determine our α_i and the products $\beta_i\gamma_i$, we still need to select the individual values of β_i and γ_i. To do so, we generate random numbers δ_i so that the $\log(\delta_i)$ are uniformly distributed in $(-1/8\log(\kappa), 1/8\log(\kappa))$, and set $\beta_i = \sqrt{\beta_i\gamma_i}/\delta_i$ and $\gamma_i = \sqrt{\beta_i\gamma_i} \cdot \delta_i$. The result is that $\sigma_1/\sigma_n = \kappa$, $\alpha_1/\alpha_n = \beta_n\gamma_n/(\beta_1\gamma_1) = \sqrt{\kappa}$, and the ratios between the largest and smallest largest and smallest β_is and γ_is are bounded by $\sqrt{\kappa}$.

The next step is to generate suitable S and T. Exploratory testing showed that xLATME produced severely ill-conditioned A, even for small n. Another idea is to generate $\widetilde{\sigma}_i$ in the same way as above, generate random orthonormal [25] $\widetilde{U}$ and $\widetilde{V}$, and take $\widetilde{S}$ as the upper-triangular factor of the QR decomposition of $\widetilde{U}\,\mathrm{diag}(\widetilde{\sigma}_i)\widetilde{V}$. We can generate $\widetilde{T}$ likewise, and then compute $S = \widetilde{S}D$ and $T = D^{-1}\widetilde{T}$, where $D = \mathrm{diag}((\widetilde{t}_{ii}/\widetilde{s}_{ii})^{1/2})$. The result is that $\widetilde{s}_{ii}\widetilde{t}_{ii} = s_{ii}^2 = t_{ii}^2$, although the condition numbers of S and T are no longer exactly equal to the condition numbers of $\widetilde{S}$ and $\widetilde{T}$, respectively.

Once we have S and T, we generate random orthonormal $\widetilde{P}$, $\widetilde{Q}$, $\widetilde{U}$, and $\widetilde{V}$, and compute $\widetilde{A} = \widetilde{P}^T S\Sigma_\alpha T\widetilde{Q}$, $\widetilde{B} = \widetilde{P}S\Sigma_\beta\widetilde{U}^T$, and $\widetilde{C} = \widetilde{V}\Sigma_\gamma T\widetilde{Q}^T$. The final step to get the triplet (A, B, C), is to run the preprocessing from Section 3 on the triplet $(\widetilde{A}, \widetilde{B}, \widetilde{C})$. Specifically, since A is nonsingular, we can get A from $\widetilde{A}$ with a QR decomposition, B from $\widetilde{B}$ with an RQ decomposition, and C from an appropriately transformed $\widetilde{C}$ with another QR decomposition.

We generate the input matrices A, B, and C in high-precision arithmetic, and again with 100 decimals of precision. We denote the κ used to generate the RSVs by κ_σ, and the κ used to generate $\widetilde{S}$ and $\widetilde{T}$ by κ_{ST}. Then, we run the implicit Kogbetliantz iteration from Algorithm 1 in double precision. We stop the iterations after at most 50 pairs of cycles, or earlier if we detect convergence after an even cycle as described in Section 4. In particular, we stop earlier when $\rho = \max_{ij}\rho_{ij}$ satisfies $0.99\rho_{\min} < \rho < 0.01$, where ρ_{ij} is as in (10). For solving the 2×2 RSVD we use Algorithm 3, and consider the tolerances $\tau_\eta = 1$, 1.01, 4, 10, 100, 10^4, 10^8, ∞. The tolerance $\tau_\eta = 4$ is of interest because of Lemma 30, and $\tau_\eta = 10^8$ because of the connection between Theorem 32 and because $10^8 \approx \varepsilon^{-1/2}$.

For every input triplet we record the number of cycle pairs before stopping, and also compute the following quantities. First, the maximum errors in the computed orthogonal matrices given by

$$e_{PQUV} = \max\{\|\overline{P}^T\overline{P} - I\|_F, \|\overline{Q}^T\overline{Q} - I\|_F, \|\overline{U}^T\overline{U} - I\|_F, \|\overline{V}^T\overline{V} - I\|_F\}/\sqrt{n}.$$

Second, the maximum errors in the transformations given by

$$e_{ABC} = \max\left\{\frac{\|\overline{P}^T A\overline{Q} - \overline{A}'\|_F}{\|A\|_F}, \frac{\|\overline{P}^T B\overline{U} - \overline{B}'\|_F}{\|B\|_F}, \frac{\|\overline{V}^T C\overline{Q} - \overline{C}'\|_F}{\|C\|_F}\right\},$$

where $\overline{A}'$, $\overline{B}'$, and $\overline{C}'$ are the output matrices of Algorithm 1. Third, e_{tril}, the largest of the Frobenius norms of the strictly lower-triangular parts of $\overline{P}^T A\overline{Q}$, $\overline{P}^T B\overline{U}$, and $\overline{V}^T C\overline{Q}$. And fourth, $e_\chi = \max_{i\in\{1,\ldots,n\}}\chi(\sigma_i, \overline{\sigma}_i)$, where

$$\chi(\sigma, \overline{\sigma}) = |\alpha\overline{\beta\gamma} - \overline{\alpha}\beta\gamma| = \frac{|\sigma - \overline{\sigma}|}{\sqrt{1+\sigma^2}\sqrt{1+\overline{\sigma}^2}} = \frac{|\sigma^{-1} - \overline{\sigma}^{-1}|}{\sqrt{1+\sigma^{-2}}\sqrt{1+\overline{\sigma}^{-2}}}$$

is the chordal metric and measures the distance between the exact and computed RSVs; see, e.g., Stewart and Sun [27, Ch. 6]. See Table 1 for the results.

Table 1: The relation between the swap tolerance, the number of iterations until convergence, and corresponding errors. Each entry has the form: mean (maximum). The mean (maximum) condition numbers of the generated matrices can be found in Table 2.

				Tolerance τ_η							
n	κ_{ST}	κ_σ	**Measure**	**1**	**1.01**	**4**	**10**	10^2	10^4	10^8	∞
10	10	10^4	pairs of cycles	8.33 (29)	5.66 (17)	3.67 (9)	3.67 (9)	3.64 (9)	3.68 (9)	3.65 (9)	3.65 (9)
			$\log_{10}(e_{PQUV})$	−14.9 (−14.3)	−14.9 (−14.4)	−15.0 (−14.4)	−15.0 (−14.5)	−15.0 (−14.5)	−15.0 (−14.5)	−15.0 (−14.5)	−15.0 (−14.5)
			$\log_{10}(e_{ABC})$	−14.4 (−13.6)	−14.6 (−14.1)	−14.8 (−14.3)	−14.8 (−14.3)	−14.8 (−14.2)	−14.8 (−14.3)	−14.8 (−14.2)	−14.8 (−14.3)
			$\log_{10}(e_{\text{tril}})$	−15.2 (−14.7)	−15.2 (−14.8)	−15.2 (−14.8)	−15.3 (−14.8)	−15.3 (−14.8)	−15.2 (−14.8)	−15.3 (−14.8)	−15.3 (−14.8)
			$\log_{10}(e_\chi)$	−8.34 (−0.14)	−15.3 (−2.00)	−15.5 (−14.0)	−15.5 (−14.0)	−15.5 (−14.0)	−15.5 (−13.9)	−15.5 (−13.9)	−15.5 (−14.1)
	10^5	10^4	pairs of cycles	8.27 (36)	5.41 (23)	3.72 (11)	3.66 (10)	3.59 (8)	3.58 (9)	3.58 (9)	3.57 (10)
			$\log_{10}(e_{PQUV})$	−14.9 (−14.2)	−14.9 (−14.2)	−15.0 (−14.4)	−15.0 (−14.4)	−15.0 (−14.5)	−15.0 (−14.5)	−15.0 (−14.4)	−15.0 (−14.4)
			$\log_{10}(e_{ABC})$	−14.4 (−13.6)	−14.6 (−13.8)	−14.7 (−14.3)	−14.7 (−14.3)	−14.7 (−14.3)	−14.7 (−14.3)	−14.7 (−14.2)	−14.7 (−14.3)
			$\log_{10}(e_{\text{tril}})$	−15.3 (−14.6)	−15.4 (−14.5)	−15.7 (−14.9)	−15.8 (−14.8)	−15.9 (−14.8)	−15.9 (−14.8)	−15.9 (−14.9)	−15.9 (−14.8)
			$\log_{10}(e_\chi)$	−7.53 (−0.11)	−12.8 (−7.16)	−12.8 (−7.71)	−12.8 (−7.45)	−12.7 (−7.80)	−12.7 (−7.37)	−12.7 (−7.19)	−12.8 (−7.84)
50	10	10^4	pairs of cycles	31.6 (50)	26.6 (50)	4.43 (11)	4.42 (10)	4.47 (10)	4.44 (10)	4.42 (12)	4.42 (11)
			$\log_{10}(e_{PQUV})$	−14.3 (−13.8)	−14.3 (−13.8)	−14.6 (−14.2)	−14.6 (−14.2)	−14.6 (−14.1)	−14.6 (−14.2)	−14.6 (−14.2)	−14.6 (−14.1)
			$\log_{10}(e_{ABC})$	−13.4 (−12.7)	−13.5 (−12.6)	−14.2 (−13.6)	−14.2 (−13.6)	−14.2 (−13.6)	−14.2 (−13.6)	−14.2 (−13.6)	−14.2 (−13.5)
			$\log_{10}(e_{\text{tril}})$	−14.6 (−14.2)	−14.6 (−14.1)	−14.8 (−14.5)	−14.8 (−14.4)	−14.8 (−14.5)	−14.8 (−14.5)	−14.8 (−14.5)	−14.8 (−14.4)
			$\log_{10}(e_\chi)$	−2.85 (−0.02)	−12.8 (−0.87)	−14.8 (−13.9)	−14.8 (−13.8)	−14.8 (−13.9)	−14.8 (−13.9)	−14.8 (−13.9)	−14.8 (−13.9)
	10^5	10^4	pairs of cycles	31.3 (50)	24.9 (50)	5.00 (21)	4.55 (13)	4.31 (11)	4.30 (10)	4.29 (11)	4.29 (10)
			$\log_{10}(e_{PQUV})$	−14.3 (−13.5)	−14.3 (−13.5)	−14.6 (−13.9)	−14.6 (−14.1)	−14.6 (−14.1)	−14.6 (−14.1)	−14.6 (−14.1)	−14.6 (−14.0)
			$\log_{10}(e_{ABC})$	−13.4 (−12.7)	−13.5 (−12.5)	−14.1 (−13.3)	−14.2 (−13.5)	−14.2 (−13.5)	−14.2 (−13.4)	−14.2 (−13.6)	−14.2 (−13.5)
			$\log_{10}(e_{\text{tril}})$	−14.7 (−14.0)	−14.7 (−14.0)	−15.1 (−14.4)	−15.2 (−14.5)	−15.3 (−14.6)	−15.3 (−14.5)	−15.3 (−14.5)	−15.3 (−14.5)
			$\log_{10}(e_\chi)$	−3.22 (−0.02)	−10.7 (−0.10)	−12.5 (−7.19)	−12.5 (−8.22)	−12.5 (−6.93)	−12.5 (−7.67)	−12.5 (−7.95)	−12.5 (−7.78)

We see that convergence is slow when $\tau_\eta = 1$, and that a small tolerance does not improve the errors. The slow convergence is expected, since we increase the maximum angle of the rotations whenever we multiply U and V by J to improve $\overline{\eta}_{\max}$. As a result, we may not have convergence before the cutoff point of 50 iterations and thus also have large e_χ. That low tolerances do not improve the remaining errors is more interesting, but may be explained by the following two observations. First, more roundoff errors get accumulated when we performs more cycles; second, larger values of $\overline{\eta}_{\max}$ do not affect the accuracy of the results. The latter matches with the observations from the previous section; that is, Algorithm 3 typically computes the 2×2 RSVD with high relative accuracy, even when $\tau_\eta = \infty$.

The table also shows us that we can dramatically improve the rate of convergence with a small increase of τ_η. For example, we see substantial improvements for $\tau_\eta = 1.01$ and already achieve a near optimal rate of convergence for $\tau_\eta = 4$. The observation that we can get fast convergence for small tolerances (larger than 1) is expected if we look at Figure 2. In particular, $\overline{\eta}_{\max}$ is close to 1 most of the time, and large $\overline{\eta}_{\max}$ are so rare that any practical difference between the larger values of τ_η is would be surprising.

One caveat here is that the results from this section depend on the way we generate the test matrices, and on the condition numbers we choose. We consider more variations of κ_{ST} and κ_σ in the next section, of which we only consider the pairs resulting in the best and worst conditioned matrix triplets in this section. In any case, we make sure to pick the κs such that $\kappa(A)$ is never more than 10^{-12}; see Table 2 for the condition numbers of the matrices generated for the results in Table 1.

Table 2: The mean (maximum) condition numbers of the matrices generated for the tests/results in Table 1.

n	κ_{ST}	κ_σ	$\log_{10}(\kappa(A))$	$\log_{10}(\kappa(B))$	$\log_{10}(\kappa(C))$
10	10	10^4	2.84 (4.00)	1.86 (3.03)	1.82 (3.20)
10	10^5	10^4	9.24 (12.0)	6.31 (9.02)	5.22 (8.69)
50	10	10^4	2.95 (3.99)	2.07 (2.92)	2.04 (3.05)
50	10^5	10^4	9.53 (12.0)	6.60 (8.91)	5.49 (8.49)

9.3 Comparison with other methods. In the last part we preprocessed the triplet $(\widetilde{A}, \widetilde{B}, \widetilde{C})$ in high-precision arithmetic to get the triplet (A, B, C). In this part we generate the former triplet in the same way, but we do the preprocessing in double-precision arithmetic instead. The purpose of this change is to try and see how different rank-revealing decompositions of A affect the accuracy of the computed RSVs (even though A is full rank). Moreover, we would like to see how these results compare to the existing methods.

We use three different methods for the preprocessing. The first two methods use the approach described in Section 3, the first with a QR decomposition with column pivoting for the compression of A, and the second with a Jacobi based SVD for the compression of A. The third method uses an LDU decomposition as described in Algorithm 5. All three methods use the implicit Kogbetliantz iteration and Algorithm 3 with $\tau_\eta = \infty$. For the existing algorithms we have Drmač's algorithm [15] and the CSD stage from the algorithm by Chu, De Lathauwer, and De Moor (CLM) [8, Sec. 3.2]. To make the latter as accurate as possible, we implement the required CS decomposition with a Jacobi-type SVD instead of the QR based approach implied by the authors. We omit the results of Zha's algorithm here, because we already have the results from Section 9.1. See Table 3 for an overview of the results.

The table show that the nonorthogonal preprocessing from Algorithm 5 yields the most accurate results, or close to the most accurate results. QR with pivoting is slightly behind the former in

Table 3: The means (maxima) of $\log_{10}(e_\chi)$ for different preprocessing approaches and RSVD algorithms.

			Preprocessing method/RSVD algorithm				
n	κ_{ST}	κ_σ	**ColPivQR**	**SVD**	**LDU**	**Drmač**	**CLM**
10	10	10^4	−15.4 (−13.7)	−15.2 (−13.3)	−15.4 (−14.1)	−15.5 (−14.2)	−15.2 (−13.7)
		10^{12}	−15.0 (−12.1)	−14.8 (−11.5)	−15.0 (−12.5)	−14.1 (−11.8)	−14.5 (−10.8)
		10^{20}	−14.3 (−10.4)	−14.0 (−9.24)	−14.2 (−10.6)	−9.93 (−5.34)	−13.3 (−8.07)
	10^3	10^4	−13.0 (−9.50)	−12.7 (−8.94)	−13.1 (−9.81)	−13.1 (−9.81)	−12.7 (−9.32)
		10^{12}	−13.0 (−8.06)	−12.8 (−6.64)	−13.1 (−8.24)	−12.7 (−8.24)	−12.5 (−7.06)
	10^5	10^4	−9.44 (−4.38)	−9.14 (−3.83)	−9.54 (−4.71)	−9.54 (−4.71)	−9.35 (−4.41)
50	10	10^4	−14.8 (−13.9)	−14.4 (−13.0)	−14.8 (−13.9)	−14.8 (−14.1)	−14.6 (−13.9)
		10^{12}	−14.7 (−12.6)	−14.3 (−11.3)	−14.6 (−12.6)	−13.0 (−10.1)	−14.1 (−11.3)
		10^{20}	−13.6 (−10.7)	−13.1 (−8.97)	−13.5 (−10.9)	−8.71 (−5.03)	−12.4 (−8.09)
	10^3	10^4	−13.0 (−10.0)	−12.2 (−8.04)	−13.1 (−10.3)	−13.1 (−10.3)	−12.7 (−9.81)
		10^{12}	−12.8 (−8.26)	−12.2 (−6.32)	−12.9 (−8.56)	−12.2 (−8.56)	−12.3 (−7.52)
	10^5	10^4	−9.36 (−5.37)	−8.46 (−3.55)	−9.51 (−5.44)	−9.51 (−5.44)	−9.20 (−4.96)

accuracy, and the new algorithm is the least accurate when using the SVD in the preprocessing phase. A possible explanation is that the SVD introduces a larger error than QR with pivoting. If true, it would suggest an interesting trade-off between a better rank decision with the SVD, and better performance and more accurate RSVs with the QR decomposition. Drmač's method does comparatively well as long as κ_σ is not too large, but is well behind in accuracy when κ_σ is large. The results for the method from Chu, De Lathauwer, and De Moor [8] (with the modified CS decomposition) are between the QR and SVD results in the best case, and slightly worse than the SVD result in the worst case.

10 Conclusions. This work introduced a new method for computing the RSVD of a matrix triplet with an implicit Kogbetliantz-type iteration. The main contributions consist of a new generalized Schur-form RSVD that we can compute with orthonormal transformations only, a new preprocessing phase that requires fewer transformations and fewer rank decisions than existing methods, and a new 2×2 triangular RSVD algorithm with favorable numerical properties. We found that the latter is numerically stable in the sense of Theorem 17 and Theorem 32. A further contribution is a new approach to extract restricted singular triplets. This approach has theoretical and practical benefits, but requires atypical scaling of the triplets.

We found that the value of $\eta_{\max}$ is critical in assessing the accuracy of the results computed by Algorithms 2 and 3. Specifically, we can both use $\eta_{\max}$ a priori through bounds, and a posteriori as the amplification factor of the errors in $\overline{U}$ and $\overline{V}$. Numerical experiments show that we can typically keep the values of the $\eta_{\max}$ small. In the rare cases that $\eta_{\max}$ is large, the results show that we can still expect the 2×2 RSVDs to have small backward errors. In fact, none of the results suggests that the bounds from Section 5 are sharp, and that the bounds are pessimistic in practice. This means that the numerical results provide empirical evidence that we can have a numerically stable RSVD without having to increase the working precision for the 2×2 RSVD.

Areas where further improvements are desirable or necessary, and potential directions for future research include the following. Better stopping conditions, a cache friendly and parallelized implementation of the Kogbetliantz phase, and most of all, a numerically sound postprocessing

phase. The latter in particular represents a major deficiency of the new algorithm, although the postprocessing phase is only necessary to compute the full RSVD. In other words, we may skip the postprocessing phase in applications where the Schur-form RSVD suffices.

We should also note that there are techniques for and aspects of existing Jacobi methods (for other matrix decompositions) that we ignored in this paper. These include, for example, the scaling of the input matrices to avoid overflow or underflow, the effects of diagonal scaling of the input matrices on the relative accuracy of the results, efficient implementations using blocking for better cache usage, quasi-cycles for faster convergence, adaptive pivot strategies, whether preconditioning is possible and useful, the (relative) accuracy of the algorithm as a whole for arbitrary or structured matrices, etc. See, e.g., [12, 16, 17] and references therein for more information.

11 Acknowledgments. I would like to thank Andreas Frommer and Michiel Hochstenbach for helpful discussions and their comments.

References.

[1] G. E. Adams, A. W. Bojanczyk, and F. T. Luk, *Computing the PSVD of two* 2×2 *triangular matrices*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 366–382, https://doi.org/10.1137/s0895479891221575.

[2] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, 3 ed., Jan. 1999, https://doi.org/10.1137/1.9780898719604.

[3] Z. Bai, *The CSD, GSVD, their applications and computations*, Tech. Report 958, University of Minnesota, 1992, http://hdl.handle.net/11299/1875. IMA Preprint Series.

[4] Z. Bai and J. W. Demmel, *Computing the generalized singular value decomposition*, SIAM J. Sci. Comput., 14 (1993), pp. 1464–1486, https://doi.org/10.1137/0914085.

[5] A. W. Bojanczyk, L. Magnus Ewerbring, F. T. Luk, and P. van Dooren, *An accurate product SVD algorithm*, Signal Process., 25 (1991), pp. 189–201, https://doi.org/10.1016/0165-1684(91)90062-n.

[6] R. P. Brent, F. T. Luk, and C. Van Loan, *Computation of the generalized singular value decomposition using mesh-connected processors*, in Real-Time Signal Processing VI, no. TR 82-528, SPIE, Nov. 1983, https://doi.org/10.1117/12.936442.

[7] J. P. Charlier, M. Vanbegin, and P. Van Dooren, *On efficient implementations of Kogbetliantz's algorithm for computing the singular value decomposition*, Numer. Math., 52 (1987), pp. 279–300, https://doi.org/10.1007/bf01398880.

[8] D. Chu, L. De Lathauwer, and B. De Moor, *On the computation of the restricted singular value decomposition via the cosine-sine decomposition*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 580–601, https://doi.org/10.1137/s0895479898346983.

[9] B. De Moor and H. Zha, *A tree of generalizations of the ordinary singular value decomposition*, Linear Algebra Appl., 147 (1991), pp. 469–500, https://doi.org/10.1016/0024-3795(91)90243-p.

[10] B. L. R. De Moor and G. H. Golub, *Generalized singular value decompositions: a proposal for a standardized nomenclature*, Tech. Report NA-89-05, University of Stanford, 1989.

[11] B. L. R. De Moor and G. H. Golub, *The restricted singular value decomposition: Properties and applications*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 401–425, https://doi.org/10.1137/0612029.

[12] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač, *Computing the singular value decomposition with high relative accuracy*, Linear Algebra Appl., 299 (1999), pp. 21–80, https://doi.org/10.1016/s0024-3795(99)00134-2.

[13] J. Demmel and K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1204–1245, https://doi.org/10.1137/0613074.

[14] Z. Drmac, *Implementation of Jacobi rotations for accurate singular value computation in floating point arithmetic*, SIAM J. Sci. Comput., 18 (1997), pp. 1200–1222, https://doi.org/10.1137/s1064827594265095.

[15] Z. Drmac, *New accurate algorithms for singular value decomposition of matrix triplets*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1026–1050, https://doi.org/10.1137/s0895479897321209.

[16] Z. Drmač and K. Veselić, *New fast and accurate Jacobi SVD algorithm. I*, SIAM J. Matrix Anal. Appl., 29 (2008), pp. 1322–1342, https://doi.org/10.1137/050639193.

[17] Z. Drmač and K. Veselić, *New fast and accurate Jacobi SVD algorithm. II*, SIAM J. Matrix Anal. Appl., 29 (2008), pp. 1343–1362, https://doi.org/10.1137/05063920x.

[18] R. D. Fierro, P. C. Hansen, and P. S. K. Hansen, *UTV tools: Matlab templates for rank-revealing UTV decompositions*, Numer. Algorithms, 20 (1999), pp. 165–194, https://doi.org/10.1023/a:1019112103049.

[19] G. E. Forsythe and P. Henrici, *The cyclic Jacobi method for computing the principal values of a complex matrix*, Trans. Amer. Math. Soc., 94 (1960), pp. 1–1, https://doi.org/10.1090/s0002-9947-1960-0109825-2.

[20] G. Guennebaud, B. Jacob, et al., *Eigen v3*. http://eigen.tuxfamily.org, 2010.

[21] E. R. Hansen, *On cyclic Jacobi methods*, J. Soc. Indust. Appl. Math., 11 (1963), pp. 448–459, https://doi.org/10.1137/0111032.

[22] M. T. Heath, A. J. Laub, C. C. Paige, and R. C. Ward, *Computing the singular value decomposition of a product of two matrices*, SIAM J. Sci. and Stat. Comput., 7 (1986), pp. 1147–1159, https://doi.org/10.1137/0907078.

[23] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 2 ed., Jan. 2002, https://doi.org/10.1137/1.9780898718027.

[24] J. Maddock, C. Kormanyos, et al., *Boost multiprecision*. https://www.boost.org, 2013.

[25] F. Mezzadri, *How to generate random matrices from the classical compact groups*, Notices Amer. Math. Soc., 54 (2007), pp. 592–604.

[26] C. C. Paige, *Computing the generalized singular value decomposition*, SIAM J. Sci. and Stat. Comput., 7 (1986), pp. 1126–1146, https://doi.org/10.1137/0907077.

[27] G. W. Stewart and J.-g. Sun, *Matrix Perturbation Theory*, Academic Press, 1 ed., 1990, https://www.elsevier.com/books/matrix-perturbation-theory/stewart/978-0-08-092613-1.

[28] H. Zha, *The restricted singular value decomposition of matrix triplets*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 172–194, https://doi.org/10.1137/0612014.

[29] H. Zha, *A numerical algorithm for computing the restricted singular value decomposition of matrix triplets*, Linear Algebra Appl., 168 (1992), pp. 1–25, https://doi.org/10.1016/0024-3795(92)90285-i.

A Proof of Lemma 11.

1. Let A, B, and C be nonsingular; then the proof follows the proof of Proposition 10.

2. If $C = \left[\begin{smallmatrix} 0 & c_{12} \\ 0 & c_{22} \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} b_{11} & b_{12} \\ 0 & 0 \end{smallmatrix}\right]$, then $M = \left[\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$. Since $c_{11} = b_{22} = 0$, we take $P = Q = J$ and compute U and V are such that $(V^TC)_{22} = (BU)_{11} = 0$. Hence

$$V^TCQ = \begin{bmatrix} c'_{11} & 0 \\ 0 & 0 \end{bmatrix}, \qquad P^TAQ = \begin{bmatrix} a'_{11} & 0 \\ a'_{21} & \underline{a'_{22}} \end{bmatrix}, \quad \text{and} \quad P^TBU = \begin{bmatrix} 0 & 0 \\ 0 & b'_{22} \end{bmatrix}.$$

3. Let $C = \left[\begin{smallmatrix} c_{11} & c_{12} \\ 0 & 0 \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$; then $M = \left[\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ and xLASV2 computes $U = V = I$. Now $1 = c_{\max} > s_{\max} = 0$ and the algorithm does not swap the columns of U and V. See below for the computation of P and Q, but note that P^TBU is zero.

4. Let $C = \left[\begin{smallmatrix} c_{11} & c_{12} \\ 0 & \underline{c_{22}} \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$; then $M = \left[\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ and xLASV2 computes $U = V = I$. Now $1 = c_{\max} > s_{\max} = 0$ and the algorithm does not swap the columns of U and V. See below for the computation of P and Q, but note that V^TCQ is nonsingular lower triangular.

5. Let $C = \left[\begin{smallmatrix} c_{11} & c_{12} \\ 0 & 0 \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} b_{11} & b_{12} \\ 0 & 0 \end{smallmatrix}\right] \neq 0$; then $M = c_{11}a_{22}B \neq 0$ and xLASV2 computes $|V| = I$ and U such that $(BU)_{12} = 0$. Now $1 = c_{\max} \geq s_{\max}$ and the algorithm does not swap the columns of U and V. It follows that $|h_{12}| + |h_{22}| = |l_{12}| + |l_{22}| = 0$ ($\eta_h = \eta_l = \infty$), $\eta_g = 1$, and $1 \leq \eta_k < \infty$. Hence, the algorithm computes Q from $G = V^TC = C$ and P from $K = V^TC\,\mathrm{adj}(A) = C\,\mathrm{adj}(A)$, which results in

$$V^TCQ = \begin{bmatrix} c_{11}' & 0 \\ 0 & 0 \end{bmatrix}, \qquad P^TAQ = \begin{bmatrix} a'_{11} & 0 \\ a'_{21} & \underline{a'_{22}} \end{bmatrix}, \quad \text{and} \quad P^TBU = \begin{bmatrix} b'_{11} & 0 \\ b'_{21} & 0 \end{bmatrix}.$$

   The product P^TAQ is lower triangular since, by construction,

$$0 = |(KP)_{12}| = |\boldsymbol{e}_1^T(V^TCQ)\,\mathrm{adj}(P^TAQ)\boldsymbol{e}_2| = |c'_{11}(P^TAQ)_{12}|.$$

   Likewise, b'_{11} is nonzero because $|c'_{11}a'_{22}b'_{11}|$ equals the largest singular value of M.

6. Let $C = \left[\begin{smallmatrix} c_{11} & c_{12} \\ 0 & \underline{c_{22}} \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} b_{11} & b_{12} \\ 0 & 0 \end{smallmatrix}\right] \neq 0$; then $M = c_{11}a_{22}B \neq 0$ and the computation of U, V, P, and Q proceeds as above, except that V^TCQ is nonsingular lower triangular.

7. Let $C = \left[\begin{smallmatrix} c_{11} & c_{12} \\ 0 & 0 \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} b_{11} & b_{12} \\ 0 & \underline{b_{22}} \end{smallmatrix}\right]$; then $M = \left[\begin{smallmatrix} m_{11} & m_{12} \\ 0 & 0 \end{smallmatrix}\right]$ and xLASV2 computes $|V| = I$ and U such that $(MU)_{12} = 0$. Now $1 = c_{\max} > s_{\max} = |u_{12}|$ since $m_{11} \neq 0$ implies that $|U| \neq |J|$, and the algorithm does not swap the columns of U and V. It follows that $\eta_g = 1$ and $1 \leq \eta_k, \eta_h, \eta_l < \infty$, so that the algorithm always computes Q from $G = V^TC = C$, but may compute P from either $K = V^TC\,\mathrm{adj}(A) = C\,\mathrm{adj}(A)$ or $L = BU$. The result is

$$V^TCQ = \begin{bmatrix} c'_{11} & 0 \\ 0 & 0 \end{bmatrix}, \qquad P^TAQ = \begin{bmatrix} a'_{11} & 0 \\ a'_{21} & \underline{a'_{22}} \end{bmatrix}, \quad \text{and} \quad P^TBU = \begin{bmatrix} b'_{11} & 0 \\ b'_{21} & \underline{b'_{22}} \end{bmatrix}.$$

   If P was computed from K, then P^TAQ is lower triangular for the same reason as in Item 5, and P^TBU is lower triangular because $0 = |(V^TMU)_{12}| = |c'_{11}a'_{22}(P^TBU)_{12}|$. If P was computed from L, then P^TAQ is lower triangular because $0 = |(V^TMU)_{12} = |c'_{11}b'_{22}(P^TAQ)_{12}|$.

8. Let $C = \left[\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} 0 & b_{12} \\ 0 & \underline{b_{22}} \end{smallmatrix}\right]$; then $M = \left[\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ and xLASV2 computes $U = V = I$. Now $1 = c_{\max} > s_{\max} = 0$ and the algorithm does not swap the columns of U and V. Computing P and Q is the same as below, but note that V^TCQ is zero.

9. Let $C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}$; then $M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ and xLASV2 computes $U = V = I$. Now $1 = c_{\max} > s_{\max} = 0$ and the algorithm does not swap the columns of U and V. Computing P and Q is the same as below, but note that $P^T BU$ is nonsingular lower triangular.

10. Let $C = \begin{bmatrix} 0 & c_{12} \\ 0 & c_{22} \end{bmatrix} \neq 0$ and $B = \begin{bmatrix} 0 & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}$; then $M = a_{11} b_{22} C \neq 0$ and xLASV2 computes V such that $(V^T C)_{22} = 0$ and $|U| = |J|$. Now $|v_{11}| = c_{\max} \leq s_{\max} = 1$, but the M is singular and no swap takes place. It follows that $|h_{12}| + |h_{22}| = |l_{12}| + |l_{22}| = 0$ ($\eta_h = \eta_l = \infty$), and $1 \leq \eta_g, \eta_k < \infty$. As a result, P and Q are computed from G and K, respectively. Furthermore $|Q| = |J|$, so that $|P| = |J|$ as well. The result is

$$V^T CQ = \begin{bmatrix} \underline{c'_{11}} & 0 \\ 0 & 0 \end{bmatrix}, \qquad P^T AQ = \begin{bmatrix} \underline{a'_{11}} & 0 \\ a'_{21} & \underline{a'_{22}} \end{bmatrix}, \quad \text{and} \quad P^T BU = \begin{bmatrix} \underline{b'_{11}} & 0 \\ b'_{21} & 0 \end{bmatrix}.$$

11. Let $C = \begin{bmatrix} 0 & c_{12} \\ 0 & c_{22} \end{bmatrix} \neq 0$ and $B = \begin{bmatrix} \underline{b_{11}} & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}$; then $M = a_{11} b_{22} C \neq 0$ and the computation of P, Q, U, and V proceeds as above, except that $P^T BU$ is nonsingular lower triangular.

12. Let $C = \begin{bmatrix} \underline{c_{11}} & c_{12} \\ 0 & \underline{c_{22}} \end{bmatrix}$ and $B = \begin{bmatrix} 0 & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}$; then $M = \begin{bmatrix} 0 & m_{12} \\ 0 & \underline{m_{22}} \end{bmatrix}$ and xLASV2 computes $|U| = |J|$ and V such that $(V^T M)_{22} = (V^T M)_{22} = 0$. Now $|v_{11}| = c_{\max} < s_{\max} = 1$ since $m_{22} \neq 0$ implies that $|V| \neq I$. However, M is singular and no swap takes place. It follows that $1 \leq \eta_g, \eta_k < \infty$ and $\eta_h = \eta_l = 1$, so that the algorithm always computes Q from $G = V^T C$, and P from $K = V^T C \operatorname{adj}(A)$. The result is

$$V^T CQ = \begin{bmatrix} \underline{c'_{11}} & 0 \\ c'_{21} & \underline{c'_{22}} \end{bmatrix}, \qquad P^T AQ = \begin{bmatrix} \underline{a'_{11}} & 0 \\ a'_{21} & \underline{a'_{22}} \end{bmatrix}, \quad \text{and} \quad P^T BU = \begin{bmatrix} \underline{b'_{11}} & 0 \\ b'_{21} & 0 \end{bmatrix}.$$

13. Let $C = \begin{bmatrix} \underline{c_{11}} & c_{12} \\ 0 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 0 & b_{12} \\ 0 & \underline{b_{22}} \end{bmatrix}$; then $M = \begin{bmatrix} 0 & m_{12} \\ 0 & 0 \end{bmatrix}$. When $m_{12} = 0$, xLASV2 computes $U = V = I$ and no swaps are necessary. Furthermore, in this case $\eta_g = \eta_l = 1$ so that P and Q are computed from B and C, respectively, resulting in

$$V^T CQ = \begin{bmatrix} \underline{c'_{11}} & 0 \\ 0 & 0 \end{bmatrix}, \qquad P^T AQ = \begin{bmatrix} \underline{a'_{11}} & 0 \\ a'_{21} & \underline{a'_{22}} \end{bmatrix}, \quad \text{and} \quad P^T BU = \begin{bmatrix} 0 & 0 \\ 0 & \underline{b'_{22}} \end{bmatrix}.$$

Again, $P^T AQ$ is lower triangular because $0 = |V^T MU| = |b'_{22} c'_{11} (P^T AQ)_{12}|$ If $m_{12} \neq 0$, then xLASV2 computes $|U| = |J|$ and $|V| = I$. Now $c_{\max} = s_{\max} = 1$ and the algorithm does not swap the columns of U and V. Without the swap, $\eta_g = 1$, $|h_{11}| + |h_{12}| = 0$ ($\eta_h = \infty$), $1 \leq \eta_k < \infty$, and $|l_{12}| + |l_{22}| = 0$ ($\eta_l = \infty$). Hence, the algorithm computes Q from $G = V^T C = C$ and P from $K = V^T C \operatorname{adj}(A) = C \operatorname{adj}(A)$, resulting in

$$V^T CQ = \begin{bmatrix} \underline{c'_{11}} & 0 \\ 0 & 0 \end{bmatrix}, \qquad P^T AQ = \begin{bmatrix} \underline{a'_{11}} & 0 \\ a'_{21} & \underline{a'_{22}} \end{bmatrix}, \quad \text{and} \quad P^T BU = \begin{bmatrix} \underline{b_{11}}' & 0 \\ b'_{21} & 0 \end{bmatrix}.$$

See Item 5 to see why $P^T AQ$ is lower triangular, and why $b'_{11} \neq 0$.