# Butterfly Matrices and the Modulus Kogbetliantz Method

**Vjeran Hari**[*1]

[1] Department of Mathematics, University of Zagreb, P.O. Box 335, 10002 Zagreb, Croatia.

Received June 26, 2005

In this report we present an adaptation of the Kogbetliantz method for computing the singular value decomposition of triangular matrices to parallel computing. Two novelties are introduced. First, the initial (square or triangular) matrix has to be brought to the special "butterfly" form. Second, the Kogbetliantz method is applied to such a matrix, using the special modulus pivot strategy. This leads to the relatively accurate algorithm amenable for parallel processing. The method can be further modified to work with blocks.



## 0 Introduction

Today the common methods for computing the singular value decomposition (SVD) of general matrices are Divide and Conquer, Differential QD and the QR method. Although fast, they first have to bring the initial matrix to the bidiagonal form. Unfortunately, this initial reduction can deteriorate the relative accuracy of the smallest singular values even in the case when the matrix is well-behaved for the accurate SVD computation. For such matrices, the one-sided Jacobi methods have proved to be very accurate (see [1] or the review paper [2]) and the latest research [3] indicates that they can be made fast. Recently, it has been shown in [10], [7] that some two-sided SVD Jacobi-type methods are also (relatively) accurate.

Here we report that the Kogbetliantz method can be well adapted for parallel processing. We can prove that besides being parallel and simple, it can be made fast and accurate. It essentially preserves the (initially created) zero elements, which has beneficial impact to its simplicity and to the quadratic asymptotic convergence. Even more, the method can be further modified to work with blocks, which leads to better using processor capabilities like efficient using of the cache memory.

In this short communication, we present only the main ideas. In Section 1, we briefly introduce the butterfly form and how to bring the general matrix to it. In Section 2, we shortly describe the method and the aims of the current research.

## 1 Reduction to Butterfly Form

For $n = 6$ and $n = 7$, the butterfly form of $n \times n$ matrix $B$ has the form

$$\begin{array}{cccccc} x & 0 & 0 & 0 & 0 & 0 \\ x & x & 0 & 0 & 0 & x \\ x & x & x & 0 & x & x \\ x & x & x & x & x & x \\ x & x & 0 & 0 & x & x \\ x & 0 & 0 & 0 & 0 & x \end{array} \quad \text{and} \quad \begin{array}{ccccccc} x & 0 & 0 & 0 & 0 & 0 & 0 \\ x & x & 0 & 0 & 0 & 0 & x \\ x & x & x & 0 & 0 & x & x \\ x & x & x & x & x & x & x \\ x & x & x & 0 & x & x & x \\ x & x & 0 & 0 & 0 & x & x \\ x & 0 & 0 & 0 & 0 & 0 & x \end{array} .$$

Here $x$ denotes the elements of $B$ which can be non-zero. It is easy to reduce any rectangular matrix $A$ to this form. The procedure is quite similar to obtaining the QR factorization. Indeed, if $A$ is $m \times n$, $m \geq n$, one can use a finite set of Householder reflectors or the appropriate sequence of Givens rotations, applying them from the left-hand side. This will preserve the relative accuracy of the singular values and vectors in the presence of rounding errors provided that $A$ is well-behaved.

* Corresponding author: e-mail: hari@math.hr, Phone: +385 1 4605 748, Fax: +385 1 4680 335

If $H_i$ and $H_i'$ denote the Householder reflectors (more precisely, the direct sum of the appropriate identity matrices and the Householder reflectors) which are used at step $i$ of the reduction, then the procedure can be described as follows

$$B = \begin{cases} H_k' H_k \cdots H_2' H_2 H_1' H_1\, A, & k = n/2, \;\; n \text{ even} \\ B = H_k' H_k \cdots H_2' H_2 H_1 A\,, & k = (n+1)/2, \;\; n \text{ odd.} \end{cases}$$

Here $B$ is the matrix in butterfly form. Each matrix $H_i$ and $H_i'$ makes an appropriate number of zeroes in the current column. The procedure is illustrated for $m \times n$ matrix $A$ with $m = 7$ and $n = 6\,, 5$. By $*$ are denoted the elements which will become zero in the following step

$$\begin{array}{cccccc} x & x & * & * & x & x \\ x & x & * & * & x & x \\ x & x & x & * & x & x \\ x & x & x & x & x & x \\ x & x & * & * & x & x \\ x & x & * & * & x & x \\ x & x & * & * & x & x \end{array} \;\rightarrow\; \begin{array}{cccccc} x & * & 0 & 0 & * & x \\ x & x & 0 & 0 & * & x \\ x & x & x & 0 & x & x \\ x & x & x & x & x & x \\ x & x & 0 & 0 & x & x \\ x & * & 0 & 0 & * & x \\ x & * & 0 & 0 & * & x \end{array} \;\rightarrow\; \begin{array}{cccccc} x & 0 & 0 & 0 & 0 & * \\ x & x & 0 & 0 & 0 & x \\ x & x & x & 0 & x & x \\ x & x & x & x & x & x \\ x & x & 0 & 0 & x & x \\ x & 0 & 0 & 0 & 0 & x \\ * & 0 & 0 & 0 & 0 & * \end{array} \;\rightarrow\; \begin{array}{cccccc} x & 0 & 0 & 0 & 0 & 0 \\ x & x & 0 & 0 & 0 & x \\ x & x & x & 0 & x & x \\ x & x & x & x & x & x \\ x & x & 0 & 0 & x & x \\ x & 0 & 0 & 0 & 0 & x \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$\begin{array}{ccccc} x & x & * & x & x \\ x & x & * & x & x \\ x & x & x & x & x \\ x & x & * & x & x \\ x & x & * & x & x \\ x & x & * & x & x \\ x & x & * & x & x \end{array} \;\rightarrow\; \begin{array}{ccccc} x & * & 0 & * & x \\ x & x & 0 & * & x \\ x & x & x & x & x \\ x & x & 0 & x & x \\ x & * & 0 & * & x \\ x & * & 0 & * & x \\ x & * & 0 & * & x \end{array} \;\rightarrow\; \begin{array}{ccccc} x & 0 & 0 & 0 & * \\ x & x & 0 & 0 & x \\ x & x & x & x & x \\ x & x & 0 & x & x \\ x & 0 & 0 & 0 & x \\ * & 0 & 0 & 0 & * \\ * & 0 & 0 & 0 & * \end{array} \;\rightarrow\; \begin{array}{ccccc} x & 0 & 0 & 0 & 0 \\ x & x & 0 & 0 & x \\ x & x & x & x & x \\ x & x & 0 & x & x \\ x & 0 & 0 & 0 & x \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\,.$$

The matrix in butterfly form is square, so we have separated it from the remaining zero rows by the line.

Since there exist very efficient implementations of the QR factorization, we shall show how easily $B$ can be obtained from $T$ which is in triangular form. The second reason why preprocessing of the initial matrix by the QR (possibly followed by the LQ) factorization is important, lies in its noticeable diagonalization effect (see [4, 3]). An upper-triangular matrix T can be transformed to the butterfly form using cheap and exact similarity by a permutation matrix $B = P^T T P$. The permutation matrix $P$ can be obtained as a product of transposition matrices,

$$P = \begin{cases} I_{12} I_{13} (I_{14} I_{23}) (I_{15} I_{24}) (I_{16} I_{25} I_{34}) \cdot (I_{1,n} I_{2,n-1} \cdots I_{k,k+1}) & \text{if } n = 2k \\ I_{12} I_{13} (I_{14} I_{23}) (I_{15} I_{24}) (I_{16} I_{25} I_{34}) \cdot (I_{1,n} I_{2,n-1} \cdots I_{k,k+2}) & \text{if } n = 2k+1\,. \end{cases}$$

Here, $I_{pq} = [e_1, \ldots, e_q, \ldots, e_p, \ldots, e_n]$ where $e_j$, $1 \le j \le n$ are the columns of the identity matrix $I_n$. For example, for $n = 6$ we have $k = n/2 = 3$ and the procedure takes the form

$$\begin{array}{cccccc} x & * & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \end{array} \;\rightarrow\; \begin{array}{cccccc} x & 0 & * & x & x & x \\ x & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \end{array} \;\rightarrow\; \begin{array}{cccccc} x & 0 & 0 & * & x & x \\ x & x & * & x & x & x \\ x & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \end{array}$$

$$\rightarrow\; \begin{array}{cccccc} x & 0 & 0 & 0 & * & x \\ x & x & 0 & * & x & x \\ x & x & x & x & x & x \\ x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \end{array} \;\rightarrow\; \begin{array}{cccccc} x & 0 & 0 & 0 & 0 & * \\ x & x & 0 & 0 & * & x \\ x & x & x & * & x & x \\ x & x & 0 & x & x & x \\ x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \end{array} \;\rightarrow\; \begin{array}{cccccc} x & 0 & 0 & 0 & 0 & 0 \\ x & x & 0 & 0 & 0 & x \\ x & x & x & 0 & x & x \\ x & x & x & x & x & x \\ x & x & 0 & 0 & x & x \\ x & 0 & 0 & 0 & 0 & x \end{array}$$

The effect of this transformation can be viewed in the following example

$$\begin{array}{cccccccc} 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\ 0 & 22 & 23 & 24 & 25 & 26 & 27 & 28 \\ 0 & 0 & 33 & 34 & 35 & 36 & 37 & 38 \\ 0 & 0 & 0 & 44 & 45 & 46 & 47 & 48 \\ 0 & 0 & 0 & 0 & 55 & 56 & 57 & 58 \\ 0 & 0 & 0 & 0 & 0 & 66 & 67 & 68 \\ 0 & 0 & 0 & 0 & 0 & 0 & 77 & 78 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 88 \end{array} \;\longrightarrow\; \begin{array}{cccccccc} 88 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 68 & 66 & 0 & 0 & 0 & 0 & 0 & 67 \\ 48 & 46 & 44 & 0 & 0 & 0 & 45 & 47 \\ 28 & 26 & 24 & 22 & 0 & 23 & 25 & 27 \\ 18 & 16 & 14 & 12 & 11 & 13 & 15 & 17 \\ 38 & 36 & 34 & 0 & 0 & 33 & 35 & 37 \\ 58 & 56 & 0 & 0 & 0 & 0 & 55 & 57 \\ 78 & 0 & 0 & 0 & 0 & 0 & 0 & 77 \end{array}\,.$$

## 2 Kogbetliantz Method under the Modulus Strategy

Kogbetliantz method is a known two-sided Jacobi-type method for computing the SVD of a square matrix. Since its discovery [8, 9] the method has been studied with respect to its global and quadratic convergence and implementation details. Later, it has been discovered [4] that it becomes simpler and more efficient if it is applied to triangular matrices under the serial (the row- or column-cyclic) pivot stragies. Then it behaves similar to the standard Jacobi method for symmetric matrices. Its relative accurcy has been recently proved [7, 10]. Here we show that if the method is applied to the matrix in butterfly form under the modulus pivot strategy, all the good properties of the serial Kogbetliantz method for triangular matrices are preserved. In addition, it becoemes amenable for parallel computing. This method will be referred here as KMMS.

Starting from the matrix $B$ in butterfly form, KMMS generates the sequence of matrices $B^{(k)} = (b_{lm}^{(k)})$, $k \geq 0$, using the rule

$$B^{(k+1)} = (U^{(k)})^T B^{(k)} V^{(k)}, \quad k \geq 0, \quad B^{(0)} = B\,.$$

Here $U^{(k)}$ and $V^{(k)}$, are simple rotations in the $(i,j)$-plane, where $i = i(k)$, $j = j(k)$, $i(k) < j(k)$ is the pivot pair. If $b_{ij}^{(k)} \neq 0$ or $b_{ji}^{(k)} \neq 0$ the rotation angles $\varphi^{(k)}$, $\psi^{(k)}$ of $U^{(k)}$, $V^{(k)}$, respectively, are determined from the requirement

$$b_{ij}^{(k+1)} = b_{ji}^{(k+1)} = 0, \quad \text{which implies} \quad \|\Omega(B^{(k+1)})\|^2 = \|\Omega(B^{(k)})\|^2 - |b_{ij}^{(k)}|^2 - |b_{ji}^{(k)}|^2.$$

Here generally, $\Omega(X) = X - \mathrm{diag}(X)$ denotes the off-diagonal part of $X$ and $\|X\|$ is the Frobenius norm of $X$. It can be shown, as in [7], that all the elements of the left-hand rotation $U^{(k)}$ and the right-hand $V^{(k)}$ can be computed with very small relative errors.

### 2.1 The Modulus Pivot Strategy

This pivot strategy was first described in [4] and the name is first mentioned in [11]. For simplicity, we can look at it as to a special cyclic (pivot) strategy. The modulus ordering of the set $\mathbf{P}_n = \{\,(p,q)\;:\;1 \leq p < q \leq n\,\}$ which defines the strategy, will be depicted using the upper-triangular part of the matrix of order $n = 7$. The displayed numbers denote the ordering (and position) of the pivot elements within one cycle. By $\mathcal{S}_t$, $1 \leq t \leq 7$ are denoted the so called rotation sets, which are the sets of pivot pairs of the elements which are simultaneously annihilated in the current “parallel step”. This means that, using three processors, all the elements $b_{ij}$, $b_{ii}$, $b_{jj}$, $(i,j)$ of $\mathcal{S}_t$ can be fetched simultaneously. In the right-most matrix, we have denoted by the same number, those pivot positions whose elements are simultaneously annihilated. In another words, these numbers denote the schedule of the parallel steps and the positions of the pivot elements in these parallel steps.

$$\begin{array}{ccccccc}
\cdot & 6 & 9 & 11 & 14 & 16 & 19 \\
 & \cdot & 12 & 15 & 17 & 20 & 1 \\
 & & \cdot & 18 & 21 & 2 & 4 \\
 & & & \cdot & 3 & 5 & 7 \\
 & & & & \cdot & 8 & 10 \\
 & & & & & \cdot & 13 \\
 & & & & & & \cdot
\end{array}
\qquad
\begin{array}{l}
\mathcal{S}_1 = \{(2,7),(3,6),(4,5)\} \\
\mathcal{S}_2 = \{(3,7),(4,6),(1,2)\} \\
\mathcal{S}_3 = \{(4,7),(5,6),(1,3)\} \\
\mathcal{S}_4 = \{(5,7),(1,4),(2,3)\} \\
\mathcal{S}_5 = \{(6,7),(1,5),(2,4)\} \\
\mathcal{S}_6 = \{(1,6),(2,5),(3,4)\} \\
\mathcal{S}_7 = \{(1,7),(2,6),(3,5)\}
\end{array}
\qquad
\begin{array}{ccccccc}
\cdot & 2 & 3 & 4 & 5 & 6 & 7 \\
 & \cdot & 4 & 5 & 6 & 7 & 1 \\
 & & \cdot & 6 & 7 & 1 & 2 \\
 & & & \cdot & 1 & 2 & 3 \\
 & & & & \cdot & 3 & 4 \\
 & & & & & \cdot & 5 \\
 & & & & & & \cdot
\end{array}$$

For example, in the third parallel step the following work is done: first the elements of the rotation matrices which annihilate the elements at positions $(4,7)$, $(5,6)$ and $(1,3)$ are simultaneously computed. Then all three processors simultaneously apply the right-hand transformations. Finally, all the processors simultaneously apply the left-hand transformations. (If the left and the right singular vectors are wanted, then the left-hand rotations are accumulated while the right singular vectors are easily computed a posteriori, like in [3]). Many ideas and implementation details from [3] can be used here.

KMMS is well suited for the computer architectures with the common memory. Note however, that the right-hand transformations can be several times faster then the left-hand ones. In our situation, this overhead cost can be almost eliminated using some efficient algorithm for transposing the matrix. So, the last two tasks can be replaced by: simultaneously apply the right-hand transformations, transpose the matrix, simultaneously apply the (originally) left-hand transformations from the right-hand side. In this approach only a bit of care should be taken when updating the left singular vector matrix.

### 2.2 Using the Butterfly Structure

If $B$ is in butterfly form, then it is permutationally similar to an upper-triangular matrix (we say that $B$ is PST), so it is also essentially triangular (ET) since $b_{pq}b_{qp} = 0$, $p < q$ holds. The butterfly matrix form and the modulus strategy go together since all generated matrices by the algorithm are PST. Indeed, the following figure depicts what happens with the matrix structure during the two consecutive cycles

$$
\begin{array}{cccccc}
\bullet & \cdot & \cdot & \cdot & \cdot & \cdot \\
\bullet & \bullet & \cdot & \cdot & \cdot & \bullet \\
\bullet & \bullet & \bullet & \cdot & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \cdot & \cdot & \bullet & \bullet \\
\bullet & \cdot & \cdot & \cdot & \cdot & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \cdot & \cdot & \cdot & \cdot & \cdot \\
\bullet & \bullet & \cdot & \cdot & \cdot & \bullet \\
\bullet & \bullet & \bullet & \cdot & \bullet & \bullet \\
\bullet & \bullet & * & \bullet & \bullet & \bullet \\
\bullet & * & \cdot & \cdot & \bullet & \bullet \\
* & \cdot & \cdot & \cdot & \cdot & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \cdot & \cdot & \cdot & \cdot & \cdot \\
\bullet & \bullet & \cdot & \cdot & \cdot & * \\
\bullet & \bullet & \bullet & \cdot & * & \bullet \\
\bullet & \bullet & \cdot & \bullet & \bullet & \bullet \\
\bullet & \cdot & \cdot & \cdot & \bullet & \bullet \\
\cdot & \cdot & \cdot & \cdot & \cdot & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \cdot & \cdot & \cdot & \cdot & \cdot \\
* & \bullet & \cdot & \cdot & \cdot & \cdot \\
\bullet & \bullet & \bullet & \cdot & \cdot & * \\
\bullet & \bullet & \bullet & \bullet & * & \bullet \\
\bullet & \bullet & \cdot & \cdot & \bullet & \bullet \\
\bullet & \cdot & \cdot & \cdot & \cdot & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \bullet & \cdot & \cdot & \cdot & \cdot \\
* & \bullet & \bullet & \cdot & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \cdot & * \\
\bullet & \bullet & \bullet & \cdot & \bullet & \bullet \\
\bullet & \bullet & \cdot & \cdot & \cdot & \bullet
\end{array}
$$

$$
\begin{array}{cccccc}
\bullet & \bullet & \cdot & \cdot & \cdot & \cdot \\
\cdot & \bullet & \cdot & \cdot & \cdot & \cdot \\
\cdot & * & \bullet & \cdot & \cdot & \cdot \\
* & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & * \\
\bullet & \bullet & \bullet & \cdot & \cdot & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \bullet & \bullet & \cdot & \cdot & \cdot \\
\cdot & \bullet & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \bullet & \cdot & \cdot & \cdot \\
\cdot & * & \bullet & \bullet & \cdot & \cdot \\
* & \bullet & \bullet & \bullet & \bullet & \cdot \\
\bullet & \bullet & \bullet & \bullet & \cdot & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\cdot & \bullet & \bullet & \cdot & \cdot & \cdot \\
\cdot & \cdot & \bullet & \cdot & \cdot & \cdot \\
\cdot & \cdot & * & \bullet & \cdot & \cdot \\
\cdot & * & \bullet & \bullet & \bullet & \cdot \\
* & \bullet & \bullet & \bullet & \bullet & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \bullet & \bullet & \bullet & \cdot & \cdot \\
\cdot & \cdot & \bullet & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \bullet & \cdot & \cdot \\
\cdot & \cdot & * & \bullet & \bullet & \cdot \\
\cdot & * & \bullet & \bullet & \bullet & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & * & \bullet & \bullet & \bullet & \bullet \\
\cdot & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \cdot & \bullet & \bullet & \cdot & \cdot \\
\cdot & \cdot & \cdot & \bullet & \cdot & \cdot \\
\cdot & \cdot & \cdot & * & \bullet & \cdot \\
\cdot & \cdot & * & \bullet & \bullet & \bullet
\end{array}
$$

$$
\begin{array}{cccccc}
\bullet & \cdot & * & \bullet & \bullet & \bullet \\
\cdot & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \cdot \\
\cdot & \cdot & \cdot & \bullet & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \bullet & \cdot \\
\cdot & \cdot & \cdot & * & \bullet & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \cdot & \cdot & * & \bullet & \bullet \\
\bullet & \bullet & * & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \cdot & \bullet & \bullet & \cdot \\
\cdot & \cdot & \cdot & \cdot & \bullet & \cdot \\
\cdot & \cdot & \cdot & \cdot & * & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \cdot & \cdot & \cdot & * & \bullet \\
\bullet & \bullet & \cdot & * & \bullet & \bullet \\
\bullet & \cdot & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \cdot & \bullet & \bullet & \bullet \\
\cdot & \cdot & \cdot & \cdot & \bullet & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \cdot & \cdot & \cdot & \cdot & * \\
\bullet & \bullet & \cdot & \cdot & * & \bullet \\
\bullet & \bullet & \bullet & * & \bullet & \bullet \\
\bullet & \cdot & \cdot & \bullet & \bullet & \bullet \\
\cdot & \cdot & \cdot & \cdot & \bullet & \bullet \\
\cdot & \cdot & \cdot & \cdot & \cdot & \bullet
\end{array}
\rightarrow
\begin{array}{cccccc}
\bullet & \cdot & \cdot & \cdot & \cdot & \cdot \\
\bullet & \bullet & \cdot & \cdot & \cdot & * \\
\bullet & \bullet & \bullet & \cdot & * & \bullet \\
\bullet & \bullet & \cdot & \bullet & \bullet & \bullet \\
\bullet & \cdot & \cdot & \cdot & \bullet & \bullet \\
\cdot & \cdot & \cdot & \cdot & \cdot & \bullet
\end{array}
$$

Note that we have used the preparatory "step zero", only in the first cycle, which annihilates the elements at positions $(6,1)$, $(5,2)$ i $(4,3)$ (which belong to the rotation set $\mathcal{S}_n$). By $\bullet$ are denoted all possibly nontrivial elements, by $\cdot$ the null-elements and by $*$ the pivot elements (which become zero). So, each cycle incorporates 7 parallel steps. More generally, when $n$ is odd, all the rotation sets contain the same number ($[\frac{n}{2}]$) of elements. For even $n$ they contain by turns $[\frac{n}{2}]$ and $[\frac{n-1}{2}]$ elements.

It can be shown (see [4]) that each iterate $B^{(k)}$ obtained after some parallel step is PST. Therefore it is ET and all the work can be done in the upper-triangle of the upper-triangular matrix $G^{(k)}$ which satisfies $G^{(k)} + G^{(k)T} = B^{(k)} + B^{(k)T}$, $k \geq 0$.

Our current research is concentrated on several aspects of KMMS: to prove the relative accuracy of the method, to generalize the method so that it works with blocks, and to prove that the block version of the method is also relatively accurate. The relative accuracy proof is closely related to the proofs given in [7] and [10]. The derivation of the block version of the method is related with the ideas described in [5] and [6]. Working with blocks means that KMMS becomes BLAS 3 algorithm which better exploits possibilities of each processor.

Other aspects, such as convergence and the overall efficiency of the methods on multi-processor computers will hopefully be studied later.

## References


[1] J. Demmel and K. Veselić, Jacobi's method is more accurate than QR. SIAM J. Matrix Anal. Appl. 13, 1204–1245 (1992).

[2] Z. Drmač, V. Hari, I. Slapničar, Advances in Jacobi methods. Proccedings of the Conference on Applied and computational mathematics, Dubrovnik 2002 (Kluwer 2003).

[3] Z. Drmač and K. Veselić, New fast and accurate Jacobi SVD algorithm. Part one. Preprint (2004).

[4] V. Hari and K. Veselić, On Jacobi methods for singular value decompositions. SIAM J. Sci. Stat. Comput. Vol. 8, No. 5, 741–754 (1987).

[5] V. Hari, On Some New Applications of the CS Decomposition. ICNAAM 2004: Extended Abstracts (Editor T. E. Simos and Ch. Tsitouras) ISBN: 3-527-40563-1 Hardcover.

[6] V. Hari, Accelerating the SVD Block-Jacobi Method. To appear in Computing.

[7] V. Hari and J. Matejaš, Accuracy of the Kogbetliantz method. Preprint (2005).

[8] E. Kogbetliantz, Diagonalization of General Complex Matrices as a New Method for Solution of Linear Equations, Proc. Imtern. Congr. Math. Amsterdam 2, 356–357 (1954).

[9] E. Kogbetliantz, Solutions of Linear Equations by Diagonalization of Coefficient Matrices, Quart. Appl. Math. 13, 123–132 (1955).

[10] T. Londre and N. H. Rhee, Numerical Stability of the Parallel Jacobi Method. To appear in SIAM J. MAA.

[11] F.T. Luk and H. Park, On parallel Jacobi orderings. SIAM J. Sci. Statist. Comput. 10, 18–26 (1989).