Singular Value Decomposition Algorithms for Embedded Systems: A Comprehensive Treatment

Claudio Turchetti, Laura Falaschetti and Lorenzo Manoni

1 Matrix Algebra Preliminaries

1.1 Householder Transformation

It is based on an $(n \times n)$ symmetric matrix of the form

$$U = I - 2\,\frac{uu^T}{u^T u} \tag{1}$$

where u is the Householder vector. A matrix of this kind is also called a Householder reflection, due to the following specific geometry property. A given vector x can always be represented as

$$x = \alpha u + w,\ \alpha \in \mathbb{R} \tag{2}$$

where w is orthogonal to u. By applying the transformation U to x one gets the vector

$$Ux = (I - 2\frac{uu^T}{u^T u})(\alpha u + w) = -\alpha u + w\ , \tag{3}$$

representing the reflected vector of x with respect to the hyperplane spanned by w. Thanks to this property the Householder transformation can be used to zero selected components of a vector. To show this let us choose u such that

$$u = x \pm \|x\| e_1 \tag{4}$$

where $e_1 = (1, 0, \ldots, 0)^T$, and $\|\cdot\|$ is the Euclidean norm of a vector. By applying the transformation U so obtained to x, then it results

$$Ux = \mp\|x\| e_1 \tag{5}$$

meaning that all the components of x but the first one are zeroed. A common choice to avoid errors when $x_1 \approx \|x\|$ is to pose $u = x + \text{sign}(x_1)\|x\|e_1$. The method can be generalized to the k-th component as follows. For the generic index $k \in \{1, \ldots, n\}$, let us define

$$u = [0, \ldots, 0, x_k \pm s, x_{k+1}, \ldots, x_n]^T \tag{6}$$

where $s = \sqrt{x_k^2 + \ldots + x_n^2}$. The resulting Householder matrix U when applied to x gives

$$Ux = [x_1, x_2, \ldots, x_{k-1}, \mp s, 0, \ldots, 0]^T \ , \tag{7}$$

that is U leaves the first $k-1$ components unchanged, changes the k-th component and zeroes all the residual $n-k$ components. It can be easily shown that U has the block form

$$U_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & \widehat{U} \end{bmatrix} \tag{8}$$

where $\widehat{U}$ acts only on the last $n-k+1$ components of x by zeroing them all but the k-th component. Similarly, the transformation

$$x^T V = x^T(I - 2\frac{v^T v}{v v^T}) \quad , \tag{9}$$

where v is a row vector, acts on the row vector x^T by zeroing some of its components. A useful property of U is that it is not necessary for the matrix to be explicitly derived, indeed the transformation

$$Ux = (I - 2\frac{uu^T}{u^T u})x = x - 2\frac{u^T x}{u^T u}u \tag{10}$$

can be written in terms of u alone.

A mathematical description of the algorithm is shown in Algorithm 1.

Algorithm 1 Householder bidiagonalization

Require: $A \in \mathbb{R}^{m \times n}$

for $k = 1, \ldots, n-1$ **do**

- Determine Householder matrix U_k such that

$U_k\, x = [x_1, x_2, \ldots, x_{k-1}, \mp s, 0, \ldots, 0]^T$

$A \leftarrow U_k\, A$

if $k < n-1$ **then**

- Determine Householder matrix V_k such that

$x^T\, V_k = [x_1, x_2, \ldots, x_k, \mp s, 0, \ldots, 0]$

$A \leftarrow A\, V_k$

end if

end for

1.2 Jacobi Rotation

Householder transformation is useful for zeroing a number of components of a vector. However when it is necessary to zero elements more selectively, Jacobi (or Givens) rotation is able to zero a selected component of a vector. It is based on the Jacobi matrix, also called Givens matrix, denoted by $J(p, q, \theta)$, of the form

$$J(p,q,\theta) = \begin{array}{cc} \left[\begin{array}{ccccccc} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & & & & & \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & & & & & \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & & & & & \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{array}\right] & \begin{array}{c} \\ \\ \mathrm{p} \\ \\ \mathrm{q} \\ \\ \\ \end{array} \\ \begin{array}{ccccccc} & & \mathrm{p} & & \mathrm{q} & & \end{array} & \end{array} \tag{11}$$

where $c = \cos\theta$ and $s = \sin\theta$. Premultiplication of a vector by $J(p, q, \theta)^T$ corresponds to a counterclockwise rotation of θ in the (p, q) plane, that zeroes the q components of the resulting vector y. Indeed, if $x \in \mathbb{R}^n$ and

$$y = J(p, q, \theta)^T x \;, \tag{12}$$

then

$$y_j = \begin{cases} cx_p - sx_q, & j = p \\ sx_p + cx_q, & j = q \\ x_j, & j \neq p, q \end{cases} \quad . \tag{13}$$

From (13) it is clear that y_q can be forced to zero by setting

$$c = \frac{x_p}{\sqrt{x_p^2 + x_q^2}}, \; s = \frac{-x_q}{\sqrt{x_p^2 + x_q^2}} \quad . \tag{14}$$

The Jacobi matrix, when applied as a similarity transformation to a symmetric matrix A,

$$B = J(p, q, \theta)^T A \, J(p, q, \theta) \; , \tag{15}$$

rotates rows and columns p and q of A through the angle θ so that the (p, q) and (q, p) entries are zeroed.

1.3 QR Factorization

This factorization is a fundamental step in QR iteration algorithms. The QR factorization of an $(m \times n)$ matrix A is given by

$$A = QR \tag{16}$$

where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{m \times n}$ is upper triangular. Having derived the properties of Householder transformation, it is straightforward to show that the upper triangular matrix R can be obtained by successive transformations

$$H_n H_{n-1} \ldots H_1 A = R \tag{17}$$

where $H_1, H_2, \ldots, H_n$ are Householder matrices, and so by setting $Q = H_1 \ldots H_n$ we obtain $A = QR$.

2 Algorithms for the Singular Value Decomposition

Let A be a real $(m \times n)$ matrix with $m \geq n$. It is known that the decomposition

$$A = U \Sigma V^T \tag{18}$$

where

$$U^T U = V^T V = V V^T = I, \; \Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n) \tag{19}$$

exists [1]. The matrix U consists of n orthonormal eigenvectors corresponding to the n largest eigenvalues of AA^T, and the matrix V consists of the orthonormal eigenvectors of A^TA. The diagonal elements of Σ are the non-negative square roots of the eigenvalues of AA^T, called singular values. Assuming

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0 \tag{20}$$

thus if $rank(A) = M$, it results $\sigma_{M+1} = \sigma_{M+2} = \ldots = \sigma_n = 0$. The decomposition (18) is called the singular value decomposition (SVD) of matrix A.

2.1 QR Iteration Algorithm

This algorithm is based on the QR factorization and the "power method". Assuming $A \in \mathbb{R}^{n \times n}$ is a symmetric matrix then, by the symmetric Schur decomposition, there exists a real orthogonal Q such that

$$Q^T A Q = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \tag{21}$$

with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Given a vector $q^{(0)} \in \mathbb{R}^n$, such that $\|q^{(0)}\| = 1$, the power method produces a sequence of vectors $q^{(k)}$ as follows:

$$\begin{aligned}
&\textbf{for } k = 1, 2, \ldots \textbf{ do}\\
&\quad z^{(k)} = A\, q^{(k-1)}\\
&\quad q^{(k)} = z^{(k)} / \|z^{(k)}\|\\
&\quad \lambda^{(k)} = [q^{(k)}]^T A\, q^{(k)}\\
&\textbf{end for}
\end{aligned} \tag{22}$$

The power method states that if $q^{(0)} \neq 0$ and $\lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n$ then $q^{(k)}$ converges to an eigenvector and $\lambda^{(k)}$ to the corresponding eigenvalue.

This method can be generalized to solve the eigenvalue problem of a symmetric matrix. To this end let us consider an $(n \times n)$ matrix Q_0 with orthonormal columns and a sequence of matrices $\{Q_k\}$ generated as follows:

$$\begin{aligned} &\textbf{for } \mathrm{k} = 1,\, 2,\, \ldots\ \textbf{do} \\ &\quad Z_k = A\,Q_{k-1} \\ &\quad Q_k\,R_k = \\ &\quad Z_k\ (\textit{QR factorization}) \\ &\textbf{end for} \end{aligned} \tag{23}$$

where the QR factorization is applied at each step to obtain the matrices Q_k and R_k, then (23) defines the so-called *orthogonal iteration*. It can be shown [2] that the matrices T_k defined by

$$T_k = Q_k^T A\,Q_k \tag{24}$$

are converging to a diagonal form whose values $\{\lambda_1^{(k)}, \ldots, \lambda_n^{(k)}\}$ converge to $\{\lambda_1, \ldots, \lambda_n\}$.

From definition of T_{k-1} we have

$$T_{k-1} = Q_{k-1}^T A\,Q_{k-1} = Q_{k-1}^T (A\,Q_{k-1}) = Q_{k-1}^T (Q_k R_k) \tag{25}$$

where the QR factorization of $A\,Q_{k-1}$ has been applied. Similarly, using (23) and orthogonality of Q_{k-1}, one gets

$$T_k = Q_k^T A Q_k = (Q_k^T A Q_{k-1})(Q_{k-1}^T Q_k) = R_k (Q_{k-1}^T Q_k) \quad . \tag{26}$$

Defining $U_k = Q_{k-1}^T Q_k$ the algorithm (23) can be rewritten as

$$\begin{aligned} &T_0 = U_0^T A U_0 \\ &\textbf{for } k = 1, 2, \ldots \textbf{ do} \\ &\quad U_k R_k = \\ &\quad T_{k-1}\ \ (\textit{QR factorization}) \\ &\quad T_k = R_k U_k \\ &\textbf{end for} \end{aligned} \tag{27}$$

where $U_0 \in \mathbb{R}^{n\times n}$ is orthogonal. Since $T_k = R_k U_k = U_k^T (U_k R_k) U_k = U_k^T T_{k-1} U_k$, it follows by induction that

$$T_k = (U_0 U_1 \ldots U_k)^T A\,(U_0 U_1 \ldots U_k) \tag{28}$$

and T_k converges to the diagonal form (21). The iteration (27) establishes the so-called QR *iteration algorithm* for symmetric matrices.

The main limitation of QR algorithm is that it is only valid for symmetric matrices, such as $A^T A$, thus the method cannot be directly applied to the matrix A.

2.2 Golub-Reinsch Algorithm

This method, developed by G. H. Golub and C. Reinsch [3], acts directly on the matrix A thus avoiding unnecessary numerical inaccuracy due to the computation of $A^T A$. The algorithm can be divided into these consecutive steps:

i) Householder's bidiagonalization;

ii) implicit QR method with shift;

2.2.1 Householder's Bidiagonalization

Given the matrix $A \in \mathbb{R}^{n\times m}$ $(n > m)$ the bidiagonal form

$$A = UBV^T,\ U \in \mathbb{R}^{n\times n},\ V \in \mathbb{R}^{m\times m} \tag{29}$$

with

$$B = \begin{bmatrix} \widehat{B} \\ 0 \end{bmatrix} \in \mathbb{R}^{n\times m},$$

$$\widehat{B} = \begin{bmatrix} \psi_1 & \phi_1 & 0 & \cdots & 0 \\ 0 & \psi_2 & \phi_2 & & \\ \vdots & & \ddots & \ddots & \\ & & & \psi_{m-1} & \phi_{m-1} \\ 0 & & & & \psi_m \end{bmatrix} \in \mathbb{R}^{m\times m}, \tag{30}$$

exists. The matrix B can be obtained from A by the successive orthogonal transformation

$$B = U_m \cdots U_1\ A\ V_1 \cdots V_{m-2} \tag{31}$$

where U_k, V_k are Householder's matrices. In particular for the k-th step:

i) an Householder matrix U_k can be defined for zeroing all the last $n-k$ components of the k-th column of B;

ii) an Householder matrix V_k can be defined for zeroing all the $m-k-1$ components of the k-th row of B.

At the end of the process the diagonalization (29) is achieved with $U = U_m \cdots U_1,\ V = V_1 \cdots V_{m-2}$. The computational cost of this process is about $\mathcal{O}(n^3)$.

2.2.2 Implicit QR Method with Shift

The symmetric QR algorithm (27) can be made more efficient in two ways:

i) by choosing U_0 such that $U_0 A U_0 = T_0$ is tridiagonal. In this way all T_k in (27) are tridiagonal and this reduces the complexity of the algorithm to $\mathcal{O}(n^2)$; once the Householder's algorithm is applied to the matrix A giving the bidiagonal matrix $\hat{B}$, the tridiagonal form T_0 can be easily obtained as $T_0 = \hat{B}^T \hat{B}$.

ii) by introducing a shift in the iteration of (27): with this change the convergence to diagonal form proceeds at a cubic rate. This result is based on the following facts:

- *a)* if $s \in \mathbb{R}$ and $T - sI = QR$ is the shifted version of T then $T_+ = RQ + sI$ is also tridiagonal;
- *b)* if s is an eigenvalue of T, $s \in \lambda(T)$, the last column of T_+ equals $s\, e_n = s(0 \dots 1)^T$, that is $T_+(n,n) = s$.

With regard to the second point the algorithm (27) modifies to the following

$$
\begin{array}{l}
T = \hat{B}^T \hat{B} \;\; (\textit{tridiagonal}) \\
\textbf{for } k = 0, 1, \dots \textbf{ do} \\
\quad \textit{Determine real shift } \mu \\
\quad UR = T - \mu I \;\; (\textit{QR factorization}) \\
\quad T = RU + \mu I \\
\textbf{end for}
\end{array}
\tag{32}
$$

where μ is a good approximate eigenvalue and

$$
T = \begin{bmatrix} a_1 & b_1 & \cdots & 0 \\ b_1 & a_2 & \cdots & 0 \\ & \ddots & \ddots & \\ & & & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix} . \tag{33}
$$

An effective choice is to shift by the eigenvalue of

$$
\begin{bmatrix} a_{n-1} & b_{n-1} \\ b_{n-1} & a_n \end{bmatrix} \tag{34}
$$

known as the Wilkinson shift and given by

$$\mu = a_n + d - \mathrm{sign}(d)\sqrt{d^2 + b_{n-1}^2},\ d = (a_{n-1} - a_n)/2 \quad . \tag{35}$$

If μ is a good approximation of the eigenvalue s, then the term b_{n-1} will be smaller after a QR step with shift μ. It has been shown [4] that with this shift strategy, (28) is cubically convergent.

A pseudo-code of the algorithm in shown in Algorithm 2.

Algorithm 2 QR iteration with shift

Require: $A \in \mathbb{R}^{m\times n}$

Apply Algorithm 1 to obtain bidiagonal $\hat{B}$

$T = \hat{B}^T \hat{B}$ tridiagonal

for $k = 1, \ldots$ **do**

- Select $B_{22}(2 \times 2)$: block matrix at the right bottom of $\hat{B}^T\hat{B}$
- Compute eigenvalues λ_1, λ_2 of B_{22}
- Determine shift $\mu = \min(\lambda_1, \lambda_2)$

$T = \mu T = UR$ *(QR factorization)*

$T = RU + \mu I$

end for

It is possible to execute the transition to $T = RU + \mu I$ without explicitly forming the matrix $T - \mu I$, thus giving the implicit shift version [2]. This is achieved by a Given rotation matrix in which $c = \cos(\theta)$ and $s = \sin(\theta)$ are such that

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a_1 - \mu \\ b_1 \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix} \quad . \tag{36}$$

However if we set $J_1 = J(1, 2, \theta)$ we have

$$T \leftarrow J_1^T T J_1 = \begin{bmatrix} x & x & + & 0 & \cdots & 0 \\ x & x & x & & & \vdots \\ + & x & x & & & 0 \\ & & & \ddots & & \\ & & & & x & x \\ 0 & \cdots & \cdots & 0 & x & x \end{bmatrix} \quad . \tag{37}$$

where the two nonzero elements "+" out of the tridiagonals appears. To "chase" these unwanted elements, we can apply rotations $J_2, \ldots, J_{n-1}$ of the form $J_i = J(i, i+1, \theta_i)$, $i = 2, \ldots, n-1$, such that if $z = J_1 J_2 \ldots J_{n-1}$ then $Z^T T Z$ is tridiagonal. In such a way, it can be shown that the tridiagonal matrix produced by this implicit shift technique is the same as the tridiagonal matrix obtained by the explicit method.

A description of implicit QR method with shift is reported in Algorithm 3, while the pseudo-code for Golub-Reinsch algorithm is described in Algorithm 4.

Algorithm 3 QR iteration with implicit shift

Require: $A \in \mathbb{R}^{m \times n}$

Apply Algorithm 1 to obtain bidiagonal $\hat{B}$

$T = \hat{B}^T \hat{B}$ tridiagonal

Compute the eigenvalue μ of $\begin{bmatrix} T_{m-1,m-1} & T_{m-1,m} \\ T_{m,m-1} & T_{m,m} \end{bmatrix}$ that is closer to $T_{m,m}$.

Choose the Givens matrix $J_1 = J(1, 2, \theta)$ such that

$J_1^T \begin{bmatrix} a_1 - \mu \\ b_1 \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix}$

$T = J_1^T T J_1$

for $k = 2, \ldots, m-1$ **do**

  $J_k = J(k, k+1, \theta_k)$

  $Z = J_1 J_2 \ldots J_k$

  $T = Z^T T Z$

end for

Algorithm 4 Golub-Reinsch

Require: $A \in \mathbb{R}^{m\times n}(m \geq n)$, ϵ a small multiple of the unit round-off

Use Algorithm 1 to compute bidiagonalization.

$$\begin{bmatrix} B \\ 0 \end{bmatrix} \leftarrow (U_1 \dots U_n)^T A (V_1 \dots V_{n-2})$$

Repeat

for $i = 1, \dots, n-1$ **do**

- Set $b_{i,i+1}$ to zero if $|b_{i,i+1}| \leq \epsilon(|b_{ii}| + |b_{i+1,i+1}|)$
- Find the largest q and the smallest p such that if

$$B = \begin{array}{cc} \begin{bmatrix} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{bmatrix} & \begin{array}{l} p \\ n-p-q \\ q \end{array} \\ \begin{array}{ccc} p & n-p-q & q \end{array} & \end{array}$$

then B_{33} is diagonal and B_{22} has a nonzero superdiagonal.

if $q = n$ **then**

STOP

end if

if any diagonal entry in B_{22} is zero **then**

zero the superdiagonal entry in same row

else

apply the Algorithm 3

end if

end for

2.3 Demmel-Kahan Algorithm

The structure of the algorithm is based on the Golub-Reinsch algorithm previously described. Nevertheless the Demmel-Kahan's algorithm [5] is able to achieve better performance than that obtained with Golub-Reinsch algorithm, both in terms of convergence speed and in terms of relative accuracy. This algorithm consists of the following main consecutive steps:

i) Householder's bidiagonalization;

ii) QR iteration with zero-shift;

The second step is a variation of the QR standard method with shift, called implicit zero-shift QR algorithm, since it corresponds to the standard algorithm when $\sigma = 0$, which computes all the singular values of a bidiagonal matrix, with guaranteed high relative accuracy.

To show the algorithm, let us take $\sigma = 0$ and refer to a 4×4 matrix example. From (26) one gets $\tan\theta_1 = -b_{12}/b_{11}$ so that the result of the first rotation is

$$B^{(1)} = BJ_1 = \begin{bmatrix} b_{11}^{(1)} & 0 & & \\ b_{21}^{(1)} & b_{22}^{(1)} & b_{23} & \\ & & b_{33} & b_{34} \\ & & & b_{44} \end{bmatrix} . \tag{38}$$

We see that $(1,2)$ entry is zero and, as it will propagate through the rest of the algorithm, this is the key of its effectiveness. After the rotation by J_2 we have

$$B^{(2)} = J_2BJ_1 = \begin{bmatrix} b_{11}^{(2)} & b_{12}^{(2)} & b_{13}^{(2)} & \\ 0 & b_{22}^{(2)} & b_{23}^{(2)} & \\ & & b_{33} & b_{34} \\ & & & b_{44} \end{bmatrix} \tag{39}$$

where

$$\begin{bmatrix} b_{12}^{(2)} & b_{13}^{(2)} \\ b_{22}^{(2)} & b_{23}^{(2)} \end{bmatrix} = \begin{bmatrix} \sin\theta_2 b_{22}^{(1)} & \sin\theta_2 b_{23} \\ \cos\theta_2 b_{22}^{(1)} & \cos\theta_2 b_{23} \end{bmatrix} \tag{40}$$

is a rank one matrix. Postmultiplication by J_3 to zero out the $(1,3)$ entry will also zero out the $(2,3)$ entry:

$$B^{(3)} = J_2BJ_1J_3 = \begin{bmatrix} b_{11}^{(2)} & b_{12}^{(3)} & 0 & \\ 0 & b_{22}^{(3)} & 0 & \\ & b_{32}^{(3)} & b_{33}^{(3)} & b_{34} \\ & & & b_{44} \end{bmatrix} . \tag{41}$$

Rotation by J_4 just repeats the situation: the submatrix of $J_4J_2BJ_1J_3$ consisting of row 2 and 3 and columns 3 and 4 is rank one, and rotation by J_5 zeroes out the $(3,4)$ entry as well as the $(2,4)$ entry. This engine repeats itself for the length of the matrix. Thus at each step of zero-shift algorithm a transformation is applied which takes f and g as input and returns r, $cs = \cos\theta$ and $sn = \sin\theta$ such that

$$\begin{bmatrix} cs & sn \\ -sn & cs \end{bmatrix} \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \quad . \tag{42}$$

The description of this algorithm is shown in Algorithm 5.

Algorithm 5 Demmel-Kahan

Require: $A \in \mathbb{R}^{m\times n}(m \geq n) \in$ a small multiple of the unit round-off

Use Algorithm 1 to compute bidiagonalization.

$$\begin{bmatrix} B \\ 0 \end{bmatrix} \leftarrow (U_1 \dots U_n)^T A (V_1 \dots V_{n-2})$$

Repeat

for $i = 1 : n-1$ **do**

- Set $b_{i,i+1}$ to zero if a relative convergence criterion is met
- Find the largest q and the smallest p such that if

$$B = \begin{array}{cc} \begin{bmatrix} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{bmatrix} & \begin{array}{l} \text{p} \\ \text{n -p -q} \\ \text{q} \end{array} \\ \begin{array}{ccc} \text{p} & \text{n-p-q} & \text{q} \end{array} & \end{array}$$

than B_{33} is diagonal and B_{22} has a nonzero superdiagonal.

if $q = n$ **then**

STOP

end if

if any diagonal entry in B_{22} is zero **then**

zero the superdiagonal entry in same row

else

apply the implicit zero-shift QR algorithm

end if

end for

2.4 Jacobi Rotation Algorithm

In this case given the real and symmetric matrix $A \in \mathbb{R}^{n\times n}$ the algorithm [6] aims to obtain a diagonal matrix $B \in \mathbb{R}^{n\times n}$ through the transformation

$$B = J^T A J \tag{43}$$

where J represents a sequence of rotation matrices. In particular for the k-th rotation or sweep can be rewritten as

$$A_{k+1} = J_k^T A_k J_k, \quad A_0 = A \tag{44}$$

where $J_k = J(p, q, \theta)$ is the Jacobi rotation matrix that rotates rows and columns p and q of A_k through the angle θ so that the (p, q) and (q, p) entries are zeroes. The p and q values are chosen properly at each iteration step. With reference to the sub matrices corresponding to the p, q columns we have

$$\begin{bmatrix} b_{pp} & b_{pq} \\ b_{qp} & b_{qq} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{45}$$

The key point of the algorithm is to determine the rotation coefficients c and s in such a way the off-diagonal terms b_{pq} and b_{qp} are zeroed. To this end the following equation

$$b_{pq} = b_{qp} = a_{pq}\,(c^2 - s^2) + (a_{pp} - a_{qq})\,s\,c = 0 \tag{46}$$

has to be solved. Posing $t = s/c$ and after some manipulation, (46) is equivalent to the equation

$$t^2 + 2t\tau - 1 = 0 \tag{47}$$

with $\tau = \frac{a_{qq}-a_{pp}}{2a_{pq}}$. Choosing the root with minimum value and corresponding to a rotation angle $\theta \leq |r/4|$, it results

$$t = \frac{\mathrm{sign}(\tau)}{|\tau| + \sqrt{\tau^2+1}}, \quad c = \frac{1}{\sqrt{t^2+1}}, \quad s = t\,c \quad . \tag{48}$$

A characteristic of the algorithm particularly important for convergence is that after a rotation the off-diagonal terms reduce. To show this property by computing the Frobenius norm of both terms in (45) and using the property that such a norm is invariant under orthogonal transformation, we have

$$a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2 = b_{pp}^2 + b_{qq}^2 + 2b_{pq}^2 = b_{pp}^2 + b_{qq}^2 \tag{49}$$

As a consequence the off-diagonal contribution off(B) to the norm $\|B\|_F^2$ after a rotation is given by

$$\begin{aligned} \text{off}(B)^2 = \|B\|_F^2 - \sum_{i=1}^{n} b_{ii}^2 = \\ \|A\|_F^2 - \sum_{i=1}^{n} a_{ii}^2 + (a_{pp}^2 + a_{qq}^2 - b_{pp}^2 - b_{qq}^2) = \\ \text{off}(A)^2 - 2a_{pq}^2 \quad . \end{aligned} \tag{50}$$

This result clearly shows that the extra-diagonal contribution is diminished of a value $2a_{pq}^2$.

The algorithm is reported in Algorithm 6.

Algorithm 6 Jacobi rotation

Require: $A \in \mathbb{R}^{n\times n}$ symmetric

$B \leftarrow A \in \mathbb{R}^{n\times n}$

Repeat

for $i = 1 : n-1$ **do**

for $j = i+1 : n$ **do**

Compute the rotations coefficients s, c such that

$$\begin{bmatrix} b_{ii} & 0 \\ 0 & b_{jj} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{51}$$

if off$(A) < \epsilon$, where off$(A) = \sqrt{\sum_{i\neq j} a_{ij}^2}$ and ϵ is a small multiple of the unit round-off **then**

STOP

end if

end for

end for

2.5 One-Sided Jacobi Rotation Algorithm

The main idea of this algorithm [7] is to rotate columns i and j of A through the angle θ so that they become orthogonal to each other. In such a way the

(i, j) element of $A^T A$ is implicitly zeroed resulting in the scalar product of the i, j columns.

Let $J(i, j, \theta)$ be the Givens matrix that when applied to the matrix A yields

$$B = (b_{:1} \cdots b_{:i} \cdots b_{:n}) = AJ = (a_{:1} \cdots a_{:i} \cdots a_{:n}) \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \ddots & & c & s & \\ & & -s & c & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{52}$$

where the i, j columns of B are given by

$$\begin{aligned} b_{:i} &= c\, a_{:i} - s\, a_{:j} \\ b_{:j} &= s\, a_{:i} + c\, a_{:j} \end{aligned} \tag{53}$$

we now determine the element $(B^T B)_{ij}$

$$(B^T B)_{ij} = b_{:i}^T \ b_{:j} = (c\, a_{:i} - s\, a_{:j})^T (s\, a_{:i} + c\, a_{:j}) \quad . \tag{54}$$

Assuming $(B^T B)_{ij} = 0$, $i \neq j$ we obtain

$$cs \left(\|a_{:i}\|^2 - \|a_{:j}\|^2 \right) + (c^2 - s^2)(a_{:i}^T \ a_{:j}) = 0 \quad . \tag{55}$$

By dividing for c^2 and posing

$$\begin{aligned} t &= s/c \\ \alpha &= \|a_{:i}\|^2 \\ \beta &= \|a_{:j}\|^2 \\ \gamma &= a_{:i}^T \ a_{:j} \end{aligned} \tag{56}$$

the following quadratic equation is obtained

$$t^2 + 2\tau t - 1 = 0 \tag{57}$$

where $\tau = (\beta - \alpha)/2\gamma$. Solving the previous equation and choosing the root that is smaller in absolute value,

$$t = \min | - \tau \pm \sqrt{1 + \tau^2}| \tag{58}$$

finally we have

$$c = \frac{1}{\sqrt{1 + t^2}}, \quad s = ct \quad . \tag{59}$$

In such a way the elements $(B^T B)_{ij}$ and $(B^T B)_{ji}$ of the product $B = AJ(i, j, \theta)$ with c and s so derived, are zeroed.

The pseudo-code of the algorithm is reported in Algorithm 7.

Algorithm 7 One-sided Jacobi rotation

Require: $A \in \mathbb{R}^{n\times n}$ symmetric

$B \leftarrow A \in \mathbb{R}^{n\times n}$

Repeat

for $i = 1 : n - 1$ **do**

  for $j = i + 1 : n$ **do**

    Compute the rotations coefficients s, c such that

$$\begin{bmatrix} b_{ii} & 0 \\ 0 & b_{jj} \end{bmatrix} = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{60}$$

    if $\text{off}(A) < \epsilon$, where $\text{off}(A) = \sqrt{\sum_{i\neq j} a_{ij}^2}$ and ϵ is a small multiple of the unit round-off **then**

      STOP

    end if

  end for

end for

2.6 Divide and Conquer Algorithm

For a square symmetric matrix A a relationship between singular values and eingenvalues, as well as between singular vectors and eigenvectors exists. Indeed, as A can be diagonalized we can write

$$A = Q\Lambda Q^T = A\,\text{sign}(\Lambda)|\Lambda|Q^T \tag{61}$$

where Λ is the eigenvalue diagonal matrix and Q is a unitary matrix (orthogonal in real case). Since we can always assume that the elements of $|\Lambda|$ are in decreasing order, then to the diagonalization (61) corresponds an SVD such that $U = Q\,\text{sign}(\Lambda)$, $\Sigma = |\Lambda|$ and $V^T = Q^T$. In words the singular values are the absolute values of eigenvalues and the singular vectors are the eigenvectors (same norm and direction, but not necessary the same versus).

For a non symmetric matrix the singular values and the singular vectors are not directly related to the eigenvalues and eigenvectors, instead there is a strict relationship with eigenvalues and eigenvectors of the symmetric matrices $A^TA \in \mathbb{R}^{M\times M}$ and $AA^T \in \mathbb{R}^{N\times N}$. In fact it can be easily shown that

$$\begin{aligned} A^TA &= V\Sigma^2V^T \\ AA^T &= U\Sigma^2U^T \end{aligned} \tag{62}$$

where A is decomposed as $A = U\Sigma V^T$. From (62) it results that the singular values of A are eigenvalues of the matrices A^TA and AA^T (except those equal to zero for the latter). Additionally the right and left singular vectors are the eigenvectors of A^TA and AA^T respectively, which can differ for a sign (note that the matrix A can be correctly reconstructed provided the correct sign is known). Therefore, on the basis of previous considerations, the singular value decomposition reduces to the eigenvalue problem of a symmetric matrix. In general, the methods for solving such a problem are iterative methods that include two stages:

i) in the first stage the matrix A is transformed to a matrix B whose structure makes the computation of the eigenvalues and eigenvectors easier. A typical choice, that is assumed here, is a tridiagonal form.

ii) in the second stage an iterative method is applied to determine the eigenvalues and eigenvectors.

With reference to the second stage the divide and conquer algorithm aims at reducing a complex problem to a singular one [8]. The algorithm is intended to be applied to a tridiagonal and symmetric matrix T of dimension $N \times N$.

2.6.1 Divide Operation

In the algorithm first an operation of "divide" is performed that transforms the matrix T to the two matrices T_1 and T_2 as

$$
T = \left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & a_2 & \ddots & & & & & \\
 & \ddots & \ddots & b_{m-1} & & & & \\
 & & b_{m-1} & a_m & b_m & & & \\
\hline
 & & & b_m & a_{m+1} & b_{m+1} & & \\
 & & & & b_{m+1} & a_{m+2} & \ddots & \\
 & & & & & \ddots & \ddots & b_{n-1} \\
 & & & & & & b_{n-1} & a_n
\end{array}\right] =
$$

$$
\left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & a_2 & \ddots & & & & & \\
 & \ddots & \ddots & b_{m-1} & & & & \\
 & & b_{m-1} & a_m \mp b_m & & & & \\
\hline
 & & & & a_{m+1} \mp b_m & b_{m+1} & & \\
 & & & & b_{m+1} & a_{m+2} & \ddots & \\
 & & & & & \ddots & \ddots & b_{n-1} \\
 & & & & & & b_{n-1} & a_n
\end{array}\right] +
$$

$$
\left[\begin{array}{cc|cc}
 & & & \\
 & \pm b_m & b_m & \\
\hline
 & b_m & \pm b_m & \\
 & & &
\end{array}\right] = \left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right] + \rho\, u\, u^T \quad (63)
$$

where $\rho = \pm b_m$ and $u = \left[\begin{smallmatrix} \pm e_m \\ e_1 \end{smallmatrix}\right]$, being e_x a column vector with the x-th element set to 1. In such a way the tridiagonal matrix T is divided into two tridiagonal matrices T_1 and T_2 of smaller dimensions. This procedure can be repeatedly applied to finally obtain matrices with suitable dimensions in order to derive the eigenvalues and eigenvectors with a reduced computational effort. Assuming the eigenvalue problem for these matrices is solved, regardless of the method used for this purpose, the following factorization results

$$
T_i = Q_i \Lambda_i Q_i^T,\ Q_i^T Q_i = I,\ i = 1, 2 \quad . \qquad (64)
$$

2.6.2 Conquer Operation

Once the decomposition of T_1 and T_2 are known, the "conquer" operation allows the matrix T to be factorized. To this end combining (63) and (64) we have

$$\left[\begin{array}{c|c} Q_1^T & \\ \hline & Q_2^T \end{array}\right]\left(\left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right] + \rho\, u\, u^T\right)\left[\begin{array}{c|c} Q_1 & \\ \hline & Q_2 \end{array}\right] = \left[\begin{array}{c|c} \Lambda_1 & \\ \hline & \Lambda_2 \end{array}\right] + \rho\, v\, v^T \tag{65}$$

where the vector v is given by

$$v = \left[\begin{array}{c|c} Q_1^T & \\ \hline & Q_2^T \end{array}\right] u = \left[\begin{array}{c} \pm Q_1^T e_m \\ Q_2^T e_1 \end{array}\right] = \left[\begin{array}{c} \pm\, \text{last row of}\, Q_1 \\ \text{first row of}\, Q_2 \end{array}\right] \tag{66}$$

Having derived the matrix (65) the problem is now to decompose it, that is to derive the eigenvalue matrix Λ and the eigenvector matrix Q such that

$$D + \rho\, v\, v^T = Q \Lambda Q^T \tag{67}$$

where $D = \left[\begin{array}{c|c} \Lambda_1 & \\ \hline & \Lambda_2 \end{array}\right]$. Suppose equation (67) is solved, then the decomposition of T is given by

$$T = \left[\begin{array}{c|c} Q_1 & \\ \hline & Q_2 \end{array}\right] Q \Lambda Q^T \left[\begin{array}{c|c} Q_1^T & \\ \hline & Q_2^T \end{array}\right] \tag{68}$$

where Λ and Q are the solution of (67). Solving equation (67) corresponds to solving a new eigenvalue problem for a matrix that is the sum of diagonal matrix D plus a term of rank one.

2.6.3 Reducing the Eigenvalue Problem to a Secular Equation

Determining the eigenvalues and eigenvectors of (65) is equivalent to solve the following equation

$$(D + \rho\, v\, v^T)\, x = \lambda x \tag{69}$$

which can be rewritten as

$$(D - \lambda I)x = -\rho\, v\, v^T\, x \quad . \tag{70}$$

It can be easily shown that the matrix $(D - \lambda I)$ can not be singular, as this condition would imply $(D - \lambda I) = 0$ and, as a consequence, $v = 0$ or $v^T = 0$.

Being $(D - \lambda I)$ non singular it is also invertible so that (70) can be solved giving

$$x = \rho\,(\lambda I - D)^{-1} v\,(v^T x) \quad . \tag{71}$$

Multiplying and dividing (71) for v^T and $v^T x$ respectively (note that $v^T x \neq 0$) yields

$$f(\lambda) = 1 - \rho\, v^T (\lambda I - D)^{-1} v = 1 - \rho \sum_{k=1}^{n} \frac{v_k^2}{\lambda - d_k} = 0 \tag{72}$$

being d_k the diagonal elements of D.

Equation (72) is known as 'secular equation' and its solutions are the eigenvalues of the matrix $D + \rho\, v\, v^T$, thus solving the eigenvalue problem started by (67). The secular equation (72) can be solved by the method called Li's algorithm [9]. Once the eigenvalues $\lambda_k,\ k = 1, \ldots, n$ are achieved by solving (72), the corresponding eigenvectors x are given by (see [10] for details)

$$x = \frac{(\lambda I - D)^{-1} v}{\|(\lambda I - D)^{-1} v\|} \quad . \tag{73}$$

The pseudo-code of the algorithm is reported in Algorithm 8.

Algorithm 8 Divide and conquer

Require: $T \in \mathbb{R}^{n \times n}$ tridiagonal symmetric

Divide T as $T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + \rho u u^T$

for $i = 1, 2$ **do**

Compute Λ_i, Q_i eigenvalues/vectors of T_i as follows:

if T_i suitably small **then**

Compute eigenvalues/vectors directly

else

Apply recursively divide and conquer algorithm to T_i

end if

end for

Use factorized T_1, T_2 to compute $D + \rho v v^T$, where

$$D = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}, v = \begin{bmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{bmatrix} u \tag{74}$$

Compute eigenvalues Λ by solving the secular equation $D + \rho v v^T = Q \Lambda Q^T$ through Li's algorithm

Compute eigenvectors as $\begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} Q$

References

[1] Alan Kaylor Cline and Inderjit S. Dhillon. *Computation of the Singular Value Decomposition.* CRC Press, Jan 2006.

[2] Gene H Golub and Charles F Van Loan. Matrix computations. *Johns Hopkins, Baltimore, MD*, 1983.

[3] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, Apr 1970.

[4] J.H. Wilkinson. Global convergene of tridiagonal QR algorithm with origin shifts. *Linear Algebra and its Applications*, 1(3):409 – 420, 1968.

[5] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific and Statistical Computing*, 11(5):873–912, 1990.

[6] George E. Forsythe and Peter Henrici. The cyclic Jacobi method for computing the principal values of a complex matrix. *Transactions of the American Mathematical Society*, 94:1–23, 1960.

[7] H. F. Kaiser. The JK Method: A Procedure for Finding the Eigenvectors and Eigenvalues of a Real Symmetric Matrix. *The Computer Journal*, 15(3):271–273, 1972.

[8] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM J. Matrix Anal. Appl.*, 16(1):79–92, Jan 1995.

[9] Ren-Cang Li. Solving secular equations stably and efficiently. Technical report, EECS Department, University of California, Berkeley, Dec 1994.

[10] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numerische Mathematik*, 36(2):177–195, Jun 1980.