Numer Algor
DOI 10.1007/s11075-008-9215-6

ORIGINAL PAPER

Computing the complete CS decomposition

Brian D. Sutton

Received: 20 October 2007 / Accepted: 21 May 2008

Abstract An algorithm for computing the complete CS decomposition of a partitioned unitary matrix is developed. Although the existence of the CS decomposition (CSD) has been recognized since 1977, prior algorithms compute only a reduced version. This reduced version, which might be called a 2-by-1 CSD, is equivalent to two simultaneous singular value decompositions. The algorithm presented in this article computes the complete 2-by-2 CSD, which requires the simultaneous diagonalization of all four blocks of a unitary matrix partitioned into a 2-by-2 block structure. The algorithm appears to be the only fully specified algorithm available. The computation occurs in two phases. In the first phase, the unitary matrix is reduced to bidiagonal block form, as described by Sutton and Edelman. In the second phase, the blocks are simultaneously diagonalized using techniques from bidiagonal SVD algorithms of Golub, Kahan, Reinsch, and Demmel. The algorithm has a number of desirable numerical features.

Keywords CS decomposition · Generalized singular value decomposition

Mathematics Subject Classifications (2000) 65F15 · 15A23 · 15A18

B. D. Sutton (✉)
Department of Mathematics,
Randolph-Macon College,
P.O. Box 5005, Ashland,
VA 23005, USA
e-mail: bsutton@rmc.edu

1 Introduction

The *complete CS decomposition* (CSD) applies to any m-by-m matrix X from the unitary group $U(m)$, viewed as a 2-by-2 block matrix,

$$X = \begin{array}{c} \\ p \\ m-p \end{array} \begin{array}{c} \begin{array}{cc} q & m-q \end{array} \\ \left[\begin{array}{c|c} X_{11} & X_{12} \\ \hline X_{21} & X_{22} \end{array}\right] \end{array}.$$

For convenience, we assume $q \le p$ and $p + q \le m$. A complete CS decomposition has the form

$$X = \begin{bmatrix} U_1 & \\ & U_2 \end{bmatrix} \left[\begin{array}{cc|cc} C & S & 0 & 0 \\ 0 & 0 & I_{p-q} & 0 \\ \hline -S & C & 0 & 0 \\ 0 & 0 & 0 & I_{m-p-q} \end{array}\right] \begin{bmatrix} V_1 & \\ & V_2 \end{bmatrix}^*, \tag{1.1}$$

$$C = \operatorname{diag}(\cos(\theta_1), \ldots, \cos(\theta_q)), \quad S = \operatorname{diag}(\sin(\theta_1), \ldots, \sin(\theta_q)),$$

in which $\theta_1, \ldots, \theta_q \in [0, \frac{\pi}{2}]$, $U_1 \in U(p)$, $U_2 \in U(m-p)$, $V_1 \in U(q)$, and $V_2 \in U(m-q)$. The letters *CS* in the term *CS decomposition* come from *cosine-sine*.

The major contribution of this paper is an algorithm for computing (1.1). We believe this to be the only fully specified algorithm available for computing the complete CS decomposition. Earlier algorithms compute only a reduced form, the "2-by-1" CSD, which is defined in the next section. The algorithm developed in this article is based on well known bidiagonal SVD algorithms [6, 8, 9] and has a number of desirable numerical properties.

The algorithm proceeds in two phases.

Phase I: *Bidiagonalization.* In the special case $p = q = \frac{m}{2}$, the decomposition is

$$X = \begin{bmatrix} P_1 & \\ & P_2 \end{bmatrix} \begin{bmatrix} B_{11}^{(0)} & B_{12}^{(0)} \\ B_{21}^{(0)} & B_{22}^{(0)} \end{bmatrix} \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}^*, \tag{1.2}$$

in which $B_{11}^{(0)}$ and $B_{21}^{(0)}$ are upper bidiagonal, $B_{12}^{(0)}$ and $B_{22}^{(0)}$ are lower bidiagonal, and P_1, P_2, Q_1, and Q_2 are q-by-q unitary. We say that the middle factor is a real orthogonal matrix in *bidiagonal block form*. (See Definition 1.1.)

Phase II: *Diagonalization.* The CSD of $\begin{bmatrix} B_{11}^{(0)} & B_{12}^{(0)} \\ B_{21}^{(0)} & B_{22}^{(0)} \end{bmatrix}$ is computed,

$$\begin{bmatrix} B_{11}^{(0)} & B_{12}^{(0)} \\ B_{21}^{(0)} & B_{22}^{(0)} \end{bmatrix} = \begin{bmatrix} U_1 & \\ & U_2 \end{bmatrix} \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} V_1 & \\ & V_2 \end{bmatrix}^*.$$

Combining the factorizations gives the CSD of X,

$$X = \begin{bmatrix} P_1 U_1 & \\ & P_2 U_2 \end{bmatrix} \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} Q_1 V_1 & \\ & Q_2 V_2 \end{bmatrix}^*. \tag{1.3}$$

Phase I is a finite-time procedure first described in [18] and Phase II is an iterative procedure based on ideas from bidiagonal SVD algorithms [6, 8, 9].

Some of the earliest work related to the CSD was completed by Jordan, Davis, and Kahan [4, 5, 12]. The CSD as we know it today and the term *CS decomposition* first appeared in a pair of articles by Stewart [16, 17]. Computational aspects of the 2-by-1 CSD are considered in [3, 13, 14, 17, 19] and later articles. A "sketch" of an algorithm for the complete CSD can be found in a paper by Hari [11], but few details are provided. For general information and more references, see [2, 10, 15].

1.1 Complete versus 2-by-1 CS decomposition

Most commonly available CSD algorithms compute what we call the *2-by-1 CS decomposition* of a matrix $\hat{X}$ with orthonormal columns partitioned into a 2-by-1 block structure. In the special case $p = q = \frac{m}{2}$, $\hat{X}$ has the form

$$\hat{X} = \begin{matrix} q \\ q \end{matrix} \begin{bmatrix} \overset{q}{\hat{X}_{11}} \\ \hat{X}_{21} \end{bmatrix},$$

and the CSD is

$$\hat{X} = \begin{bmatrix} U_1 & \\ & U_2 \end{bmatrix} \begin{bmatrix} C \\ -S \end{bmatrix} V_1^*.$$

A naive algorithm for computing the 2-by-1 CSD is to compute two SVD's,

$$\begin{cases} \hat{X}_{11} = U_1 C V_1^* \\ \hat{X}_{21} = (-U_2) S V_1^*, \end{cases}$$

reordering rows and columns and adjusting signs as necessary to make sure that the two occurrences of V_1^* are identical and that $C^2 + S^2 = I$. This works in theory if no two singular values of $\hat{X}_{11}$ are repeated, but in practice it works poorly when there are clustered singular values. Still, the basic idea can form the backbone of an effective algorithm for the 2-by-1 CSD [17, 19].

Unfortunately, many algorithms for the 2-by-1 CSD do not extend easily to the complete 2-by-2 CSD. The problem is the more extensive sharing of singular vectors evident below (still assuming $p = q = \frac{m}{2}$):

$$\begin{cases} X_{11} = U_1 C V_1^* & X_{12} = U_1 S V_2^* \\ X_{21} = (-U_2) S V_1^* & X_{22} = U_2 C V_2^*. \end{cases} \tag{1.4}$$

All four unitary matrices U_1, U_2, V_1, and V_2 play dual roles, providing singular vectors for two different blocks of X. Enforcing these identities has proven difficult over the years.

Our algorithm, unlike the naive algorithm, is designed to compute the four SVD's in (1.4) *simultaneously*, so that no discrepancies ever arise.

1.2 Applications

Unlike existing 2-by-1 CSD algorithms, the algorithm developed here fully solves Jordan's problem of angles between linear subspaces of $\mathbb{R}^n$ [12]. If the columns of matrices X and Y are orthonormal bases for two subspaces of $\mathbb{R}^n$, then the *principal angles* and *principal vectors* between the subspaces can be computed in terms of the SVD of $X^T Y$ [15]. The complete CSD, equivalent to four SVD's, simultaneously provides principal vectors for these subspaces and their orthogonal complements.

In addition, our algorithm can be specialized to compute the two-by-one CSD and hence has application to the generalized singular value decomposition.

1.3 Numerical properties

The algorithm is designed for numerical stability. All four blocks of the partitioned unitary matrix are treated simultaneously and with equal regard, and no cleanup procedure is necessary at the end of the algorithm. In addition, a new representation for orthogonal matrices with a certain structure guarantees orthogonality, even on a floating-point architecture [18].

1.4 Efficiency

As with popular bidiagonal SVD algorithms, Phase I (bidiagonalization) is often more expensive than Phase II (diagonalization). For the special case $p = q = \frac{m}{2}$, bidiagonalization requires about $2m^3$ flops—about $\frac{8}{3}q^3$ flops to bidiagonalize each block of $\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$ and about $\frac{4}{3}q^3$ flops to accumulate each of P_1, P_2, Q_1, and Q_2, for a total of about $4\left(\frac{8}{3}q^3\right) + 4\left(\frac{4}{3}\right)q^3 = 2m^3$ flops.

1.5 Overview of the algorithm

1.5.1 Bidiagonal block form

During Phase I, the input unitary matrix is reduced to *bidiagonal block form*. A matrix in this form is real orthogonal and has a specific sign pattern. Bidiagonal block form was independently formulated by Sutton in 2005 [18]. Some similar results appear in a 1993 paper by Watkins [20]. The matrix structure and a related decomposition have already been applied to a problem in random matrix theory by Edelman and Sutton [7].

Definition 1.1 Given $\theta = (\theta_1, \ldots, \theta_q) \in [0, \frac{\pi}{2}]^q$ and $\phi = (\phi_1, \ldots, \phi_{q-1}) \in [0, \frac{\pi}{2}]^{q-1}$, let $c_i = \cos\theta_i$, $s_i = \sin\theta_i$, $c'_i = \cos\phi_i$, and $s'_i = \sin\phi_i$. Define $B_{ij}(\theta, \phi)$, $i, j = 1, 2$, to be q-by-q bidiagonal matrices, as follows.

$$\left[\begin{array}{c|c} B_{11}(\theta,\phi) & B_{12}(\theta,\phi) \\ \hline B_{21}(\theta,\phi) & B_{22}(\theta,\phi) \end{array}\right]$$

$$= \left[\begin{array}{cccc|cccc} c_1 & -s_1 s'_1 & & & s_1 c'_1 & & & \\ & c_2 c'_1 & \ddots & & c_2 s'_1 & s_2 c'_2 & & \\ & & \ddots & -s_{q-1}s'_{q-1} & & \ddots & \ddots & \\ & & & c_q c'_{q-1} & & & c_q s'_{q-1} & s_q \\ \hline -s_1 & -c_1 s'_1 & & & c_1 c'_1 & & & \\ & -s_2 c'_1 & \ddots & & -s_2 s'_1 & c_2 c'_2 & & \\ & & \ddots & -c_{q-1}s'_{q-1} & & \ddots & \ddots & \\ & & & -s_q c'_{q-1} & & & -s_q s'_{q-1} & c_q \end{array}\right]. \qquad (1.5)$$

Any matrix of the form

$$\left[\begin{array}{c|c} B_{11}(\theta,\phi) & B_{12}(\theta,\phi) \\ \hline B_{21}(\theta,\phi) & B_{22}(\theta,\phi) \end{array}\right]$$

is said to be in *bidiagonal block form* and is necessarily real orthogonal.

To clarify (1.5), the $(q-1, q-1)$ entry of $B_{12}(\theta, \phi)$ is $s_{q-1}c'_{q-1}$, and the $(q-1, q-1)$ entry of $B_{22}(\theta, \phi)$ is $c_{q-1}c'_{q-1}$. Also, if $q = 1$, then the matrices are defined by

$$\left[\begin{array}{c|c} B_{11}(\theta,\phi) & B_{12}(\theta,\phi) \\ \hline B_{21}(\theta,\phi) & B_{22}(\theta,\phi) \end{array}\right] = \left[\begin{array}{c|c} c_1 & s_1 \\ \hline -s_1 & c_1 \end{array}\right].$$

As stated in the definition, any matrix whose entries satisfy the relations of (1.5) is necessarily real orthogonal. The reverse is true as well—any orthogonal matrix X with the bidiagonal structure and sign pattern of (1.5) is expressible in terms of some θ and ϕ. (This is implicit in [7, 18].) Furthermore, every unitary matrix is equivalent to a matrix in bidiagonal block form, as stated in the next theorem.

Theorem 1.2 *Given any m-by-m unitary matrix X and integers p, q such that $0 \le q \le p$ and $p + q \le m$, there exist matrices $P_1 \in U(p)$, $P_2 \in U(m-p)$, $Q_1 \in U(q)$, and $Q_2 \in U(m-q)$ such that*

$$X = \left[\begin{array}{cc} P_1 & \\ & P_2 \end{array}\right] \left[\begin{array}{cc|cc} B_{11}(\theta,\phi) & B_{12}(\theta,\phi) & 0 & 0 \\ 0 & 0 & I_{p-q} & 0 \\ \hline B_{21}(\theta,\phi) & B_{22}(\theta,\phi) & 0 & 0 \\ 0 & 0 & 0 & I_{m-p-q} \end{array}\right] \left[\begin{array}{cc} Q_1 & \\ & Q_2 \end{array}\right]^*$$

for some $\theta = (\theta_1, \ldots, \theta_q) \in [0, \frac{\pi}{2}]^q$ and $\phi = (\phi_1, \ldots, \phi_{q-1}) \in [0, \frac{\pi}{2}]^{q-1}$.

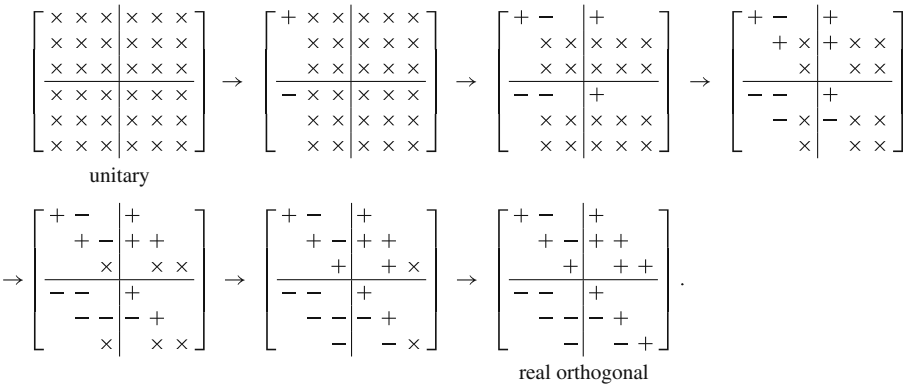

Fig. 1 Reduction to bidiagonal block form

A proof of the theorem has already been published in [7, 18], along with an algorithm for computing the decomposition. The algorithm applies pairs of Householder reflectors to the left and right of X, causing the structure to evolve as in Fig. 1. This serves as Phase I of the CSD algorithm.

1.5.2 Simultaneous SVD steps

Phase II of the algorithm simultaneously applies the bidiagonal SVD algorithm of Golub and Reinsch [9, 10] to each of the four blocks of the matrix produced by Phase I.

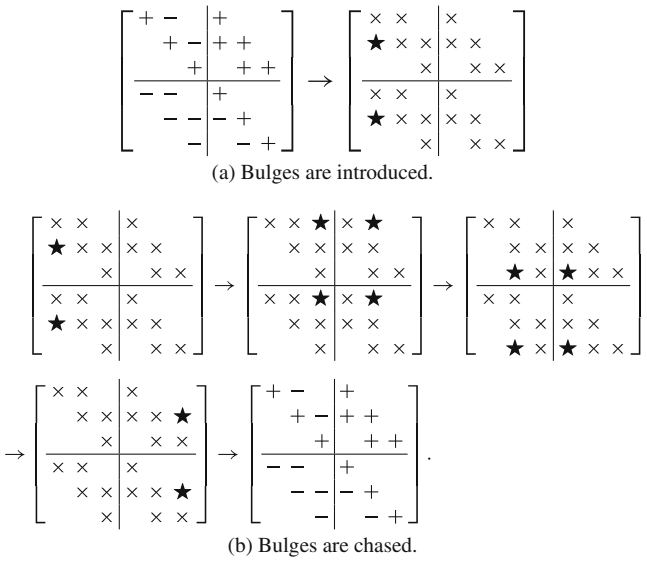

Fig. 2 CSD step

The bidiagonal SVD algorithm is an iterative scheme. Given an initial bidiagonal matrix $B^{(0)}$, the algorithm produces a sequence $B^{(0)} \to B^{(1)} \to B^{(2)} \to \cdots \to \Sigma$ converging to a diagonal matrix of singular values. Implicitly, the step from $B^{(n)}$ to $B^{(n+1)}$ involves a QR factorization of $(B^{(n)})^T B^{(n)} - \sigma^2 I$, for some appropriately chosen $\sigma \geq 0$, but in practice the matrix $(B^{(n)})^T B^{(n)}$ is never explicitly formed. Instead, the transformation from $B^{(n)}$ to $B^{(n+1)}$ is accomplished through a sequence of Givens rotations. The first Givens rotation introduces a "bulge," and the subsequent rotations "chase the bulge" away.

Our algorithm applies this idea simultaneously to all four blocks to execute a *CSD step*. First, two bulges are introduced by a Givens rotation (Fig. 2a), and then the bulges are chased away, also by Givens rotations (Fig. 2b). The end result is a new matrix in bidiagonal block form whose blocks tend to be closer to diagonal than the original blocks.

1.5.3 The driver routine

The algorithm as a whole proceeds roughly as follows.

- Execute Algorithm **bidiagonalize** to transform X to bidiagonal block form. (See Fig. 1).
- Until convergence,
  - Execute Algorithm **csd_step** to apply four simultaneous SVD steps. (See Fig. 2.)

The algorithm as a whole is represented by Fig. 3.

Matrices in bidiagonal block form may be represented implicitly in terms of θ and ϕ from Definition 1.1. In fact, the overall algorithm implicitly represents the sequence of Fig. 3b as

$$\left(\theta^{(0)}, \phi^{(0)}\right) \to \left(\theta^{(1)}, \phi^{(1)}\right) \to \left(\theta^{(2)}, \phi^{(2)}\right) \to \cdots \to \left(\theta^{(N)}, \phi^{(N)}\right).$$

$$X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \to \begin{bmatrix} B_{11}^{(0)} & B_{12}^{(0)} \\ B_{21}^{(0)} & B_{22}^{(0)} \end{bmatrix}$$

(a) Reduction to bidiagonal block form

$$\begin{bmatrix} B_{11}^{(0)} & B_{12}^{(0)} \\ B_{21}^{(0)} & B_{22}^{(0)} \end{bmatrix} \to \begin{bmatrix} B_{11}^{(1)} & B_{12}^{(1)} \\ B_{21}^{(1)} & B_{22}^{(1)} \end{bmatrix} \to \begin{bmatrix} B_{11}^{(2)} & B_{12}^{(2)} \\ B_{21}^{(2)} & B_{22}^{(2)} \end{bmatrix} \to \cdots \to \begin{bmatrix} B_{11}^{(N)} & B_{12}^{(N)} \\ B_{21}^{(N)} & B_{22}^{(N)} \end{bmatrix} = \begin{bmatrix} C & S \\ -S & C \end{bmatrix}$$

(b) Iteration of the CSD step

Fig. 3 Computing the complete CSD

The implicitly represented matrices are exactly orthogonal, even in floating-point. The process stops when $\phi^{(N)}$ is sufficiently close to $(0, \ldots, 0)$; then the blocks of (1.5) are diagonal up to machine precision.

1.6 Overview of the article

The remainder of the article is organized as follows.

Section	Title
2	Phase I: Algorithm **bidiagonalize**
3	Reviewing and extending the SVD step
4	Phase II: Algorithm **csd_step**
5	Algorithm **csd**
6	On numerical stability

The final section contains results of numerical tests on a BLAS/LAPACK-based implementation, which is available from the author's web site.

2 Phase I: Algorithm bidiagonalize

Phase I of the CSD algorithm is to transform the partitioned unitary matrix X to bidiagonal block form.

Specification 2.1 *Given an m-by-m unitary matrix X and integers p, q with $0 \le q \le p$ and $p + q \le m$,* **bidiagonalize**(X, p, q) *should compute $\theta^{(0)} = (\theta_1^{(0)}, \ldots, \theta_q^{(0)}) \in [0, \frac{\pi}{2}]^q$, $\phi^{(0)} = (\phi_1^{(0)}, \ldots, \phi_{q-1}^{(0)}) \in [0, \frac{\pi}{2}]^{q-1}$, $P_1 \in U(p)$, $P_2 \in U(m-p)$, $Q_1 \in U(q)$, and $Q_2 \in U(m-q)$ such that*

$$X = \begin{bmatrix} P_1 & \\ & P_2 \end{bmatrix} \left[\begin{array}{cc|cc} B_{11}^{(0)} & B_{12}^{(0)} & 0 & 0 \\ 0 & 0 & I_{p-q} & 0 \\ \hline B_{21}^{(0)} & B_{22}^{(0)} & 0 & 0 \\ 0 & 0 & 0 & I_{m-p-q} \end{array}\right] \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}^*, \tag{2.1}$$

in which $B_{ij}^{(0)} = B_{ij}(\theta^{(0)}, \phi^{(0)})$, $i, j = 1, 2$, are bidiagonal matrices defined in (1.5).

The algorithm is inspired by the bidiagonal reduction of Golub and Kahan [8] and has already appeared in [7, 18]. It is reproduced here. Matlab-style indexing is used—$A(i, j)$ refers to the i, j entry of A; $A(i : k, j : l)$ refers to the submatrix of A in rows $i, \ldots, k$ and columns $j, \ldots, l$; $A(i : k, :)$ refers to the submatrix of A in rows $i, \ldots, k$; and so on. Also, **house**(x) constructs a

Householder reflector $F = \omega(I - \beta vv^*)$ for which the first entry of Fx is real and nonnegative and the remaining entries are zero. (This is an abuse of common terminology—F is not Hermitian if ω is not real.) If given the empty vector (), **house** returns an identity matrix. Finally, c_i, s_i, c'_i, and s'_i are shorthand for $\cos\theta_i^{(0)}$, $\sin\theta_i^{(0)}$, $\cos\phi_i^{(0)}$, and $\sin\phi_i^{(0)}$, respectively.

Algorithm 2.2 (bidiagonalize)

$Y := X$;

for $i := 1, \ldots, q$

$$u_1 := \begin{cases} Y(1:p,1) & \text{if } i = 1 \\ c'_{i-1}Y(i:p,i) + s'_{i-1}Y(i:p,q-1+i) & \text{if } i > 1; \end{cases}$$

$$u_2 := \begin{cases} -Y(p+1:m,1) & \text{if } i = 1 \\ -c'_{i-1}Y(p+i:m,i) - s'_{i-1}Y(p+i:m,q-1+i) & \text{if } i > 1; \end{cases}$$

$\theta_i^{(0)} := \mathbf{atan2}(\|u_2\|, \|u_1\|)$;

$P_1^{(i)} := \begin{bmatrix} I_{i-1} & \\ & \mathbf{house}(u_1)^* \end{bmatrix}$; $P_2^{(i)} := \begin{bmatrix} I_{i-1} & \\ & \mathbf{house}(u_2)^* \end{bmatrix}$;

$Y := \begin{bmatrix} P_1^{(i)} & \\ & P_2^{(i)} \end{bmatrix}^* Y$;

$$v_1 := \begin{cases} -s_i Y(i, i+1:q) - c_i Y(p+i, i+1:q) & \text{if } i < q \\ () & \text{if } i = q; \end{cases}$$

$v_2 := s_i Y(i, q+i:m) + c_i Y(p+i, q+i:m)$;

if $i < q$, then

$\phi_i^{(0)} := \mathbf{atan2}(\|v_1\|, \|v_2\|)$;

$Q_1^{(i)} := \begin{bmatrix} I_i & \\ & \mathbf{house}(v_1^*)^* \end{bmatrix}$;

else

$Q_1^{(q)} := I_q$;

end if

$Q_2^{(i)} := \begin{bmatrix} I_{i-1} & \\ & \mathbf{house}(v_2^*)^* \end{bmatrix}$;

$Y := Y \begin{bmatrix} Q_1^{(i)} & \\ & Q_2^{(i)} \end{bmatrix}$;

end for

$P_1 := P_1^{(1)} \cdots P_1^{(q)}$; $P_2 := P_2^{(1)} \cdots P_2^{(q)}$; $Q_1 := Q_1^{(1)} \cdots Q_1^{(q-1)}$; $Q_2 := Q_2^{(1)} \cdots Q_2^{(q)}$;

Using an LQ factorization, diagonalize the submatrix of Y lying in rows $q+1, \ldots, p$, $p+q+1, \ldots, m$ and columns $2q+1, \ldots, m$, updating Q_2 as necessary;

Theorem 2.3 *Algorithm* **bidiagonalize** *satisfies Specification* 2.1.

The proof is available in [7, 18].

We illustrate the algorithm by concentrating on the case $m=6$, $p=3$, $q=3$. In the beginning, the matrix entries can have any signs,

$$Y := X = \left[\begin{array}{ccc|ccc} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \hline \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{array}\right].$$

To introduce zeros into column 1, two Householder reflectors, based on $u_1 = Y(1{:}3, 1)$ and $u_2 = -Y(4{:}6, 1)$, respectively, are applied.

$$Y := \left[\begin{array}{c|c} \textbf{house}(u_1) & \\ \hline & \textbf{house}(u_2) \end{array}\right] Y = \left[\begin{array}{ccc|ccc} + & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ \hline - & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{array}\right].$$

In fact, because Y is unitary, $\|u_1\|^2 + \|u_2\|^2 = 1$, and $\theta_1^{(0)}$ is defined so that

$$Y = \left[\begin{array}{ccc|ccc} c_1 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ \hline -s_1 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{array}\right].$$

Next, the algorithm focuses on rows 1 and 4. Now something nonobvious happens. $Y(1, 2{:}3)$ and $Y(4, 2{:}3)$ are colinear (because $Y(:, 1)$ is orthogonal to $Y(:, 2)$ and $Y(:, 3)$), as are $Y(1, 4{:}6)$ and $Y(4, 4{:}6)$. (By *colinear*, we mean that the absolute value of the inner product equals the product of norms.) The row vectors v_1 and v_2 are defined so that v_1 is colinear with $Y(1, 2{:}3)$ and $Y(4, 2{:}3)$ and v_2 is colinear with $Y(1, 4{:}6)$ and $Y(4, 4{:}6)$. Computing $\phi_1^{(0)}$ and multiplying by Householder reflectors gives

$$Y := Y \left[\begin{array}{c|c} 1 & \\ \hline & \begin{array}{c|c} \textbf{house}(v_1^*) & \\ \hline & \textbf{house}(v_2^*) \end{array} \end{array}\right] = \left[\begin{array}{ccc|ccc} c_1 & -s_1 s_1' & 0 & s_1 c_1' & 0 & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ \hline -s_1 & -c_1 s_1' & 0 & c_1 c_1' & 0 & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \end{array}\right].$$

The algorithm proceeds in a similar fashion. Now, $Y(2{:}3, 2)$ and $Y(2{:}3, 4)$ are colinear, as are $Y(5{:}6, 2)$ and $Y(5{:}6, 4)$. By computing $\theta_2^{(0)}$ and applying Householder reflectors, we obtain

$$Y := \left[\begin{array}{ccc|ccc} c_1 & -s_1 s_1' & 0 & s_1 c_1' & 0 & 0 \\ 0 & c_2 c_1' & \times & c_2 s_1' & \times & \times \\ 0 & 0 & \times & 0 & \times & \times \\ \hline -s_1 & -c_1 s_1' & 0 & c_1 c_1' & 0 & 0 \\ 0 & -s_2 c_1' & \times & -s_2 s_1' & \times & \times \\ 0 & 0 & \times & 0 & \times & \times \end{array}\right],$$

then another pair of Householder reflectors gives

$$Y := \left[\begin{array}{ccc|ccc} c_1 & -s_1 s_1' & 0 & s_1 c_1' & 0 & 0 \\ 0 & c_2 c_1' & -s_2 s_2' & c_2 s_1' & s_2 c_2' & 0 \\ 0 & 0 & \times & 0 & \times & \times \\ \hline -s_1 & -c_1 s_1' & 0 & c_1 c_1' & 0 & 0 \\ 0 & -s_2 c_1' & -c_2 s_2' & -s_2 s_1' & c_2 c_2' & 0 \\ 0 & 0 & \times & 0 & \times & \times \end{array}\right],$$

and so on. Note that the final matrix is represented implicitly by $\theta_1^{(0)}, \ldots, \theta_q^{(0)}$ and $\phi_1^{(0)}, \ldots, \phi_{q-1}^{(0)}$, so that it is exactly orthogonal, even on a floating-point architecture.

The following theorem will be useful later. A *diagonal signature matrix* is a diagonal matrix whose diagonal entries are ± 1.

Theorem 2.4 *If B_{11} and B_{21} are upper bidiagonal, B_{12} and B_{22} are lower bidiagonal, and*

$$\left[\begin{array}{cc} B_{11} & B_{12} \\ B_{21} & B_{22} \end{array}\right]$$

is real orthogonal (but not necessarily having the sign pattern required for bidiagonal block form), then there exist diagonal signature matrices D_1, D_2, E_1, E_2 such that

$$\left[\begin{array}{cc} D_1 & \\ & D_2 \end{array}\right] \left[\begin{array}{cc} B_{11} & B_{12} \\ B_{21} & B_{22} \end{array}\right] \left[\begin{array}{cc} E_1 & \\ & E_2 \end{array}\right]$$

is a real orthogonal matrix in bidiagonal block form.

Proof Run **bidiagonalize**. P_1, P_2, Q_1, and Q_2 are products of Householder reflectors that are in fact diagonal signature matrices because the input matrix already has the correct zero/nonzero pattern. Let $D_1 = P_1$, $D_2 = P_2$, $E_1 = Q_1$, and $E_2 = Q_2$. □

3 Reviewing and extending the SVD step

In [8, 9], Golub, Kahan, and Reinsch developed an SVD method based on implicit QR iteration. The bulge-chasing method, as modified by Demmel and Kahan [6], is implemented as one of the most heavily used SVD routines in LAPACK. Phase II of our CSD algorithm is based on this SVD method.

Given a real upper bidiagonal matrix B and a shift $\sigma \geq 0$, the SVD step of Golub and Reinsch applies a unitary equivalence $\bar{B} = S^T B T$ derived from QR iteration on $B^T B - \sigma^2 I$. The step is designed so that iteration will drive a superdiagonal entry of $\bar{B}$ to zero (especially quickly if the shifts lie near singular values of B).

This section reviews the SVD step of Golub and Reinsch. There are two notable aspects not present in most descriptions.

- First, a certain left-right symmetry is emphasized—not only is the SVD step equivalent to a QR step on $B^T B - \sigma^2 I$; it is also equivalent to a QR step on $BB^T - \sigma^2 I$.
- Second, the SVD step is extended to handle any number of zeros on the bidiagonal band of B. (The original SVD step of Golub and Reinsch requires special handling as soon as a zero appears. The modification by Demmel and Kahan relaxes this, but only for zeros on the main diagonal and only when the shift is $\sigma = 0$.) The modification is necessary in Section 4.

3.1 QR step

The SVD iterations of interest are derived from QR iteration for symmetric tridiagonal matrices.

Given a real symmetric tridiagonal matrix A and a shift $\lambda \in \mathbb{R}$, a single QR step is accomplished by computing a QR factorization

$$A - \lambda I = QR, \tag{3.1}$$

then reversing the factors and putting λI back,

$$\bar{A} := RQ + \lambda I. \tag{3.2}$$

Note that the resulting $\bar{A}$ is symmetric tridiagonal, because $RQ = Q^T(A - \lambda I)Q$ is upper Hessenberg and symmetric.

Note that if A is *unreduced*, i.e., its subdiagonal entries are all nonzero, then Q and R are unique up to signs. (Specifically, every QR factorization is of the form $A - \lambda I = (QD)(DR)$ for some diagonal signature matrix D.) However, if A has zero entries on its subdiagonal, then making the QR factorization unique requires extra care. The following definition introduces a "preferred" QR factorization. There are two important points: (1) the existence of the forthcoming CSD step relies on the uniqueness of the preferred QR factorization, and (2) the handling of noninvertible A supports the CSD deflation procedure. Below, the notation $A \oplus B$ refers to the block diagonal matrix $\left[\begin{smallmatrix} A & \\ & B \end{smallmatrix}\right]$.

Definition 3.1 Let A be an m-by-m real symmetric tridiagonal matrix. Express A as

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_r \end{bmatrix},$$

in which each A_i is either an unreduced tridiagonal matrix or 1-by-1. A *preferred QR factorization* for A is a QR factorization with a special form. There are two cases.

Case 1: *A is invertible.* Then a preferred QR factorization has the form

$$A = \begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_r \end{bmatrix} \begin{bmatrix} R_1 & & & \\ & R_2 & & \\ & & \ddots & \\ & & & R_r \end{bmatrix}$$

and satisfies the following conditions:

1. every Q_i has the same number of rows and columns as A_i, is real orthogonal and upper Hessenberg, and has positive subdiagonal and determinant one (unless it is one-by-one, in which case it equals the scalar 1), and
2. every R_i has the same number of rows and columns as A_i and is upper triangular.

Case 2: *A is not invertible.* Define $Q_1, \ldots, Q_r$ and $R_1, \ldots, R_r$ as in the first case, let k be the least index identifying a noninvertible A_k, and let l be the index of this block's last row and column in the overall matrix A. (Note that the first zero diagonal entry of $R_1 \oplus \cdots \oplus R_r$ must be in position (l, l).) Also, let

$$P = \begin{bmatrix} I_{l-1} & 0 & 0 \\ 0 & 0 & \pm 1 \\ 0 & I_{m-l} & 0 \end{bmatrix},$$

with the sign chosen so that $\det(P) = 1$. A preferred QR factorization for A is $A = QR$ with $Q = (Q_1 \oplus \cdots \oplus Q_k \oplus I_{m-l})P$ and $R = P^T(R_1 \oplus \cdots \oplus R_k \oplus A_{k+1} \oplus \cdots \oplus A_r)$.

Theorem 3.2 *The terminology is valid: every "preferred QR factorization" really is a QR factorization.*

Theorem 3.3 *If A is a real symmetric tridiagonal matrix, then a preferred QR factorization $A = QR$ exists and is unique.*

The proofs are straightforward.

Theorem 3.4 *Apply the QR step* (3.1–3.2) *to a real symmetric tridiagonal matrix A using the preferred QR factorization. If the shift λ is an eigenvalue of A, then deflation occurs immediately: the resulting $\bar{A}$ has the form*

$$\bar{A} = \left[\begin{array}{c|c} * & \\ \hline & \lambda \end{array}\right].$$

Proof $A - \lambda I$ is not invertible, so its preferred QR factorization satisfies Case 2 of Definition 3.1. The rest of the proof uses the notation of that definition. The last row of R_k contains all zeros, so row l of $R_1 \oplus \cdots \oplus R_k \oplus (A_{k+1} - \lambda I) \oplus \cdots \oplus (A_r - \lambda I)$ contains all zeros. The permutation matrix P is designed to slide this row of all zeros to the bottom of R, and hence the last row of RQ also contains all zeros. By symmetry, the last column of RQ also contains only zeros, and so RQ is block diagonal with a one-by-one block equaling zero in the bottom-right corner. Adding λI to produce $\bar{A} = RQ + \lambda I$ makes the bottom-right entry equal λ. □

The orthogonal factor Q in a preferred QR factorization can be expressed as a product of Givens rotations in a particularly simple way. A 2-by-2 *Givens rotation* with an angle of θ is a matrix $\left[\begin{smallmatrix} c & -s \\ s & c \end{smallmatrix}\right]$ in which $c = \cos\theta$ and $s = \sin\theta$. More generally, an m-by-m *Givens rotation* is an m-by-m real orthogonal matrix with a 2-by-2 principal submatrix that is a Givens rotation and whose principal submatrix lying in the other rows and columns is the $(m-2)$-by-$(m-2)$ identity matrix.

Theorem 3.5 *The orthogonal factor Q in a preferred QR factorization of an m-by-m real symmetric tridiagonal matrix can be expressed uniquely as a product of Givens rotations $G_1 \cdots G_{m-1}$ in which G_j rotates columns j and $j+1$ through an angle in $[0, \pi)$.*

Proof Existence is proved by Algorithm 3.7 below. Uniqueness is guaranteed by the upper Hessenberg structure: inductively, the $(j+1, j)$ entry of Q determines G_j. □

Algorithm 3.6 (givens) This algorithm computes a Givens rotation, with a corner case defined in a particularly important way.

1. Given a nonzero 2-by-1 vector x, **givens**(m, i_1, i_2, x) constructs an m-by-m Givens rotation whose submatrix lying in rows and columns i_1 and i_2 is a 2-by-2 Givens rotation $G = \left[\begin{smallmatrix} c & -s \\ s & c \end{smallmatrix}\right]$ with angle in $[0, \pi)$ such that $G^T x = (\pm\|x\|, 0)^T$.
2. **givens**$(m, i_1, i_2, (0,0)^T)$ is defined to equal **givens**$(m, i_1, i_2, (0,1)^T)$—it rotates through an angle of $\frac{\pi}{2}$.

The choice to rotate through $\frac{\pi}{2}$ when $x = (0, 0)^T$ allows the following algorithm to handle Cases 1 and 2 of the preferred QR factorization in a uniform way.

Algorithm 3.7 (**qr_step**) Given an m-by-m real symmetric tridiagonal matrix A and a shift $\lambda \in \mathbb{R}$, the following algorithm performs one QR step. See Theorem 3.8. It is an extension of the idea on pp. 418–420 of [10].

$\bar{A} := A - \lambda I$;
for $i = 1, \ldots, m-1$
  $v = \begin{cases} \bar{A}(i : i+1, i) & \text{if } i = 1 \text{ or } \bar{A}(i : i+1, i-1) \text{ is the zero vector} \\ \bar{A}(i : i+1, i-1) & \text{otherwise;} \end{cases}$
  $G_i := \mathbf{givens}(m, i, i+1, v)$;
  $\bar{A} := G_i^T \bar{A} G_i$;
end for
$\bar{A} := \bar{A} + \lambda I$;

Theorem 3.8 *Run Algorithm* 3.7 *to compute* $G_1, \ldots, G_{m-1}$ *and* $\bar{A}$*, and define* $Q = G_1 \cdots G_{m-1}$ *and* $R = Q^T(A - \lambda I)$*. Then* $A - \lambda I = QR$ *is a preferred QR factorization and* (3.1) *and* (3.2) *are satisfied.*

Proof The proof is broken into two cases.

Case 1: λ *is not an eigenvalue of* A. The proof is by induction on r, the number of unreduced blocks of A.
The base case is $r = 1$. In this case, the algorithm executes the usual bulge-chasing QR step. See [10].
For the inductive step, suppose $r > 1$ and assume that the theorem has been proved for smaller values of r. Let s be the number of rows and columns in the final block A_r. The first $m - s - 1$ executions of the loop neither observe nor modify the final s rows or columns of $\bar{A}$ and by induction compute $Q_1, \ldots, Q_{r-1}$ and $R_1, \ldots, R_{r-1}$ of the preferred QR factorization. At the beginning of the loop with $i = m - s$, the $(m - s, m - s - 1)$-entry of $\bar{A}$ is nonzero because $A - \lambda I$ is invertible, and the $(m - s + 1, m - s - 1)$-entry of $\bar{A}$ is zero, so G_{m-r} is set to an identity matrix. When the loop continues with $i = m - s + 1$, the algorithm proceeds as in [10].

Case 2: λ *is an eigenvalue of* A. Let l be the index of the first column of A that is a linear combination of the earlier columns. The algorithm proceeds as in Case 1 until the loop with $i = l$. At that point, the leading $(l-1)$-by-$(l-1)$ principal submatrix of the new tridiagonal matrix is fixed (ignoring the addition of λI at the very end), and considering the proof of Theorem 3.4, the lth row of $\bar{A}$ is all zeros. Therefore, G_l is a rotation by $\frac{\pi}{2}$, which has the effect of swapping

rows l and $l+1$ of $\bar{A}$. Now, the $(l+1)$st row of $\bar{A}$ is all zeros, so by induction, all remaining Givens rotations have angle $\frac{\pi}{2}$ and the row of zeros is pushed to the bottom of $\bar{A}$. This constructs Q and R as in Case 2 of Definition 3.1. □

3.2 SVD step

To compute the SVD of a bidiagonal matrix B, we start with the idea of applying QR iteration to $B^TB - \sigma^2 I$. Because the formation of B^TB is problematic in floating-point, each QR step must be executed implicitly, working directly with the entries of B. The following definition of the SVD step is unconventional but equivalent to the usual definition.

Definition 3.9 Let B be a real bidiagonal matrix (either upper bidiagonal or lower bidiagonal) and σ a nonnegative real number. Matrices $\bar{B}$, S, and T are obtained from an *SVD step* if

$$BB^T - \sigma^2 I = SL^T \quad \text{and} \quad B^TB - \sigma^2 I = TR$$

are preferred QR factorizations and

$$\bar{B} = S^TBT.$$

Note that

$$\bar{B}\bar{B}^T = L^TS + \sigma^2 I \quad \text{and} \quad \bar{B}^T\bar{B} = RT + \sigma^2 I,$$

i.e., an SVD step on B implicitly computes QR steps for BB^T and B^TB.

Theorem 3.10 *The SVD step exists and is uniquely defined.*

Proof This follows immediately from the existence and uniqueness of the preferred QR factorization. □

Theorem 3.11 *If B is upper bidiagonal, then $\bar{B}$ is upper bidiagonal. If B is lower bidiagonal, then $\bar{B}$ is lower bidiagonal.*

The proof is presented after Lemma 3.17 below.

Before going any further with the SVD step, we need a utility routine.

Algorithm 3.12 (**bulge_start**) Given a 2-by-1 vector x and a shift $\sigma \geq 0$, **bulge_start** computes a vector colinear with $(x_1^2 - \sigma^2, x_1x_2)^T$. If $(x_1^2 - \sigma^2, x_1x_2)^T$ is the zero vector, then **bulge_start** returns the zero vector.

The implementation of **bulge_start** is omitted. LAPACK's DBDSQR provides guidance [1].

The following algorithm computes an SVD step for an upper bidiagonal matrix. It can also handle lower bidiagonal matrices by taking transposes as appropriate.

Algorithm 3.13 (**svd_step**) Given an m-by-m upper bidiagonal matrix B and a shift $\sigma \geq 0$, the following algorithm computes one SVD step. See Theorem 3.14.

1 $\bar{B} := B$;

3 for $i := 1, \ldots, m-1$

5 $\quad v := \begin{cases} \textbf{bulge\_start}(\bar{B}(i, i:i+1)^T, \sigma) & \text{if } i=1 \text{ or } \bar{B}(i-1, i:i+1) = (0,0) \\ \bar{B}(i-1, i:i+1)^T & \text{otherwise;} \end{cases}$

6 $\quad T_i := \textbf{givens}(m, i, i+1, v)$;

7 $\quad \bar{B} := \bar{B} T_i$;

9 $\quad u := \begin{cases} \textbf{bulge\_start}(\bar{B}(i:i+1, i+1), \sigma) & \text{if } \bar{B}(i:i+1, i) = (0,0)^T \\ \bar{B}(i:i+1, i) & \text{otherwise;} \end{cases}$

10 $\quad S_i := \textbf{givens}(m, i, i+1, u)$;

11 $\quad \bar{B} := S_i^T \bar{B}$;

13 end for

15 $S := S_1 \cdots S_{m-1}; \quad T := T_1 \cdots T_{m-1}$;

Theorem 3.14 *Given a real upper bidiagonal matrix B and a shift $\sigma \geq 0$,*

1. *Algorithm* **3.13** *computes an SVD step.*
2. *$\bar{B}$ is real upper bidiagonal.*

The proof follows immediately from Theorem 3.8 and the following three lemmas.

Lemma 3.15 *Upon completion of line 7, $\bar{B}$ is upper bidiagonal, with the possible exception of a "bulge" at $(i+1, i)$. Upon completion of line 11, $\bar{B}$ is upper bidiagonal, with the possible exception of a "bulge" at $(i, i+2)$ if $i < m-1$. Upon completion of the entire algorithm, $\bar{B}$ is upper bidiagonal.*

The proof is omitted; the bulge-chasing nature of the algorithm is standard.

Lemma 3.16 *Let B be an m-by-m real upper bidiagonal matrix and σ a nonnegative real number. Run Algorithm* 3.13 *to produce $T_1, \ldots, T_{m-1}$, and run Algorithm* 3.7 *with $A = B^T B$ and $\lambda = \sigma^2$ to produce $G_1, \ldots, G_{m-1}$. Then $T_i = G_i$ for $i = 1, \ldots, m-1$.*

The proof can be found in the Appendix.

Lemma 3.17 *Let B be an m-by-m real upper bidiagonal matrix and σ a nonnegative real number. Run Algorithm* 3.13 *to produce $S_1, \ldots, S_{m-1}$, and run Algorithm* 3.7 *with $A = BB^T$ and $\lambda = \sigma^2$ to produce $G_1, \ldots, G_{m-1}$. Then $S_i = G_i$ for $i = 1, \ldots, m-1$.*

The proof can be found in the Appendix.

Proof of Theorem 3.11 If B is upper bidiagonal, then $\bar{B}$ can be obtained from Algorithm 3.13, which produces upper bidiagonal matrices. If B is lower bidiagonal, then apply the same argument to B^T. □

4 Phase II: Algorithm csd_step

Now we can return to the CSD algorithm. Phase I, which was already seen, transforms the original partitioned unitary matrix to bidiagonal block form. Phase II, which is developed now, iteratively applies the SVD step to each of the four blocks of this matrix. Algorithm **csd_step** executes a single step in the iteration.

4.1 Existence of the CSD step

Definition 4.1 Let $\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$ be a real orthogonal matrix in bidiagonal block form and μ and ν nonnegative shifts satisfying $\mu^2 + \nu^2 = 1$. The matrix equation

$$\begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix} = \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} \begin{bmatrix} S_1 & \\ & S_2 \end{bmatrix}^T \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} T_1 & \\ & T_2 \end{bmatrix} \begin{bmatrix} E_1 & \\ & E_2 \end{bmatrix}$$

effects a *CSD step* if

1. $B_{11} \mapsto S_1^T B_{11} T_1$ and $B_{22} \mapsto S_2^T B_{22} T_2$ are SVD steps with shift μ,
2. $B_{12} \mapsto S_1^T B_{12} T_2$ and $B_{21} \mapsto S_2^T B_{21} T_1$ are SVD steps with shift ν,
3. D_1, D_2, E_1, and E_2 are diagonal signature matrices, and
4. $\begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix}$ is a matrix in bidiagonal block form.

The existence of the CSD step is not obvious at first glance. It depends on several SVD steps being related in very specific ways, e.g., B_{11} and B_{21} having the same right orthogonal factor T_1. The following theorem establishes that yes, indeed, the CSD step exists, and its proof makes clear the necessity of the restriction $\mu^2 + \nu^2 = 1$.

Theorem 4.2 *The CSD step exists and the result*

$$\begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix}$$

is uniquely defined.

Proof Begin with the identities

$$B_{11}^T B_{11} - \mu^2 I = -(B_{21}^T B_{21} - \nu^2 I), \tag{4.1}$$

$$B_{22}^T B_{22} - \mu^2 I = -(B_{12}^T B_{12} - \nu^2 I), \tag{4.2}$$

$$B_{11} B_{11}^T - \mu^2 I = -(B_{12} B_{12}^T - \nu^2 I), \tag{4.3}$$

$$B_{22} B_{22}^T - \mu^2 I = -(B_{21} B_{21}^T - \nu^2 I). \tag{4.4}$$

Each is proved using orthogonality and the relation $\mu^2 + \nu^2 = 1$. For example, the orthogonality of $\left[\begin{smallmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{smallmatrix}\right]$ implies $B_{11}^T B_{11} + B_{21}^T B_{21} = I$, and splitting the right-hand-side I into $\mu^2 I + \nu^2 I$ and rearranging gives (4.1).

Define $\bar{B}_{ij} = S_{ij} B_{ij} T_{ij}$ by an SVD step with the appropriate shift (μ if $i = j$ or ν if $i \neq j$). This produces a total of eight orthogonal factors S_{ij}, T_{ij}, but only four are required for the CSD step. In fact, by uniqueness of the preferred QR factorization, $T_{11} = T_{21}$, $T_{12} = T_{22}$, $S_{11} = S_{12}$, and $S_{21} = S_{22}$. (For example, T_{11} is the orthogonal factor in the preferred QR factorization $B_{11}^T B_{11} - \mu^2 I = T_{11} R_{11}$, and T_{21} is the orthogonal factor in the preferred QR factorization $B_{21}^T B_{21} - \nu^2 I = T_{21} R_{21}$. Considering (4.1) and the uniqueness of the preferred QR factorization, we must have $T_{21} = T_{11}$ and $R_{21} = -R_{11}$.) Hence, it is legal to define $T_1 = T_{11} = T_{21}$, $T_2 = T_{12} = T_{22}$, $S_1 = S_{11} = S_{12}$, and $S_2 = S_{21} = S_{22}$.

Finally, D_1, D_2, E_1, and E_2 are designed to fix the sign pattern required for a matrix in bidiagonal block form. Their existence is guaranteed by Theorem 2.4.

Regarding uniqueness, S_1, S_2, T_1, and T_2 are uniquely defined by the preferred QR factorization, so $S_i^T B_{ij} T_j$, i, $j = 1, 2$, are uniquely defined. This uniquely defines the absolute values of the entries of $\bar{B}_{ij}$, i, $j = 1, 2$, and the signs are specified by the definition of bidiagonal block form. □

The obvious way to compute a CSD step is to compute four SVD steps and then to combine them together as in the proof. Of course, when working in floating-point, the identities such as $T_{11} = T_{21}$ typically will not hold exactly. In fact, when singular values are clustered, the computed T_{11} and T_{21} may not even bear a close resemblance.

Our approach is to interleave the computation of the four SVD steps, taking care to compute S_1, S_2, T_1, and T_2 once and only once. We find that when one block of a matrix provides unreliable information (specifically, a very short vector whose direction is required for a Givens rotation), another block may come to the rescue, providing more reliable information. Hence, the

redundancy in the partitioned orthogonal matrix, rather than being a stumbling block, is actually an aid to stability.

4.2 Algorithm specification

The following is a specification for Algorithm **csd_step**, which accomplishes one step in Phase II of the CSD algorithm.

Specification 4.3 *The input to* **csd_step** *should consist of*

1. $\theta^{(n)} \in [0, \frac{\pi}{2}]^q$ *and* $\phi^{(n)} \in [0, \frac{\pi}{2}]^{q-1}$, *implicitly defining a matrix in bidiagonal block form* $\left[\begin{smallmatrix} B_{11}^{(n)} & B_{12}^{(n)} \\ B_{21}^{(n)} & B_{22}^{(n)} \end{smallmatrix}\right]$,
2. *integers* $1 \le \underline{i} < \bar{i} \le q$ *identifying the current block in a partially deflated matrix—*$\left[\begin{smallmatrix} B_{11}^{(n)} & B_{12}^{(n)} \\ B_{21}^{(n)} & B_{22}^{(n)} \end{smallmatrix}\right]$ *must have the form*

$$\left[\begin{array}{cc|cc} * & 0 & * & \\ & B_{11}^{(n)}(\underline{i}:\bar{i},\underline{i}:\bar{i}) & 0 & B_{12}^{(n)}(\underline{i}:\bar{i},\underline{i}:\bar{i}) \\ & * & & 0\ * \\ \hline * & 0 & * & \\ & B_{21}^{(n)}(\underline{i}:\bar{i},\underline{i}:\bar{i}) & 0 & B_{22}^{(n)}(\underline{i}:\bar{i},\underline{i}:\bar{i}) \\ & * & & 0\ * \end{array}\right], \tag{4.5}$$

3. *shifts* $0 \le \mu, \nu \le 1$ *satisfying* $\mu^2 + \nu^2 = 1$.

The algorithm should compute one CSD step with shifts μ *and* ν *to effect*

$$\left[\begin{array}{c|c} B_{11}^{(n)}(\underline{i}:\bar{i},\underline{i}:\bar{i}) & B_{12}^{(n)}(\underline{i}:\bar{i},\underline{i}:\bar{i}) \\ \hline B_{21}^{(n)}(\underline{i}:\bar{i},\underline{i}:\bar{i}) & B_{22}^{(n)}(\underline{i}:\bar{i},\underline{i}:\bar{i}) \end{array}\right] \mapsto \left[\begin{array}{c|c} \bar{B}_{11} & \bar{B}_{12} \\ \hline \bar{B}_{21} & \bar{B}_{22} \end{array}\right]. \tag{4.6}$$

The output should consist of $\theta^{(n+1)} \in [0, \frac{\pi}{2}]^q$ *and* $\phi^{(n+1)} \in [0, \frac{\pi}{2}]^{q-1}$, *implicitly defining the matrix* $\left[\begin{smallmatrix} B_{11}^{(n+1)} & B_{12}^{(n+1)} \\ B_{21}^{(n+1)} & B_{22}^{(n+1)} \end{smallmatrix}\right]$ *that results from replacing the appropriate submatrices from* (4.5) *with the corresponding submatrices of* (4.6), *and orthogonal matrices* $U_1^{(n+1)}$, $U_2^{(n+1)}$, $V_1^{(n+1)}$, $V_2^{(n+1)}$ *such that*

$$\begin{bmatrix} B_{11}^{(n+1)} & B_{12}^{(n+1)} \\ B_{21}^{(n+1)} & B_{22}^{(n+1)} \end{bmatrix} = \begin{bmatrix} U_1^{(n+1)} & \\ & U_2^{(n+1)} \end{bmatrix}^* \begin{bmatrix} B_{11}^{(n)} & B_{12}^{(n)} \\ B_{21}^{(n)} & B_{22}^{(n)} \end{bmatrix} \begin{bmatrix} V_1^{(n+1)} & \\ & V_2^{(n+1)} \end{bmatrix}. \tag{4.7}$$

4.3 The algorithm

The naive idea for implementing the above specification is to compute four SVD steps separately. As mentioned in the proof of Theorem 4.2, this produces eight orthogonal factors when only four are needed. In fact, in light of (4.1–4.4) and Theorem 3.5, the Givens rotations defining S_1, S_2, T_1, and T_2 come in identical pairs. Of course, in floating-point, the paired Givens rotations

Algorithm 4.4 (csd_step) This algorithm performs one CSD step on a matrix in bidiagonal block form and satisfies Specification 4.3. Its correctness is established in Theorem 4.5.

Explicitly construct $\left[\begin{smallmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{smallmatrix}\right]$ from $(\theta^{(n)}, \phi^{(n)})$;

Initialize $\left[\begin{smallmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{smallmatrix}\right] := \left[\begin{smallmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{smallmatrix}\right]$;

$v_{11} := \textbf{bulge\_start}(\bar{B}_{11}(\underline{i}, \underline{i} : \underline{i}+1)^T, \mu)$;
$v_{21} := \textbf{bulge\_start}(\bar{B}_{21}(\underline{i}, \underline{i} : \underline{i}+1)^T, \nu)$;
$v_1 := \textbf{merge}(v_{11}, v_{21})$;
$T_{1,\underline{i}} := \textbf{givens}\left(q, \underline{i}, \underline{i}+1, v_1\right)$;
$\left[\begin{smallmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{smallmatrix}\right] := \left[\begin{smallmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{smallmatrix}\right] \left[\begin{smallmatrix} T_{1,\underline{i}} & \\ & I \end{smallmatrix}\right]$;

$$u_{11} := \begin{cases} \bar{B}_{11}(\underline{i} : \underline{i}+1, \underline{i}) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{11}(\underline{i} : \underline{i}+1, \underline{i}+1), \mu) & \text{otherwise;} \end{cases}$$

$u_{12} := \textbf{bulge\_start}(\bar{B}_{12}(\underline{i} : \underline{i}+1, \underline{i}), \nu)$;

$$u_{21} := \begin{cases} \bar{B}_{21}(\underline{i} : \underline{i}+1, \underline{i}) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{21}(\underline{i} : \underline{i}+1, \underline{i}+1), \nu) & \text{otherwise;} \end{cases}$$

$u_{22} := \textbf{bulge\_start}(\bar{B}_{22}(\underline{i} : \underline{i}+1, \underline{i}), \mu)$;
$u_1 := \textbf{merge}(u_{11}, u_{12})$; $u_2 := \textbf{merge}(u_{21}, u_{22})$;
$S_{1,\underline{i}} := \textbf{givens}(q, \underline{i}, \underline{i}+1, u_1)$; $S_{2,\underline{i}} := \textbf{givens}(q, \underline{i}, \underline{i}+1, u_2)$;
$\left[\begin{smallmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{smallmatrix}\right] := \left[\begin{smallmatrix} S_{1,\underline{i}} & \\ & S_{2,\underline{i}} \end{smallmatrix}\right]^T \left[\begin{smallmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{smallmatrix}\right]$;

for $i := \underline{i}+1, \ldots, \bar{i}-1$

$$v_{11} := \begin{cases} \bar{B}_{11}(i-1, i : i+1) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{11}(i, i : i+1), \mu) & \text{otherwise;} \end{cases}$$

$$v_{21} := \begin{cases} \bar{B}_{21}(i-1, i : i+1) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{21}(i, i : i+1), \nu) & \text{otherwise;} \end{cases}$$

$$v_{12} := \begin{cases} \bar{B}_{12}(i-1, i-1 : i) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{12}(i, i-1 : i), \nu) & \text{otherwise;} \end{cases}$$

$$v_{22} := \begin{cases} \bar{B}_{22}(i-1, i-1 : i) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{22}(i, i-1 : i), \mu) & \text{otherwise;} \end{cases}$$

$v_1 := \textbf{merge}(v_{11}, v_{21})$; $v_2 := \textbf{merge}(v_{12}, v_{22})$;
$T_{1,i} := \textbf{givens}(q, i, i+1, v_1)$; $T_{2,i-1} := \textbf{givens}(q, i-1, i, v_2)$;
$\left[\begin{smallmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{smallmatrix}\right] := \left[\begin{smallmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{smallmatrix}\right] \left[\begin{smallmatrix} T_{1,i} & \\ & T_{2,i-1} \end{smallmatrix}\right]$;

would not actually be identical. Hence, the naive algorithm suffers from the standpoints of efficiency (each Givens rotation is needlessly computed twice) and stability (what happens when two computed Givens rotations disagree?).

Algorithm 4.4 (continued)

$$u_{11} := \begin{cases} \bar{B}_{11}(i : i+1, i) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{11}(i : i+1, i+1), \mu) & \text{otherwise;} \end{cases}$$

$$u_{12} := \begin{cases} \bar{B}_{12}(i : i+1, i-1) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{12}(i : i+1, i), \nu) & \text{otherwise;} \end{cases}$$

$$u_{21} := \begin{cases} \bar{B}_{21}(i : i+1, i) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{21}(i : i+1, i+1), \nu) & \text{otherwise;} \end{cases}$$

$$u_{22} := \begin{cases} \bar{B}_{22}(i : i+1, i-1) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{22}(i : i+1, i), \mu) & \text{otherwise;} \end{cases}$$

$u_1 := \textbf{merge}(u_{11}, u_{12})$; $u_2 := \textbf{merge}(u_{21}, u_{22})$;

$S_{1,i} := \textbf{givens}(q, i, i+1, u_1)$; $S_{2,i} := \textbf{givens}(q, i, i+1, u_2)$;

$\begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix} := \begin{bmatrix} S_{1,i} & \\ & S_{2,i} \end{bmatrix}^T \begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix}$;

end for

$$v_{12} := \begin{cases} \bar{B}_{12}(\bar{i}-1, \bar{i}-1 : \bar{i}) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{12}(\bar{i}, \bar{i}-1 : \bar{i}), \nu) & \text{otherwise;} \end{cases}$$

$$v_{22} := \begin{cases} \bar{B}_{22}(\bar{i}-1, \bar{i}-1 : \bar{i}) & \text{if nonzero} \\ \textbf{bulge\_start}(\bar{B}_{22}(\bar{i}, \bar{i}-1 : \bar{i}), \mu) & \text{otherwise;} \end{cases}$$

$v_2 := \textbf{merge}(v_{12}, v_{22})$;

$T_{2,\bar{i}-1} := \textbf{givens}(q, \bar{i}-1, \bar{i}, v_2)$;

$\begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix} := \begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix} \begin{bmatrix} I & \\ & T_{2,\bar{i}-1} \end{bmatrix}$;

$U_1^{(n+1)} := S_{1,\underline{i}} \cdots S_{1,\bar{i}-1}$; $U_2^{(n+1)} := S_{2,\underline{i}} \cdots S_{2,\bar{i}-1}$;

$V_1^{(n+1)} := T_{1,\underline{i}} \cdots T_{1,\bar{i}-1}$; $V_2^{(n+1)} := T_{2,\underline{i}} \cdots T_{2,\bar{i}-1}$;

Fix signs in $\begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix}$ to match (1.5), negating columns of $U_1^{(n+1)}$, $U_2^{(n+1)}$, $V_1^{(n+1)}$, and $V_2^{(n+1)}$ as necessary;

Compute $(\theta^{(n+1)}, \phi^{(n+1)})$ to implicitly represent $\begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix}$;

Our solution, suggested by Fig. 2, is to execute the four SVD steps simultaneously, computing each Givens rotation once and only once through a bulge-chasing procedure that treats all four blocks with equal regard.

Algorithm 4.4 (csd_step) executes a CSD step.

The algorithm makes use of a routine called **merge**. In the absence of roundoff error, **merge** is essentially a no-op; given two vectors in any one-dimensional subspace, it returns a vector in the same subspace. (And

merge$((0,0)^T,(0,0)^T)=(0,0)^T$.) In the presence of roundoff error, **merge** is used to ameliorate small differences resulting from previous roundoff errors.

Note that it is possible and preferable to enforce signs and compute $\theta^{(n+1)}$ and $\phi^{(n+1)}$ as Givens rotations are applied, instead of waiting until the end of the algorithm.

Theorem 4.5 *If arithmetic operations are performed exactly, then Algorithm* **csd_step** *satisfies Specification* 4.3.

Proof The proof is organized into three parts. First, it is shown that the algorithm computes SVD steps for the top-left and bottom-right blocks. Then, it is shown that the algorithm computes SVD steps for the top-right and bottom-left blocks. Finally, utilizing the proof of Theorem 4.2, it is shown that the algorithm as stated simultaneously computes SVD steps of all four blocks and hence a CSD step for the entire matrix.

For the first part, temporarily delete lines 6, 11, 12, 21, 22, 29, 30, 38, 47, and 49 and make the following replacements:

Line	Original	Replacement
7	$v_1 := \mathbf{merge}(v_{11}, v_{21});$	$v_1 := v_{11};$
14	$u_1 := \mathbf{merge}(u_{11}, u_{12}); u_2 := \mathbf{merge}(u_{21}, u_{22});$	$u_1 := u_{11}; u_2 := u_{22};$
24	$v_1 := \mathbf{merge}(v_{11}, v_{21}); v_2 := \mathbf{merge}(v_{12}, v_{22});$	$v_1 := v_{11}; v_2 := v_{22};$
32	$u_1 := \mathbf{merge}(u_{11}, u_{12}); u_2 := \mathbf{merge}(u_{21}, u_{22});$	$u_1 := u_{11}; u_2 := u_{22};$
40	$v_2 := \mathbf{merge}(v_{12}, v_{22});$	$v_2 := v_{22};$

All references to the top-right and bottom-left blocks have been removed, and the resulting algorithm is equivalent to running Algorithm **svd_step** on the top-left and bottom-right blocks. Hence, $U_1^{(n+1)}$ and $V_1^{(n+1)}$ are the outer factors from an SVD step on the top-left block, and $U_2^{(n+1)}$ and $V_2^{(n+1)}$ are the outer factors from an SVD step on the bottom-right block.

For the second part, revert back to the original algorithm but make the following changes, intended to focus attention on the top-right and bottom-left blocks: Delete lines 5, 10, 13, 20, 23, 28, 31, 39, 47, and 49 and make the following replacements:

Line	Original	Replacement
7	$v_1 := \mathbf{merge}(v_{11}, v_{21});$	$v_1 := v_{21};$
14	$u_1 := \mathbf{merge}(u_{11}, u_{12}); u_2 := \mathbf{merge}(u_{21}, u_{22});$	$u_1 := u_{12}; u_2 := u_{21};$
24	$v_1 := \mathbf{merge}(v_{11}, v_{21}); v_2 := \mathbf{merge}(v_{12}, v_{22});$	$v_1 := v_{21}; v_2 := v_{12};$
32	$u_1 := \mathbf{merge}(u_{11}, u_{12}); u_2 := \mathbf{merge}(u_{21}, u_{22});$	$u_1 := u_{12}; u_2 := u_{21};$
40	$v_2 := \mathbf{merge}(v_{12}, v_{22});$	$v_2 := v_{12};$

This time, the algorithm is equivalent to running Algorithm **svd_step** on the top-right and bottom-left blocks, and so $U_1^{(n+1)}$ and $V_2^{(n+1)}$ are the outer factors

from an SVD step on the top-right block and $U_2^{(n+1)}$ and $V_1^{(n+1)}$ are the outer factors from an SVD step on the bottom-left block.

By the existence of the CSD step, the two versions of the algorithm discussed above produce identical $U_1^{(n+1)}$, $U_2^{(n+1)}$, $V_1^{(n+1)}$, and $V_2^{(n+1)}$. Hence, by Theorem 3.5, the two versions produce identical Givens rotations. Therefore, in each invocation of **merge** in the final algorithm, one of the following holds: both vectors are nonzero and colinear, or one vector is the zero vector and the other vector points along the second coordinate axis, or both vectors equal the zero vector. In any case, the call to **merge** is well defined. Therefore, the final algorithm simultaneously computes SVD steps for all four blocks.

The final two lines of code could be implemented (inefficiently) with a single call to **bidiagonalize**. It is straightforward to check that this would only change the signs of entries and would compute $\theta^{(n+1)}$ and $\phi^{(n+1)}$. □

5 Algorithm csd

Algorithm **csd** is the driver algorithm, responsible for initiating Phase I and Phase II.

Given an m-by-m unitary matrix X and integers p, q with $0 \le q \le p$ and $p + q \le m$, **csd**(X, p, q) should compute $\theta = (\theta_1, \ldots, \theta_q) \in [0, \frac{\pi}{2}]^q$, $U_1 \in U(p)$, $U_2 \in U(m-p)$, $V_1 \in U(q)$, and $V_2 \in U(m-q)$ satisfying

$$X \approx \begin{bmatrix} U_1 & \\ & U_2 \end{bmatrix} \left[\begin{array}{cc|cc} C & & S & 0 & 0 \\ & 0 & 0 & I_{p-q} & 0 \\ \hline -S & & C & 0 & 0 \\ & 0 & 0 & 0 & I_{m-p-q} \end{array}\right] \begin{bmatrix} V_1 & \\ & V_2 \end{bmatrix}^*,$$

with $C = \mathrm{diag}(\cos(\theta_1), \ldots, \cos(\theta_q))$ and $S = \mathrm{diag}(\sin(\theta_1), \ldots, \sin(\theta_q))$. It would be preferable to replace the approximate equality with an error bound, but a formal stability analysis is left to the future.

csd's responsibility is to invoke **bidiagonalize** once and then **csd_step** repeatedly. See Algorithm 5.1. The subtleties are the deflation procedure and the choice of shifts.

Deflation If the matrix obtained after n CSD steps,

$$\begin{bmatrix} B_{11}^{(n)} & B_{12}^{(n)} \\ B_{21}^{(n)} & B_{22}^{(n)} \end{bmatrix} = \begin{bmatrix} B_{11}(\theta^{(n)}, \phi^{(n)}) & B_{12}(\theta^{(n)}, \phi^{(n)}) \\ B_{21}(\theta^{(n)}, \phi^{(n)}) & B_{22}(\theta^{(n)}, \phi^{(n)}) \end{bmatrix},$$

has the form (4.5), then the next CSD step can focus on just a principal submatrix. The indices $\underline{i}$ and $\bar{i}$ are determined by $\phi^{(n)}$ as follows.

$$\begin{cases} \phi_{\bar{i}}^{(n)} = \cdots = \phi_{q-1}^{(n)} = 0 \\ \text{none of } \phi_{\underline{i}}^{(n)}, \ldots, \phi_{\bar{i}-1}^{(n)} \text{ equal } 0 \\ \underline{i} \text{ is as small as possible} \end{cases} \tag{5.1}$$

Algorithm 5.1 (**csd**) Given X, p, and q, the following algorithm computes the complete CS decomposition in terms of θ, U_1, U_2, V_1, and V_2.

1 Find $\theta^{(0)}$, $\phi^{(0)}$, P_1, P_2, Q_1, and Q_2 with **bidiagonalize**;
2 If any $\theta_i^{(0)}$ or $\phi_i^{(0)}$ is negligibly different from 0 or $\frac{\pi}{2}$, then round;

4 $n := 0$;

6 Set $\underline{i}$ and $\bar{i}$ as in (5.1);

8 while $\bar{i} > 1$

10 if some $\theta_i^{(n)}$, $\underline{i} \le i \le \bar{i}$, is negligibly far from $\frac{\pi}{2}$, then
11 $\mu := 0$; $\nu := 1$;
12 elseif some $\theta_i^{(n)}$, $\underline{i} \le i \le \bar{i}$, is negligibly far from 0, then
13 $\mu := 1$; $\nu := 0$;
14 else
15 Select shifts μ and ν satisfying $\mu^2 + \nu^2 = 1$; (* see discussion *)
16 end if

18 Compute $\theta^{(n+1)}$, $\phi^{(n+1)}$, $U_1^{(n+1)}$, $U_2^{(n+1)}$, $V_1^{(n+1)}$, $V_2^{(n+1)}$ with **csd_step**;

20 $n := n + 1$;
21 If any $\theta_i^{(n)}$ or $\phi_i^{(n)}$ is negligibly different from 0 or $\frac{\pi}{2}$, then round;
22 Set $\underline{i}$ and $\bar{i}$ as in (5.1);

24 end while

26 $U_1 := P_1((U_1^{(1)} U_1^{(2)} \cdots U_1^{(n)}) \oplus I_{p-q})$; $U_2 := P_2((U_2^{(1)} U_2^{(2)} \cdots U_2^{(n)}) \oplus I_{m-p-q})$;
27 $V_1 := Q_1(V_1^{(1)} V_1^{(2)} \cdots V_1^{(n)})$; $V_2 := Q_2((V_2^{(1)} V_2^{(2)} \cdots V_2^{(n)}) \oplus I_{m-2q})$;

Hence, deflation occurs when some $\phi_i^{(n)}$ transitions from nonzero to zero.

Note that some $\theta_i^{(n)}$ may equal 0 or $\frac{\pi}{2}$ or some $\phi_i^{(n)}$ may equal $\frac{\pi}{2}$, producing zero entries in the matrix without satisfying (5.1). Fortunately, in this case one of the blocks has a zero on its diagonal, the appropriate shift is set to zero, and deflation occurs in the one block just as in the bidiagonal SVD algorithm of Demmel and Kahan [6]. (This is the reason for Case 2 of the preferred QR factorization.) It is easy to check that then one of the $\phi_i^{(n+1)}$ becomes zero. Hence, as soon as a zero appears anywhere in any of the four bidiagonal bands, the entire matrix in bidiagonal block form deflates in at most one more step (assuming exact arithmetic).

Choice of shift There are two natural possibilities for choosing shifts in line 15.

– One possibility is to let μ or ν be a Wilkinson shift (the smaller singular value of the trailing 2-by-2 submatrix of one of the blocks) from which the other shift follows by $\mu^2 + \nu^2 = 1$. This seems to give preference to one block over the other three, but as mentioned above, as soon as one block attains a zero, the other three blocks follow immediately.
– Another possibility is to let μ and ν be singular values of appropriate blocks, i.e., "perfect shifts." This keeps the number of CSD steps as small as possible [10, p. 417] at the cost of extra singular value computations.

Based on what is known about the SVD problem and limited testing with the CSD algorithm, Wilkinson shifts appear to be the better choice, but more real world experience is desirable.

6 On numerical stability

So far, the input matrix X has been assumed exactly unitary, and exact arithmetic has been assumed as well. What happens in a more realistic environment? The algorithm is designed for numerical stability. All four blocks are considered simultaneously and with equal regard. Nearly every computation is based on information from two different sources, from which the algorithm can choose the more reliable source.

Below, some numerical issues and strategies are addressed. Then results from numerical experiments are presented. A BLAS/LAPACK-based implementation is available from the author's web site for further testing.

6.1 Numerical issues and strategies

The better of two vectors in **bidiagonalize**. As mentioned in the discussion after Algorithm 2.2, most of the Householder reflectors used in the bidiagonalization procedure are determined by pairs of colinear vectors. If the input matrix X is not exactly unitary, then some of the pairs of vectors will not be exactly colinear. Each of the computations for u_1, u_2, v_1, and v_2 in Algorithm 2.2 performs an averaging of two different vectors in which the vector of greater norm is weighted more heavily. (Vectors of greater norm tend to provide more reliable information about direction.)

Implementation of **bulge_start**. The implementation of **bulge_start** (Algorithm 3.12) requires care to minimize roundoff error. LAPACK's DBDSQR provides guidance [1].

Implementation of **merge**. In **csd_step**, most Givens rotations can be constructed from either (or both) of two colinear vectors. The function **merge** in the pseudocode is shorthand for a somewhat more complicated procedure in our implementation:

– If the Givens rotation is chasing an existing bulge in one block and introducing a new bulge in the other block, then base the Givens rotation

entirely on the existing bulge. (Maintaining bidiagonal block form is crucial.)
- If the Givens rotation is chasing existing bulges in both blocks, then take a weighted average of the two vectors in the call to **merge**, waiting the longer vector more heavily. (Vectors of greater norm provide more reliable information about direction.)
- If the Givens rotation is introducing new bulges into both blocks, then base the computation solely on the block associated with the smaller shift, either μ or ν. (In particular, when one shift is zero, this strategy avoids roundoff error in **bulge_start**.)

Representation of $\theta^{(n)}$ and $\phi^{(n)}$. Angles of 0 and $\frac{\pi}{2}$ play a special role in the deflation procedure. Because common floating-point architectures represent angles near 0 more precisely than angles near $\frac{\pi}{2}$, it may be advisable to store any angle ψ as a pair $(\psi, \frac{\pi}{2} - \psi)$. This may provide a minor improvement but appears to be optional.

Rounding of $\theta^{(n)}$ and $\phi^{(n)}$ in **csd**. To encourage fast termination, some angles in $\theta^{(n)}$ and $\phi^{(n)}$ may need to be rounded when they are negligibly far from 0 or $\frac{\pi}{2}$. The best test for negligibility will be the subject of future work.

Disagreement of singular values when using perfect shifts. If in **csd**, the shifts μ and ν are chosen to be perfect shifts, i.e., singular values of appropriate blocks, then the computed shifts may not satisfy $\mu^2 + \nu^2 = 1$ exactly in the presence of roundoff error. Empirically, satisfying $\mu^2 + \nu^2 = 1$ appears to be crucial. Either μ or ν should be set to a singular value no greater than $\frac{1}{\sqrt{2}}$ and then the other shift computed from $\mu^2 + \nu^2 = 1$.

6.2 Results of numerical experiments

The sharing of singular vectors in (1.4) is often seen as a hindrance to numerical computation. But in our algorithm, the redundancy appears to be a source of robustness—when one block provides questionable information, a neighboring block may provide more reliable information. Numerical tests support this claim.

Our criteria for a stable CSD computation,

$$X \approx \begin{bmatrix} U_1 & \\ & U_2 \end{bmatrix} \left[\begin{array}{c|c|cc} C & S & 0 & 0 \\ \hline 0 & 0 & I_{p-q} & 0 \\ \hline -S & C & 0 & 0 \\ \hline 0 & 0 & 0 & I_{m-p-q} \end{array}\right] \begin{bmatrix} V_1 & \\ & V_2 \end{bmatrix}^*,$$

are the following: If X is nearly unitary,

$$\|X^* X - I_m\|_2 = \varepsilon, \tag{6.1}$$

then we desire $\theta = (\theta_1, \ldots, \theta_q)$ and U_1, U_2, V_1, V_2 such that

$$C = \operatorname{diag}(\cos(\theta_1), \ldots, \cos(\theta_q)) \qquad S = \operatorname{diag}(\sin(\theta_1), \ldots, \sin(\theta_q))$$

$$\|U_1^* U_1 - I_p\|_2 \approx \varepsilon \qquad \|U_2^* U_2 - I_{m-p}\|_2 \approx \varepsilon$$

$$\|V_1^* V_1 - I_q\|_2 \approx \varepsilon \qquad \|V_2^* V_2 - I_{m-q}\|_2 \approx \varepsilon$$

$$\begin{bmatrix} C \\ 0 \end{bmatrix} = U_1^T (X_{11} + E_{11}) V_1, \quad \|E_{11}\|_2 \approx \varepsilon$$

$$\begin{bmatrix} S & 0 & 0 \\ 0 & I_{p-q} & 0 \end{bmatrix} = U_1^T (X_{12} + E_{12}) V_2, \quad \|E_{12}\|_2 \approx \varepsilon$$

$$\begin{bmatrix} -S \\ 0 \end{bmatrix} = U_2^T (X_{21} + E_{21}) V_1, \quad \|E_{21}\|_2 \approx \varepsilon$$

$$\begin{bmatrix} C & 0 & 0 \\ 0 & 0 & I_{m-p-q} \end{bmatrix} = U_2^T (X_{22} + E_{22}) V_2, \quad \|E_{22}\|_2 \approx \varepsilon.$$

Van Loan's example. Our first test case is based on an example of Van Loan [19]. Let

$$X_{11} = \begin{bmatrix} 0.220508860423 & -0.114095899416 & 0.001410518052 & 0.309131888087 \\ 0.075149984350 & 0.552192330457 & 0.309420137864 & 0.519525649668 \\ 0.346099513974 & -0.465523358094 & -0.147474170901 & 0.284504924779 \\ 0.200314808251 & 0.015869922033 & 0.063768831702 & 0.364621650530 \end{bmatrix}$$

$$X_{12} = \begin{bmatrix} 0.123868614848 & -0.424487382687 & 0.756283107266 & -0.274401793502 \\ 0.505660921957 & 0.028765021298 & -0.138696588123 & 0.219160328651 \\ -0.068044487719 & -0.292950312278 & -0.202722377746 & 0.655183291894 \\ -0.339461927716 & -0.307319405113 & -0.530848659627 & -0.575436177767 \end{bmatrix}$$

$$X_{21} = \begin{bmatrix} -0.149903307775 & 0.456869095895 & -0.814555019070 & 0.205461483909 \\ -0.132593956233 & 0.403919514293 & 0.374067025998 & -0.294979263882 \\ 0.631588073183 & 0.226164206817 & 0.132173742848 & 0.047014825861 \\ -0.588949720476 & -0.205112923304 & 0.239887841318 & 0.537774110108 \end{bmatrix}$$

$$X_{22} = \begin{bmatrix} -0.211288905103 & -0.065095708488 & 0.064582503584 & 0.100169729053 \\ -0.422173038686 & -0.565182436669 & 0.079260723473 & 0.297296887111 \\ -0.473064671229 & 0.502284642254 & 0.218767959397 & 0.079539299401 \\ -0.403033356444 & 0.250518329548 & 0.166101999167 & 0.107399029584 \end{bmatrix},$$

and let $X = \left[\begin{smallmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{smallmatrix}\right]$. Van Loan considered the submatrix $\left[\begin{smallmatrix} X_{11} \\ X_{21} \end{smallmatrix}\right]$.

X satisfies (6.1) with $\varepsilon \approx 3.4 \times 10^{-12}$. Our implementation produces θ, U_1, U_2, V_1, and V_2 for which $\|U_1^T U_1 - I_4\|_2 \approx 1.2 \times 10^{-15}$, $\|U_2^T U_2 - I_4\|_2 \approx 1.4 \times 10^{-15}$, $\|V_1^T V_1 - I_4\|_2 \approx 5.8 \times 10^{-16}$, $\|V_2^T V_2 - I_4\|_2 \approx 1.7 \times 10^{-15}$, $\|E_{11}\|_2 \approx 1.3 \times 10^{-12}$, $\|E_{12}\|_2 \approx 6.1 \times 10^{-13}$, $\|E_{21}\|_2 \approx 1.6 \times 10^{-12}$, and $\|E_{22}\|_2 \approx 9.3 \times 10^{-13}$. The algorithm performs stably.

Haar measure. Let X be a random 40-by-40 orthogonal matrix from Haar measure, let $p = 18$ and $q = 15$, and compute the CS decomposition of X using **csd**. Actually, it is impossible to sample exactly from Haar measure in floating-point, so define X by the following Matlab code.

```
[X,R] = qr(randn(40)); X = X * diag(sign(randn(40,1)));
```

Over 1000 trials of our implementation, $\|U_1^T U_1 - I_{18}\|_2$, $\|U_2^T U_2 - I_{22}\|_2$, $\|V_1^T V_1 - I_{15}\|_2$, $\|V_2^T V_2 - I_{25}\|_2$, $\|E_{11}\|_2$, $\|E_{12}\|_2$, $\|E_{21}\|_2$, and $\|E_{22}\|_2$ were all less than 2ε, where ε was defined to be the greater of ten times machine epsilon or $\|X^T X - I_{40}\|_2$.

Clusters of singular values. Let $\delta_1, \delta_2, \delta_3, \ldots, \delta_{21}$ be independent and identically distributed random variables each with the same distribution as $10^{-18U(0,1)}$, in which $U(0,1)$ is a random variable uniformly distributed on $[0,1]$. For $i = 1, \ldots, 20$, let $\theta_i = \frac{\pi}{2} \cdot \frac{\sum_{k=1}^{i} \delta_k}{\sum_{k=1}^{21} \delta_k}$, and let $C = \mathrm{diag}(\cos(\theta_1), \ldots, \cos(\theta_{20}))$, $S = \mathrm{diag}(\sin(\theta_1), \ldots, \sin(\theta_{20}))$, and

$$X = \begin{bmatrix} U_1 & \\ & U_2 \end{bmatrix} \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} V_1 & \\ & V_2 \end{bmatrix}^T,$$

in which U_1, U_2, V_1, and V_2 are random orthogonal matrices from Haar measure. (These matrices can be sampled as X was sampled in the previous test case.) The random matrix X is designed so that C and S have clustered singular values, as well as singular values close to 0 and 1. Such singular values break the naive CSD algorithm. Compute the CSD of X with $p = q = 20$ using **csd**. Over 1000 trials of our implementation, $\|U_1^T U_1 - I_{18}\|_2$, $\|U_2^T U_2 - I_{22}\|_2$, $\|V_1^T V_1 - I_{15}\|_2$, $\|V_2^T V_2 - I_{25}\|_2$, $\|E_{11}\|_2$, $\|E_{12}\|_2$, $\|E_{21}\|_2$, and $\|E_{22}\|_2$ were all less than 3ε, with ε defined as in the previous test case.

$\theta^{(0)}$ and $\phi^{(0)}$ chosen uniformly from $\left[0, \frac{\pi}{2}\right]$. Choose $\theta_1, \ldots, \theta_{20}$ and $\phi_1, \ldots, \phi_{19}$ independently and uniformly from the interval $\left[0, \frac{\pi}{2}\right]$, and let

$$X = \begin{bmatrix} B_{11}(\theta, \phi) & B_{12}(\theta, \phi) \\ B_{21}(\theta, \phi) & B_{22}(\theta, \phi) \end{bmatrix}.$$

Compute the CSD of X with $p = q = 20$ using **csd**. Over 1000 trials, $\|U_1^T U_1 - I_{18}\|_2$, $\|U_2^T U_2 - I_{22}\|_2$, $\|V_1^T V_1 - I_{15}\|_2$, $\|V_2^T V_2 - I_{25}\|_2$, $\|E_{11}\|_2$, $\|E_{12}\|_2$, $\|E_{21}\|_2$, and $\|E_{22}\|_2$ were all less than 4ε, with ε defined as above.

$\theta^{(0)}$ and $\phi^{(0)}$ chosen randomly from $\left\{0, \frac{\pi}{4}, \frac{\pi}{2}\right\}$. Repeat the previous test case, but with $\theta_1, \ldots, \theta_{20}$ and $\phi_1, \ldots, \phi_{19}$ chosen uniformly from the three-element set $\left\{0, \frac{\pi}{4}, \frac{\pi}{2}\right\}$. This produces test matrices with many zeros, which can tax the novel aspects of our extension of the SVD step. Over 1000 trials, $\|U_1^T U_1 - I_{18}\|_2$, $\|U_2^T U_2 - I_{22}\|_2$, $\|V_1^T V_1 - I_{15}\|_2$, $\|V_2^T V_2 - I_{25}\|_2$, $\|E_{11}\|_2$, $\|E_{12}\|_2$, $\|E_{21}\|_2$, and $\|E_{22}\|_2$ were all less than ε, with ε defined as above.

Acknowledgement The thoughtful comments of an anonymous referee are appreciated.

Appendix: Additional proofs

A.1 Proof of Lemma 3.16

Proof The proof is by induction on i.

That $T_1 = G_1$ is straightforward to prove.

Assume by induction that $T_j = G_j$, $j = 1, \ldots, i-1$. At line 6 of Algorithm 3.13, $\bar{B} = S_{i-1}^T \cdots S_1^T B T_1 \cdots T_{i-1}$, and so

$$\bar{B}^T \bar{B} - \sigma^2 I = T_{i-1}^T \cdots T_1^T (B^T B - \sigma^2 I) T_1 \cdots T_{i-1}.$$

At line 4 of Algorithm 3.7,

$$\bar{A} = G_{i-1}^T \cdots G_1^T (A - \lambda I) G_1 \cdots G_{i-1}.$$

By the induction hypothesis and the fact that $B^T B - \sigma^2 I = A - \lambda I$, we have (at the current step),

$$\bar{B}^T \bar{B} - \sigma^2 I = \bar{A}. \tag{A.1}$$

We show $T_i = G_i$ up to signs using three cases.

Case 1: $\bar{A}(i : i+1, i-1)$ *is nonzero.* From (A.1), $\bar{A}(i : i+1, i-1) = \bar{B}(i-1, i-1)\bar{B}(i-1, i : i+1)^T$. Hence, $\bar{B}(i-1, i : i+1)^T$ is nonzero, $\bar{A}(i : i+1, i-1)$ is colinear with $\bar{B}(i-1, i : i+1)^T$, and $T_i = G_i$.

Case 2: $\bar{A}(i : i+1, i-1)$ *and* $\bar{B}(i-1, i : i+1)^T$ *equal the zero vector.* Then the Givens rotations T_i and G_i are based on $\bar{A}(i : i+1, i)$ and $(\|\bar{B}(:, i)\|^2 - \sigma^2, \bar{B}(:, i)^T \bar{B}(:, i+1))^T$, respectively, which are equal according to (A.1). Hence, $T_i = G_i$.

Case 3: $\bar{A}(i : i+1, i-1)$ *is the zero vector but* $\bar{B}(i-1, i : i+1)$ *is nonzero.* Then $\bar{B}$ must have the form

$$\bar{B} = \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & 0 & a & b \\ & & & c & d \\ & & & & * \end{bmatrix}$$

with a and b not both zero. Let's roll back the previous Givens rotation:

$$\left(S_{i-1}^T\right)^{-1} \bar{B} = S_{i-1} \bar{B} = \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & 0 & x & \\ & & 0 & y & z \\ & & & & * \end{bmatrix}.$$

Hence,

$$\bar{A} = \bar{B}^T\bar{B} - \sigma^2 I = (S_{i-1}\bar{B})^T(S_{i-1}\bar{B}) - \sigma^2 I$$
$$= \begin{bmatrix} * & * & & & & \\ * & * & * & & & \\ & * & * & 0 & 0 & \\ & & 0 & x^2+y^2-\sigma^2 & yz & \\ & & 0 & yz & * & * \\ & & & & * & * \end{bmatrix}.$$

Now, $S_{i-1} = \mathbf{givens}(i-1, i, (x^2-\sigma^2, xy)^T) = \pm\frac{1}{r}\begin{bmatrix} x^2-\sigma^2 & -xy \\ xy & x^2-\sigma^2 \end{bmatrix}$, in which $r = \|(x^2-\sigma^2, xy)^T\|$, so

$$\bar{B} = S_{i-1}^T S_{i-1}\bar{B} = \begin{bmatrix} * & * & & & & \\ & * & * & & & \\ & & 0 & \pm\frac{x}{r}(x^2+y^2-\sigma^2) & \pm\frac{x}{r}yz & \\ & & 0 & \pm\frac{y}{r}(-\sigma^2) & \pm\frac{1}{r}(x^2-\sigma^2)z & \\ & & & & * & * \\ & & & & & * \end{bmatrix}.$$

By assumption, $\bar{B}(i-1, i:i+1)^T$ is nonzero, and hence by inspection $\bar{B}(i-1, i:i+1)^T$ and $\bar{A}(i:i+1, i)$ are colinear. Therefore, $T_i = G_i$. □

A.2 Proof of Lemma 3.17

Proof Assume by induction that $S_j = G_j$ for $j = 1, \ldots, i-1$. At line 10 of Algorithm 3.13, $\bar{B} = S_{i-1}^T \cdots S_1^T B T_1 \cdots T_i$, and so

$$\bar{B}\bar{B}^T - \sigma^2 I = S_{i-1}^T \cdots S_1^T (BB^T - \sigma^2 I) S_1 \cdots S_{i-1}.$$

At line 4 of Algorithm 3.7,

$$\bar{A} = G_{i-1}^T \cdots G_1^T (A - \lambda I) G_1 \cdots G_{i-1}.$$

By the induction hypothesis and the fact that $BB^T - \sigma^2 I = A - \lambda I$, we have (at the current step),

$$\bar{B}\bar{B}^T - \sigma^2 I = \bar{A}. \tag{A.2}$$

We show $S_i = G_i$ using three cases.

Case 1: $\bar{A}(i:i+1, i-1)$ *is nonzero*. From (A.2), $\bar{A}(i:i+1, i-1) = \bar{B}(i-1, i)\bar{B}(i:i+1, i)$. Hence, $\bar{B}(i:i+1, i)$ is nonzero, $\bar{A}(i:i+1, i-1)$ is colinear with $\bar{B}(i:i+1, i)$, and $S_i = G_i$.

Case 2: $\bar{A}(i:i+1, i-1)$ *and* $\bar{B}(i:i+1, i)$ *equal the zero vector*. Then the Givens rotations G_i and S_i are based on $\bar{A}(i:i+1, i)$ and $(\|\bar{B}(i, :)\|^2 - \sigma^2, \bar{B}(i, :)\bar{B}(i+1, :))$, respectively, which are equal according to (A.2). Hence, $S_i = G_i$.

Case 3: $\bar{A}(i:i+1, i-1)$ *is the zero vector but* $\bar{B}(i:i+1, i)$ *is nonzero*. Then $\bar{B}$ must have the form

$$\bar{B} = \begin{bmatrix} * & * & & & & \\ & * & 0 & & & \\ & & a & c & & \\ & & b & d & * & \\ & & & & * & * \\ & & & & & * \end{bmatrix}$$

with a and b not both zero. Let's roll back the previous Givens rotation:

$$\bar{B}T_i^{-1} = \bar{B}T_i^T = \begin{bmatrix} * & * & & & & \\ & * & 0 & 0 & & \\ & & x & y & & \\ & & & z & * & \\ & & & & * & * \\ & & & & & * \end{bmatrix}.$$

Hence,

$$\begin{aligned} \bar{A} = \bar{B}\bar{B}^T - \sigma^2 I &= \left(\bar{B}T_i^T\right)\left(\bar{B}T_i^T\right)^T - \sigma^2 I \\ &= \begin{bmatrix} * & * & & & & \\ * & * & 0 & & 0 & \\ & 0 & x^2+y^2-\sigma^2 & & yz & \\ & 0 & & yz & * & * \\ & & & & * & * \\ & & & & & * \end{bmatrix}. \end{aligned}$$

Now, $T_i = \mathbf{givens}(i, i+1, (x^2-\sigma^2, xy)^T) = \pm\frac{1}{r}\begin{bmatrix} x^2-\sigma^2 & -xy \\ xy & x^2-\sigma^2 \end{bmatrix}$, in which $r = \|(x^2-\sigma^2, xy)^T\|$, so

$$\bar{B} = \bar{B}T_i^T T_i = \begin{bmatrix} * & * & & & & \\ & * & 0 & & 0 & \\ & & \pm\frac{x}{r}(x^2+y^2-\sigma^2) & & \pm\frac{y}{r}(-\sigma^2) & \\ & & & \pm\frac{x}{r}yz & \pm\frac{1}{r}(x^2-\sigma^2)z & * \\ & & & & & * \\ & & & & & * \end{bmatrix}.$$

By assumption, $\bar{B}(i:i+1, i)$ is nonzero, and hence by inspection $\bar{B}(i:i+1, i)$ and $\bar{A}(i:i+1, i)$ are colinear. Therefore, $S_i = G_i$. □

References

1. Anderson, E., Bai, Z., Bischof, C., Blackford, L.S., Demmel, J., Dongarra, J.J., Du Croz, J., Hammarling, S., Greenbaum, A., McKenney, A., Sorensen, D.: LAPACK Users' guide, 3rd edn. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1999)

2. Bai, Z.: The CSD, GSVD, their applications and computations. Preprint Series 958. Institute for Mathematics and its Applications, University of Minnesota (1992, April)
3. Bai, Z., Demmel, J.: Computing the generalized singular value decomposition. SIAM J. Sci. Comput. **14**(6), 1464–1486 (1993)
4. Davis, C., Kahan, W.M.: Some new bounds on perturbation of subspaces. Bull. Am. Math. Soc. **75**, 863–868 (1969)
5. Davis, C., Kahan, W.M.: The rotation of eigenvectors by a perturbation, III. SIAM J. Numer. Anal. **7**, 1–46 (1970)
6. Demmel, J., Kahan, W.: Accurate singular values of bidiagonal matrices. SIAM J. Sci. Statist. Comput. **11**(5), 873–912 (1990)
7. Edelman, A., Sutton, B.D.: The beta-Jacobi matrix model, the CS decomposition, and generalized singular value problems. Found. Comput. Math. **8**(2), 259–285 (2008)
8. Golub, G., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix. J. Soc. Ind. Appl. Math., Ser. B Numer. Anal. **2**, 205–224 (1965)
9. Golub, G.H., Reinsch, C.: Handbook series linear algebra: Singular value decomposition and least squares solutions. Numer. Math. **14**(5), 403–420 (1970)
10. Golub, G.H., Van Loan, C.F.: Matrix computations, 3rd edn. Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD (1996)
11. Hari, V.: Accelerating the SVD block-Jacobi method. Computing **75**(1), 27–53 (2005)
12. Jordan, C.: Essai sur la géométrie à *n* dimensions. Bull. Soc. Math. Fr. **3**, 103–174 (1875)
13. Paige, C.C.: Computing the generalized singular value decomposition. SIAM J. Sci. Statist. Comput. **7**(4), 1126–1146 (1986)
14. Paige, C.C., Saunders, M.A.: Towards a generalized singular value decomposition. SIAM J. Numer. Anal. **18**(3), 398–405 (1981)
15. Paige, C.C., Wei, M.: History and generality of the CS decomposition. Linear Algebra Appl. **208/209**, 303–326 (1994)
16. Stewart, G.W.: On the perturbation of pseudo-inverses, projections and linear least squares problems. SIAM Rev. **19**(4), 634–662 (1977)
17. Stewart, G.W.: Computing the *CS* decomposition of a partitioned orthonormal matrix. Numer. Math. **40**(3), 297–306 (1982)
18. Sutton, B.D.: The stochastic operator approach to random matrix theory, Ph.D. thesis. Massachusetts Institute of Technology, Cambridge, MA (2005)
19. Van Loan, C.: Computing the CS and the generalized singular value decompositions. Numer. Math. **46**(4), 479–491 (1985)
20. Watkins, D.S.: Some perspectives on the eigenvalue problem. SIAM Rev. **35**(3), 430–471 (1993)