Proceedings of the International Multiconference on Computer Science and Information Technology pp. 343–348
ISBN 978-83-60810-27-9
ISSN 1896-7094

Parallelization of SVD of a Matrix-Systolic Approach

Halil Snopce
South East Europian University, Ilindenska bb., 1200 Tetovo, Republic of Macedonia
Email: h.snopce@seeu.edu.mk

Ilir Spahiu
Pedagogical Faculty "Kliment Ohridski", bul. 'Krste Misirkov' bb. 1000 Skopje, Republic of Macedonia
Email: i.spahiu@seeu.edu.mk

***Abstract*—In this paper we investigate the parallelization of Hestenes-Jacobi method for computing the SVD of an mxn matrix using systolic arrays. In the case of real matrix an array of R^2 processors is proposed, such that each row contains n columns. In order to extend this idea we have presented three transformations which are used for transforming the complex into the real matrix. After the additional computations, we show how the same array may be used for the SVD of a complex matrix.**

I. Introduction

THE Singular Value Decomposition (SVD) is a matrix decomposition of a great importance in different engineering applications. It is particularly useful in the areas of signal and image processing, robotics, pattern recognition etc. This decomposition can be used for determining the rank of a matrix in numerically reliable manner [2,7]. Different approaches are known in the literature for the parallel computation of the SVD of a matrix where the elements are meanly real numbers. In [5], an expandable array for parallel computing of SVD of large matrices is proposed. In [11], the use of CORDIC method for SVD is demonstrated. In [14] is given a fast Jacobi-like algorithm for the parallel solution of the SVD not focusing in CORDIC form, but by applying approximate rotations. In some applications one needs to use complex matrices as well (like in beam-forming algorithms in the signal processing [8]). In this paper we propose an approach which offers parallelization using systolic arrays. First we give the method of constructing the corresponding systolic array where the matrix is real, and then we analyze how the corresponding systolic array for the SVD can be designed if the elements of the matrix are complex numbers. This differs from the case of real matrix, because of the fact that complex arithmetic and matrix transformations in this arithmetic require greater number of computational steps. In [9] it is proposed the SVD of a matrix with complex elements where the matrix is with special structure. On the other hand in [10] this idea is extended for an arbitrary complex 2x2 matrix. In [12] is presented a systolic design concept for iterative algorithms when the VLSI design keeps evolving into nanoscale. In [15] is designed the parallelism of the so called Hestenes-Jacobi method for the SVD using the ring array. A systolic array for the computation of the SVD is presented in [5]. The array uses $(n/2)^2$ processors and is capable of processing the SVD of a square matrix. The time complexity is $O(n \log n)$, which is to be compared with the best serial algorithms, which have a $O(n^3)$ time complexity. In [13,16] the efficiency of this array is improved. In this paper it is given a model of systolic array for the SVD of an mxn matrix which can be found with the Hestenes-Jacobi method. Then, in the case of a complex matrix, there are given some transformations which transform the complex into the real matrix. Using these transformations the same systolic array can be used for the SVD of complex matrix.

II. The SVD of a Matrix

The SVD of an $m \times n$ matrix A is given by:

$$A = U\Sigma V^T =$$

$$\begin{bmatrix} u_1 & u_2 & \dots & u_{m-1} & u_m \end{bmatrix}_{m\times m} \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \dots & & & \\ 0 & 0 & \sigma_r & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}_{m\times n} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_{n-1}^T \\ v_n^T \end{bmatrix}_{n\times n} \quad (1)$$

where U and V are orthogonal $m \times m$ and $n \times n$ matrices respectively, (i.e., $U^T U = I_m$ and $VV^T = I_n$) and Σ is a diagonal $m \times n$ matrix such that $\Sigma = diag(\sigma_1, \sigma_2, \dots)$; $\sigma_1 \ge \sigma_2 \ge \dots \ge \sigma_r \ge \sigma_{r+1} = \dots = \sigma_k = 0$ where $r = rankA$. In the case $m=n,\ \Sigma$ is square diagonal matrix of order n . In the definition given above the σ_i -s are the singular values of A .

The SVD decomposition is often based on diagonalizing rotations which are orthogonal transformations which preserve Eigenvalues and Eigenvectors as well as singular values and singular vectors. The sequence of rotations A_k , such that $\lim_{k\to\infty} A_k = \Sigma$, is applied during the process.

III. Jacobi Rotations

The classical Jacobi rotation [1] has been used by Jacobi in 19th century as a tool for solving the least square problem. This rotation is also called Given's rotation. This method uses a sequence of plane rotations to diagonalize a symmetric $n \times n$ real matrix A. We denote the Jacobi rotation of an angle θ in the (i, j) plane by $J(i, j, \theta)$. This is a square matrix equal to the identity matrix except the four additional elements in the intersection of i-th and j-th rows and columns:

$$J(i,j,\theta)=\begin{array}{c c} & \begin{array}{ccccccc} & & i & & j & & \end{array} \\ \begin{array}{c} \\ \\ i \\ \\ j \\ \\ \\ \end{array} & \begin{bmatrix} 1 & & & & & & \\ & \cdots & & & & & \\ & & c & \cdots & s & & \\ & & & \cdots & & & \\ & & -s & \cdots & c & & \\ & & & & & \cdots & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

where $c = \cos\theta$ and $s = \sin\theta$. It's not difficult to verify that this is an orthogonal transformation because of the fact that $J(i, j,\theta)^T J(i, j,\theta) = I$ for each θ. The rotation angle θ is chosen such that to annihilate the $n(n-1)$ off-diagonal elements of the given matrix. Each Jacobi transformation brings the matrix A closer to the diagonal form [2].

In the two sided Jacobi method for the SVD decomposition of a nonsymmetric matrix, the annihilation of the off-diagonal elements is done using a pair of rotations [3], such that:

$$\begin{bmatrix} a_{ii}^{k+1} & 0 \\ 0 & a_{jj}^{k+1} \end{bmatrix} = \begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix}^T \cdot \begin{bmatrix} a_{ii}^k & a_{ij}^k \\ a_{ji}^k & a_{jj}^k \end{bmatrix} \cdot \begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix} \quad (2)$$

If θ_1 and θ_2 are the angles generating the pairs (s_1, c_1) and (s_2, c_2) then the solution given in [3] is:

$$tg(\theta_1+\theta_2) = \frac{a_{ji}^k + a_{ij}^k}{a_{jj}^k - a_{ii}^k}, \quad tg(\theta_2-\theta_1) = \frac{a_{ji}^k - a_{ij}^k}{a_{jj}^k + a_{ii}^k} \quad (3)$$

IV. Hestenes-Jacobi Method

Computationally better results offers the so called Hestenes-Jacobi method [4]. This method works with a smaller unit of computation (only with the rows of the matrix) compared with the first explained Jacobi method (which modifies both the rows and columns). Since Hestenes-Jacobi transformations are orthogonal transformations which leave singular values unchanged, it is not important how many such transformations are applied. For a given $m \times n$ matrix A the Hestenes-Jacobi method produces an orthogonal matrix U as a product of plane rotations and a matrix N which rows are orthogonal.

$$UA = N = [\eta_1 \quad \eta_2 \quad \ldots \quad \eta_n];\ \eta_i^T \eta_j = 0 \ \text{ for } i \neq j \quad (4)$$

The matrix N can be normalized by computing the square of the row norms $\sigma_i = \eta_i^T \eta_i$ and writing: $N = \Sigma\Sigma^{-1}N = \Sigma V$ (in this case the computation of V is done by dividing the element η_i of N by $\sigma_i = \eta_i^T \eta_i$. Because of $\eta_i^T \eta_j = 0$ for $i \neq j$, the matrix Σ is a diagonal matrix). The matrix V is orthogonal and the σ_i are the non-negative squares of the singular values. Because U is orthogonal we have:

$$UA = \Sigma V \Rightarrow A = U^T \Sigma V \quad (5)$$

This equation is the same as the equation of the SVD of the matrix A. Multiplying both sides of (5) by A^T we have:

$$A^T A = (U^T \Sigma V)^T U^T \Sigma V = V^T \Sigma^T U U^T \Sigma V = V^T \Sigma^T \Sigma V$$

This means that Hestenes computation mathematically is equivalent to the Jacobi method applied to $A^T A$ where singular values are equal to the squares of σ_i.

The orthogonal matrix U may be taken as a plane rotation matrix $J(i, j, \theta)$. If $A_1 = A$, then $A_{k+1} = A_k J$. The iterations result in the matrix N defined by (4). Working with a submatrix of order 2×2 we have:

$$\begin{aligned} \begin{bmatrix} a_i^{k+1} & a_j^{k+1} \end{bmatrix} &= \begin{bmatrix} a_i^k & a_j^k \end{bmatrix} \cdot \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \\ &= \begin{bmatrix} a_i^k \cdot c + a_j^k \cdot s & -a_i^k \cdot s + a_j^k \cdot c \end{bmatrix} \end{aligned} \quad (6)$$

From the condition of orthogonality we have:

$$(a_i^k \cdot c + a_j^k \cdot s)^T \cdot (-a_i^k \cdot s + a_j^k \cdot c) = 0 \quad (7)$$

From these relations the iteration formulae for updating the value of an inner product is given by the relation bellow:

$$\begin{aligned} &(a_i^{k+1})^T \cdot a_j^{k+1} = (c^2 - s^2)(a_i^k)^T \cdot a_j^k + cs\left[(a_i^k)^T a_i^k - (a_j^k)^T a_j^k\right] \\ &\Leftrightarrow (a_i^{k+1})^T \cdot a_j^{k+1} = (c^2 - s^2)(a_i^k)^T \cdot a_j^k + cs\left[\left\|a_i^k\right\|^2 - \left\|a_j^k\right\|^2\right] \end{aligned} \quad (8)$$

Writing $g_{ij}=\left(a_i\right)^T \cdot a_j$, the following equality will be fulfilled:

$$\left(c^2-s^2\right) g_{ij}+cs\left[\|a_i^k\|^2-\|a_j^k\|^2\right]=0 \qquad (9)$$

From the equality (9) we can obtain the relation:

$$\lambda=ctg\, 2\theta=\frac{\cos 2\theta}{\sin 2\theta}=\frac{\|a_j^k\|^2-\|a_i^k\|^2}{2g_{ij}} \qquad (10)$$

Putting $t=tg\,\theta$, to the relation $tg\, 2\theta=\frac{2tg\,\theta}{1-tg^2\theta}$ there will be obtained the new relation $\frac{1}{\lambda}=\frac{2t}{1-t^2}$ which is equivalent to the quadratic equation $t^2+2\lambda t-1=0$. One solution for t and then for c and s is given by the relation below:

$$t=-\operatorname{sgn}\lambda\left(|\lambda|+\sqrt{1+\lambda^2}\right) \text{ and } c=\frac{1}{\sqrt{1+t^2}} \;;\; s=ct \qquad (11)$$

V. Systolic Array for the SVD of a Matrix

We present a systolic array consisting of 9 processors (generally the number of processors is RxR). The systolic array first computes the values $g_{ij}=\left(a_i\right)^T \cdot a_j$. After receiving the row norms $\|a_i\|^2$ and $\|a_j\|^2$, each processor computes the rotation values s_{ij} and c_{ij} according to the formulas (8) and (10). The initial position of the corresponding systolic array is given in the figure below:

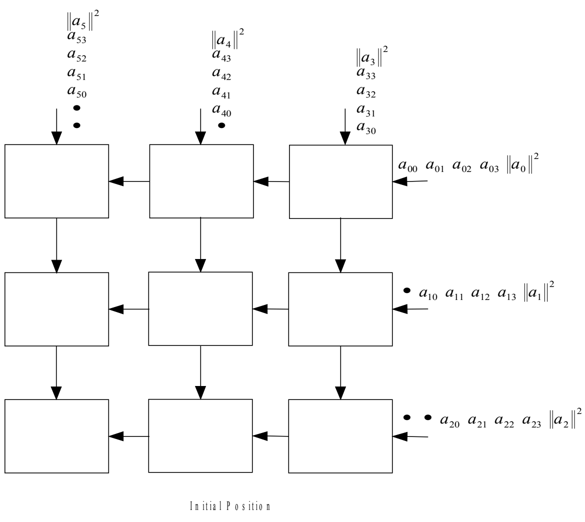


Fig. 1 Systolic computation of rotation angles according (11) for R=3 and n=4

In Fig. 1 it is shown the first step of the systolic array for computing the SVD of a matrix of order mxn $(m=6,\ n=4)$. After 13 steps (the data movement proceeds systolically such that the first three rows move horizontally from right to left and the last three rows move vertically from top to bottom) the values c_{ij} and s_{ij} are computed. The processor that receives row i from the right and row j from the top computes c_{ij} and s_{ij} . Then, the same array may be used for obtaining the correspondent values of the matrix A_{k+1} according to the relation (6). This is the second part of the systolic array. In the same way as in the previous case, the processor receives values from right and top. Then it uses the calculated values for c_{ij} and s_{ij} , and the generalized form of the relation (6) to compute the values of a^{k+1} . The initial form of this array is given in the Fig. 2.

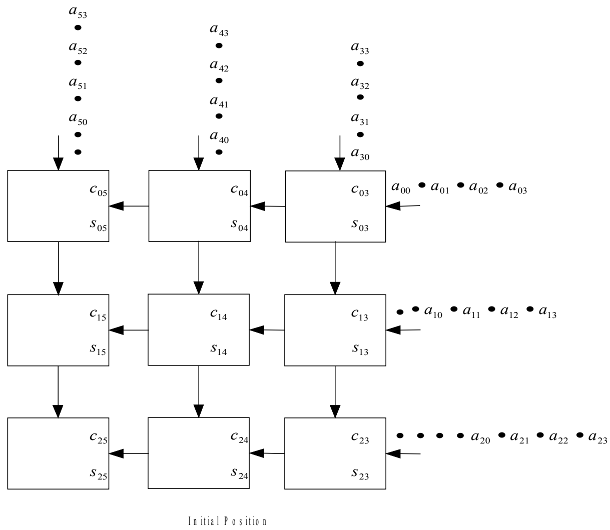


Fig. 2 Obtaining the values of A_{k+1} using the values c_{ij} and s_{ij} computed in fig. 1

In the presented cases there are required 13 time steps for computing the values of c_{ij} and s_{ij} (Fig.1), as well as 15 time steps in the calculation of the values of A_{k+1} (Fig. 2). In general, in the case of Fig.1 there are required $n+1$ time steps for the computation of the values of inner product $g_{ij}=\left(a_i\right)^T \cdot a_j$. According to (11) there are used 4 time steps for the computation (one step for each of factors λ , c , t , s). We have to take into the consideration that this computation will start 2(R-1) time steps later. So the total number of time steps in the case of fig. 1 is n +2R+3. In the case of Fig. 2 there are used more number of steps because of the use of two multiplicative and additive operations. The total number is 2(n +R)+1. In the case presented in Fig. 2 these operations are merged together.

Therefore the exact number of time steps will be twice bigger. So, the total number of time steps will be n+2R+3+4(n+R)+1=5n+6R+4.

VI. SVD OF A COMPLEX MATRIX

The polar representation of each complex number $z = a + ib$ may be written in the form $z = R_z e^{i\theta_z}$, where

$$e^{i\theta} = \cos\theta + i\sin\theta, \quad R_z = \sqrt{a^2+b^2},$$

$$\theta_z = \tan^{-1}\left(\frac{b}{a}\right) \text{ and } 0 \le \theta_z \le 2\pi$$

Let

$$\mathrm{M} = \begin{bmatrix} a_{11}+ib_{11} & a_{12}+ib_{12} \\ a_{21}+ib_{21} & a_{22}+ib_{22} \end{bmatrix} = \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \qquad (12)$$

be a 2x2 complex matrix. There are proposed several unitary transformations for the diagonalization of a complex 2x2 matrix. The two-sided unitary transformation proposed in [3] is:

$$\begin{bmatrix} c_\phi e^{i\theta_\alpha} & -s_\phi e^{i\theta_\beta} \\ s_\phi e^{i\theta_\gamma} & c_\phi e^{i\theta_\delta} \end{bmatrix} \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \begin{bmatrix} c_\psi e^{i\theta_\xi} & -s_\psi e^{i\theta_\eta} \\ s_\psi e^{i\theta_\mu} & c_\psi e^{i\theta_\omega} \end{bmatrix} = \begin{bmatrix} We^{i\theta_\omega} & 0 \\ 0 & Ze^{i\theta_z} \end{bmatrix} \qquad (13)$$

where

$$\begin{aligned}
\tan(\theta_\alpha - \theta_\beta) &= -\frac{AC\sin(\theta_a - \theta_c) + BD(\theta_b - \theta_d)}{AC\cos(\theta_\alpha - \theta_\beta) + BD\cos(\theta_b - \theta_d)} \\
\tan(\theta_\eta - \theta_\omega) &= -\frac{AB\sin(\theta_a - \theta_b) + CD(\theta_c - \theta_d)}{AB\cos(\theta_a - \theta_b) + CD\cos(\theta_c - \theta_d)} \qquad (14) \\
\tan(\theta_\phi - \theta_\psi) &= -\frac{Be^{i(\theta_\alpha+\theta_\omega+\theta_b)} - Ce^{i(\theta_\beta+\theta_\eta+\theta_c)}}{De^{i(\theta_\beta+\theta_\omega+\theta_d)} - Ce^{i(\theta_\alpha+\theta_\eta+\theta_a)}} \\
\tan(\theta_\phi + \theta_\psi) &= -\frac{Be^{i(\theta_\alpha+\theta_\omega+\theta_b)} - Ce^{i(\theta_\beta+\theta_\eta+\theta_c)}}{De^{i(\theta_\beta+\theta_\omega+\theta_d)} - Ce^{i(\theta_\alpha+\theta_\eta+\theta_a)}}
\end{aligned}$$

If M is a real matrix then conditions given above can be simplified taking:

$$\theta_\alpha = \theta_\beta = \theta_\gamma = \theta_\delta = 0$$

The obtained result is identical to the two by two transformation given in (3).

Another method is proposed in [6]. This method is given by the two-sided unitary transformation:

$$\begin{bmatrix} \chi^{\frac{1}{2}} c_\xi & -\chi^{\frac{1}{2}} s_\xi \\ -\chi^{\frac{1}{2}} s_{\xi'} & \chi^{\frac{1}{2}} c_{\xi'} \end{bmatrix} \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \begin{bmatrix} \chi^{\frac{1}{2}} c_z & -\chi^{\frac{1}{2}} s_{z'} \\ \chi^{\frac{1}{2}} s_z & \chi^{\frac{1}{2}} c_{z'} \end{bmatrix} = \begin{bmatrix} We^{i\theta_\omega} & 0 \\ 0 & Ze^{i\theta_z} \end{bmatrix} \qquad (15)$$

where

$$\begin{aligned}
& z = \psi + i\phi, \; z' = \psi - i\phi, \\
& \xi = \varepsilon + i\eta, \; \xi' = \varepsilon - i\eta, \\
& \chi = \sec h2\phi
\end{aligned}$$

Transformations in (15) are unitary because the fact that $\chi(c^2+s^2)=1$.

The main objective in the ordering the SVD of a complex matrix is transforming that into the real form. For that purpose some special forms of transformations are given which will help in that contest.

Transformation 1: The transformation that converts two elements of a complex 2x2 matrix into real values (two elements of a second row) is given by:

$$\begin{bmatrix} e^{i\theta_\alpha} & 0 \\ 0 & e^{i\theta_\alpha} \end{bmatrix} \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \begin{bmatrix} e^{i\theta_\beta} & 0 \\ 0 & e^{-i\theta_\beta} \end{bmatrix} = \begin{bmatrix} Ae^{i\theta_\omega} & Be^{i\theta_x} \\ C & D \end{bmatrix} \qquad (16)$$

where

$$\theta_\alpha = -\frac{\theta_d + \theta_c}{2} \text{ and } \theta_\beta = \frac{\theta_d - \theta_c}{2}$$

Proof: The left side of (16) is equal with:

$$\begin{bmatrix} A\cdot e^{i(\theta_\alpha+\theta_a+\theta_\beta)} & B\cdot e^{i(\theta_\alpha+\theta_b-\theta_\beta)} \\ C\cdot e^{i(\theta_\alpha+\theta_c+\theta_\beta)} & D\cdot e^{i(\theta_\alpha+\theta_d-\theta_\beta)} \end{bmatrix}$$

Summing and subtracting the equalities $2\theta_\alpha = -\theta_d - \theta_c$ and $2\theta_\beta = \theta_d - \theta_c$ we have: $\theta_\alpha + \theta_\beta + \theta_c = 0$ and $\theta_\alpha + \theta_d - \theta_\beta = 0$. Hence, the two elements of the second row will be real numbers.

Similarly there can be obtained two other transformations given below:

Transformation 2: The transformation that converts two elements of a complex 2x2 matrix into real values (two elements of a second column) is given by:

$$\begin{bmatrix} e^{i\theta_\alpha} & 0 \\ 0 & e^{-i\theta_\alpha} \end{bmatrix}\begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix}\begin{bmatrix} e^{i\theta_\beta} & 0 \\ 0 & e^{i\theta_\beta} \end{bmatrix} = \begin{bmatrix} Ae^{i\theta_\omega} & B \\ Ce^{i\theta_y} & D \end{bmatrix} \quad (17)$$

where

$$\theta_\alpha = \frac{\theta_d - \theta_b}{2} \text{ and } \theta_\beta = -\frac{\theta_d + \theta_b}{2}$$

Transformation 3: The transformation that converts two diagonal elements (main diagonal) of a complex 2x2 matrix into real values is given by:

$$\begin{bmatrix} e^{i\theta_\alpha} & 0 \\ 0 & e^{i\theta_\alpha} \end{bmatrix}\begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix}\begin{bmatrix} e^{i\theta_\beta} & 0 \\ 0 & e^{-i\theta_\beta} \end{bmatrix} = \begin{bmatrix} A & Be^{i\theta_x} \\ Ce^{i\theta_y} & D \end{bmatrix} \quad (18)$$

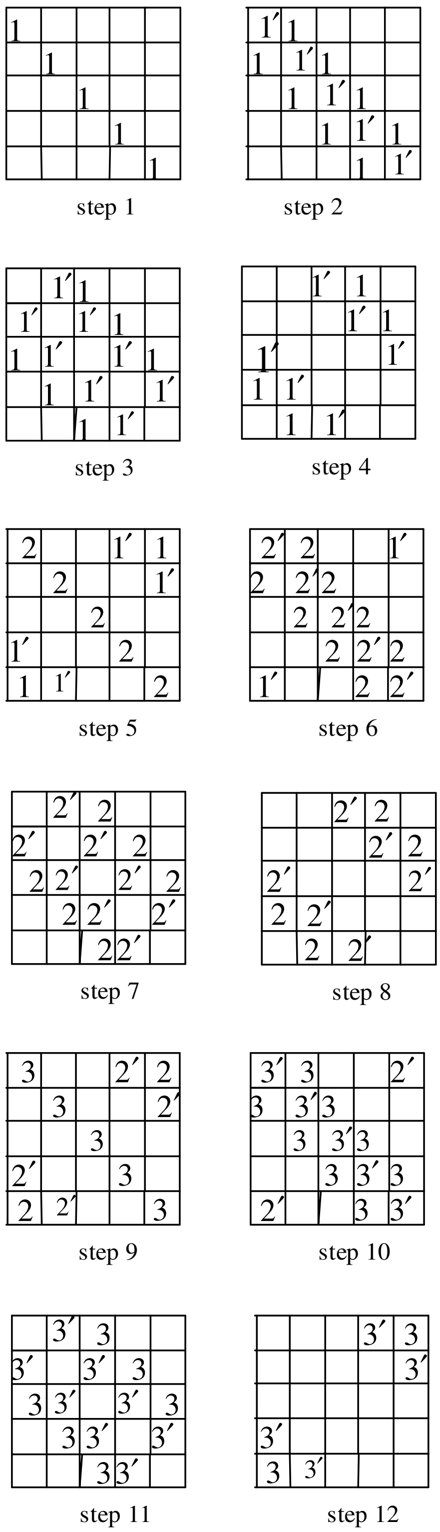
where

$$\theta_\alpha = -\frac{\theta_d + \theta_a}{2} \text{ and } \theta_\beta = \frac{\theta_d - \theta_a}{2}$$

Now, considering the transformations given above, it is possible to give general explanation of this procedure. For a complex matrix given as in (12), the first step is applying the transformation 1. After applying the transformation 2 and the transformation 3 the result will be a real matrix. Then the procedure is the same as in equations (2) and (3).

What about the systolic array of the SVD for a complex matrix? Each of the three transformations requires the generation and use of six angles, four unitary (a,b,c,d) and two rotational (α,β). These six angles must be propagated along both the rows and columns of processors on the main diagonal. The angles are generated by these processors and then are used for the diagonalization of a 2x2 complex matrices.

To improve the systolic array in the aspect of computation steps which are used, the performance will be as follows: during the computation by the diagonal processors using transformation 2, at the same time the nearest off diagonal processors may be used for computing using transformation 3. Similarly, the computations according transformation 3, may be performed before finishing the computations according transformation 1 and transformation 2. Thus, while the main diagonal are still computing the values according transformation 2, the nearest off diagonal processors are applying the transformation 1 etc. This pipelining of transformations by formulas (16), (17) and (18) is given in the following figure:



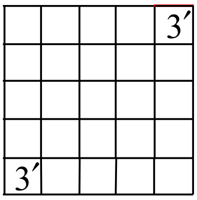


Fig. 3 Steps of computations on systolic array for the transforming the complex into the real matrix

After this procedure, the obtained matrix will be a real matrix, so the same systolic array as in figures 1 and 2 may be used for calculations.

References

[1] Carl G. J. Jacobi. *Uber eine neue Auflosungsart der bei der Methode der kleinsten Quadrate vorkommenden linearen Gleichungen.* Astronomishe Nachricten, 22, 1845. English translation by G.W. Stewart, Technical Report 2877, Department of Computer Science, University of Maryland, April 1992.

[2] G. H. Golub and C. F. Van Loan. *Matrix Computations.* John Hopkins University Press, Baltimore and London, 2nd edition, 1993.

[3] G. E. Forsythe and P. Henrici. *The cyclic Jacobi Method for Computing the Principal Values of a Complex Matrix.* Transactions of the American Mathematical Society, 94(1): 1-23, 1966.

[4] M. R.Hestenes. *Inversion of Matrices by Biorthogonalization and Related Results.* Journal of the Society for Industrial and Applied Mathematics, 6(1): 51-90, March 1958.

[5] R. P. Brent, F. T. Luk, and C. F. Van Loan. *Computation of the Singular Value Decomposition using Mesh-Connected Processors.* Journal of VLSI and Computer Systems, 1(3):242-270, 1985.

[6] E. G. Kogbetliantz, *Solution of Linear Equations by Diagonalization of Coefficients Matrix.* Quarterly of Applied Mathematics, 14(2): 123-132, 1955.

[7] G. H. Golub and C. F. Van Loan. *Matrix Computations,* Second edition . Johns Hopkins University Press,Baltimore, MD, 1989.

[8] C. M. Rader. Wafer Scale Systolic Array for Adaptive Antenna Proccessing. IEEE Int. Conf. on Acoustics, Speech and Signal Proccessing, 2069-2071, 1988.

[9] A. J. Van Der Veen and E. F. Deprettere. *A Parallel VLSI Direction Finding Algorithm.* Proc. SPIE Advanced Algorithms and Architectures for Signal Processing, 975(III): 289-299, 1988.

[10] J. R. Cavallaro and A. C. Elster. A Cordic *Proccessor Array for the SVD of a Complex Matrix*. In R. J. Vaccaro, editor, SVD and Signal Processing II, 227-239. Elsevier, New York, 1991.

[11] J. R. Cavallaro and F. T. Luk *CORDIC Arithmetic for an SVD Processor*. Journal of Parallel and Distributed Computing, 5(3):271-290, 1988.

[12] C. C. Sun and J. Gotze, *A VLSI design for parallel iterative algorithms.* Proceedings of the 9th int. conf. on Communications and information technologies, pp. 688-692, IEEE press Piscataway, NY, USA, 2009.

[13] Ahmedsaid, A., Amira, A., and Bouridane, *A.: Improved SVD systolic array and implementation on FPGA,* in: IEEE International Conference on Field Programmable Technologie, pp. 3-42, 2003.

[14] Gotze, J., Paul, S., and Sauer, M.: *An Efficient Jacobi-Like Algorithm for Parallel Eigenvalue Computation,* IEEE Transactions on Computers, 42, 1058-1065, 1993.

[15] Franklin T. Luk, A Parallel *Method for Computing the Generalized Singular Value Decomposition,* Journal of Parallel and Distributed Computing, volume 2, issue 3, pp. 250-260, august 1985,

[16] N. Suryanarayanan, *Efficient Architectures for Eigen Value Decomposition,* 2009, unpublished. (http://homepages.cae.wisc.edu/~ece734/project/s09/nikhil_rpt.pdf)