MODIFIED GIVENS ROTATIONS AND THEIR APPLICATION TO MATRIX INVERSION

Lei Ma, Kevin Dickson, John McAllister, John McCanny

Institute of Electronics, Communications and Information Technology, Queen's University, Belfast

ABSTRACT

Complex wireless communication systems such as MIMO require high-performance real-time implementations of operations such as matrix inversion. This paper presents two novel algorithms for this application. A novel modified conventional Givens rotations (MCGR) method has been derived which offers high-performance implementation since it avoids high-latency angle-based architectures, such as CORDIC. Furthermore, a novel modified squared Givens rotations (MSGR) method has been proposed which extends the original SGR method for complex valued data, and also corrects erroneous results in the original SGR method when zeros occur on the diagonal of the matrix either initially or during processing. In addition, both of the proposed methods avoid complex dividers in the matrix inversion, thus minimising the complexity of potential real-time implementations.

Index Terms— Matrix inversion, Matrix factorisation, Givens rotations

1. INTRODUCTION

The emerging field of multiple-input multiple-output (MIMO) systems for wireless communications (e.g. WiFi, WiMAX and *ad hoc* networks) promises to enhance the quantity of information that can be reliably transferred through a bandwidth constrained wireless channel. Widespread adoption of these techniques in practical applications depends on the effectiveness and efficiency of their real-time implementations, which has prompted research into algorithms for real-time implementation of critical fundamental operations, such as matrix inversion.

Methods for computing matrix inversion can be divided into two categories: *iterative* and *direct* [1]. Iterative methods require an initial estimate of the solution and subsequent updates based on calculation of the previous estimate error. Normally, these iterative methods involve high-complexity sequential matrix computations and are not particularly suitable for high-performance implementation. Direct methods include Gaussian elimination, Cholesky decomposition, LU decomposition and QR decomposition (QRD). These typically compute the solution in a known, finite number of operations and are, therefore, more suitable for time-constrained applications.

QRD has attracted particular attention for matrix inversion for MIMO applications for a number of reasons. Firstly, the numerical stability of QRD is well known [2]. In addition, the development of real-time architectures for QRD for recursive least squares (RLS) in adaptive beamforming and RADAR [3] is a well researched area.

Real-time implementations of emerging MIMO systems (e.g. T-BLAST [4]) demand high-performance and low-complexity matrix inversion operations on both real and complex valued data. Several methods have been proposed for the computation of QRD. Givens rotations [5] have the advantage over Householder transformations [6] that they can be more easily parallelised and are therefore more appropriate for hardware implementation. In turn, several methods exist for the computation of Givens rotations, i.e. conventional Givens rotations (CGR) [5], squared Givens rotations (SGR) [7] and CORDIC [8]. The iterative nature of CORDIC can preclude it from high-throughput systems [9] whilst CGR requires resource-expensive square-root operations, prompting the emergence of SGR-based algorithms which remove the need for these operations. However, issues exist with SGR [7] in that it is derived for real valued data only, and it produces erroneous results when zeros occur on the diagonal elements of the matrix either initially or during processing. These requirements prompted the modified Givens rotations methods proposed in this paper. In Section 3, a modified conventional Givens rotations (MCGR) method that avoids the use of high-latency iterative CORDIC and also minimises computational complexity by avoiding complex divides during back substitution stage is outlined. In Section 4, this is extended to modified squared Givens rotations (MSGR) for both real and complex valued matrices, while in Section 5, the technique for coping with zeros on the diagonal is outlined. The proposed MSGR method is a comprehensive, reliable and low-complexity solution suitable for matrix inversion operations in high-performance applications such as MIMO systems.

2. QRD-BASED MATRIX INVERSION

Computation of the inverse of a matrix **A** can be performed by firstly decomposing the matrix into a form which can be more easily inverted. For example, QRD of a matrix **A** results in

$$\mathbf{A} = \mathbf{QR} \tag{1}$$

where, **Q** is an orthogonal matrix (for a complex valued matrix **A**, **Q** is a unitary matrix) and **R** is an upper triangular matrix. The inversion of the matrix **A** is then given by

$$\mathbf{A}^{-1} = (\mathbf{QR})^{-1} = \mathbf{R}^{-1}\mathbf{Q}^{-1} = \mathbf{R}^{-1}\mathbf{Q}^{H} \quad (2).$$

After QRD, inversion is much simpler because the inversion of the upper triangular matrix **R** can be derived using back substitution, as in (3) where $\mathbf{W}=\mathbf{R}^{-1}$, and the inversion of **Q** is simply its Hermitian transpose.

$$W_{ij} = \begin{cases} -(\sum_{k=1}^{j-1} w_{ik} r_{kj}) / r_{jj} & i < j \\ 1/r_{ij} & i = j \\ 0 & i > j \end{cases} \quad (3)$$

3. MODIFIED CGR

In this section, we derive a MCGR method for matrices with complex values which not only avoids angle calculations (usually computed using high-latency CORDIC), but also avoids complex divide operations in the subsequent back substitution stage. In order to study the CGR method, we first take the example of rotating a 2×2 matrix of real values, $[a_1\ a_2;\ b_1\ b_2]$. A CGR is performed in order to zero the lower diagonal element, as in (4) with θ given by (5).

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \\ 0 & y_2 \end{bmatrix} \quad (4)$$

$$\theta = \tan^{-1}(b_1 / a_1) \quad (5)$$

This method of directly computing angles and subsequent rotations employs a technique such as CORDIC, which supports such operations (e.g. inverse tan). However, in order to avoid high-latency CORDIC and to instead use conventional arithmetic operations, it is necessary to rework the rotation matrix in (4) as

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \frac{1}{\sqrt{a_1^2 + b_1^2}} \begin{bmatrix} a_1 & b_1 \\ -b_1 & a_1 \end{bmatrix} \quad (6).$$

If we then consider a 2×2 matrix of complex values, e.g.

$$\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} = \begin{bmatrix} A_1 e^{i\theta_{a1}} & A_2 e^{i\theta_{a2}} \\ B_1 e^{i\theta_{b1}} & B_2 e^{i\theta_{b2}} \end{bmatrix} \quad (7)$$

then using the method proposed by Coleman and Van Loan [10], a complex CGR can be described using two rotation angles (θ_1 and θ_2) as in (8)-(10).

$$\begin{bmatrix} \cos\theta_1 & \sin\theta_1 e^{i\theta_2} \\ -\sin\theta_1 e^{i\theta_2} & \cos\theta_1 \end{bmatrix} \begin{bmatrix} A_1 e^{i\theta_{a1}} & A_2 e^{i\theta_{a2}} \\ B_1 e^{i\theta_{b1}} & B_2 e^{i\theta_{b2}} \end{bmatrix} = \begin{bmatrix} X_1 e^{i\theta_{x1}} & X_2 e^{i\theta_{x2}} \\ 0 & Y_2 e^{i\theta_{y2}} \end{bmatrix} \quad (8)$$

$$\theta_1 = \tan^{-1}(B_1 / A_1) \quad (9)$$

$$\theta_2 = \theta_{a1} - \theta_{b1} \quad (10).$$

The rotation matrix in (8) may be expanded as

$$\begin{bmatrix} \cos\theta_1 & \sin\theta_1 e^{i\theta_2} \\ -\sin\theta_1 e^{i\theta_2} & \cos\theta_1 \end{bmatrix} = \begin{bmatrix} e^{i\theta_{a1}} & 0 \\ 0 & e^{i\theta_{b1}} \end{bmatrix} \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} e^{-i\theta_{a1}} & 0 \\ 0 & e^{-i\theta_{b1}} \end{bmatrix} \quad (11).$$

Use of this method produces complex values on the diagonal of the resultant upper triangular matrix, which significantly increases the computational complexity of the back substitution stage, as in (3). In particular, complex divides are required on the diagonal elements, which considerably increase the complexity of a potential hardware implementation.

In order to address this issue, a three angle complex rotation (TACR) method is proposed in [11] which generates only real values on the diagonal. The TACR rotation matrix, written as in (12), can be applied to a $N\times K$ matrix to generate an upper triangular matrix with real values on the diagonal, apart from the diagonal element on the N^{th} row, which will be a complex value. To make this real, the N^{th} row must be rotated by $-\phi$ (equivalent to multiplying this row by $e^{-i\phi}$) where ϕ is the angle of the diagonal element.

$$\begin{bmatrix} \cos\theta_1 e^{-i\theta_{a1}} & \sin\theta_1 e^{-i\theta_{b1}} \\ -\sin\theta_1 e^{-i\theta_{a1}} & \cos\theta_1 e^{-i\theta_{b1}} \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} e^{-i\theta_{a1}} & 0 \\ 0 & e^{-i\theta_{b1}} \end{bmatrix} \quad (12)$$

As noted earlier, for the case of real values, these methods (Coleman and Van Loan and TACR) involve angle calculations which imply high-latency CORDIC implementations. However, it is possible to modify the TACR method for the general case which can then be implemented using either CORDIC or conventional arithmetic. Here, we propose a modified conventional Givens rotations (MCGR) method by multiplying the TACR rotation matrix by a unitary matrix, i.e.

$$\begin{bmatrix} 1 & 0 \\ 0 & e^{i(\theta_{a1}+\theta_{b1})} \end{bmatrix} \begin{bmatrix} \cos\theta_1 e^{-i\theta_{a1}} & \sin\theta_1 e^{-i\theta_{b1}} \\ -\sin\theta_1 e^{-i\theta_{a1}} & \cos\theta_1 e^{-i\theta_{b1}} \end{bmatrix} = \begin{bmatrix} \cos\theta_1 e^{-i\theta_{a1}} & \sin\theta_1 e^{-i\theta_{b1}} \\ -\sin\theta_1 e^{i\theta_{b1}} & \cos\theta_1 e^{i\theta_{a1}} \end{bmatrix} = \frac{1}{\sqrt{|a_1|^2 + |b_1|^2}} \begin{bmatrix} a_1^* & b_1^* \\ -b_1 & a_1 \end{bmatrix} \quad (13)$$

where, z^* is the complex conjugate of z. The MCGR rotation matrix (13) can now be implemented using either high-latency angle based methods (e.g. CORDIC) or conventional arithmetic operators. In addition, as with TACR, the diagonal element on the N^{th} row can be made real by rotating the row according to the angle of this diagonal element.

4. MODIFIED SGR

The use of resource-expensive square-root operations in CGR prompted the development of the SGR method [7] for calculating the upper triangular matrix without the need for these operations and with reduced multiplications compared to CGR. However, [7] addresses real valued data only and produces erroneous results when zeros occur on the diagonal. In this section, a MSGR method for complex valued matrices is presented before the proposition of a

novel technique to handle the case of zeros on the diagonal in Section 5.

Using SGR, the matrix **A** is decomposed as follows

$$\mathbf{A} = \mathbf{Q}_A \mathbf{D}_U^{-1} \mathbf{U} \tag{14}$$

where, $\mathbf{Q}_A$ in general is not an orthogonal matrix, and **U** is an upper triangular matrix,

$$\mathbf{Q}_A = \mathbf{Q}\mathbf{D}_R \tag{15}$$

$$\mathbf{D}_R = diag(\mathbf{R}) \tag{16}$$

$$\mathbf{U} = \mathbf{D}_R \mathbf{R} \tag{17}$$

$$\mathbf{D}_U = diag(\mathbf{U}) = diag(\mathbf{D}_R \mathbf{R}) = \mathbf{D}_R^2 \tag{18}.$$

The inversion of the matrix **A** is then given by

$$\mathbf{A}^{-1} = (\mathbf{Q}_A \mathbf{D}_U^{-1} \mathbf{U})^{-1} = \mathbf{U}^{-1} (\mathbf{Q}_A \mathbf{D}_U^{-1})^{-1} \tag{19}.$$

As **U** is an upper triangular matrix, its inversion can be found by back substitution, as in (3). Also, since

$$\begin{aligned} (\mathbf{Q}_A \mathbf{D}_U^{-1})^{-1} &= (\mathbf{Q}\mathbf{D}_R^{-1})^{-1} = \mathbf{D}_R \mathbf{Q}^{-1} \\ &= \mathbf{D}_R \mathbf{Q}^H = (\mathbf{Q}\mathbf{D}_R)^H = \mathbf{Q}_A^H \end{aligned} \tag{20}$$

inversion of this component is simply a Hermitian transpose.

To illustrate the MSGR method, consider a 3×4 matrix of complex values as in (21).

$$\begin{bmatrix} \mathbf{r} \\ \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \end{bmatrix} \tag{21}$$

MSGR generates an upper triangular matrix, eliminating a_1, b_1 and b_2 in a three-stage approach.

Stage 1: Rotate rows **r** and **a** to eliminate element a_1 (i.e. make $\overline{a}_1$ =0). From Section 3, MCGR for rows **r** and **a** with complex values (13) can be written as

$$q = (r_1^* r_1 + a_1^* a_1)^{1/2} \tag{22}$$

$$\overline{\mathbf{r}} = q^{-1}(r_1^* \mathbf{r} + a_1^* \mathbf{a}) \tag{23}$$

$$\overline{\mathbf{a}} = q^{-1}(-a_1 \mathbf{r} + r_1 \mathbf{a}) \tag{24}.$$

From (23), we can see that

$$\overline{r}_1^* = \overline{r}_1 = q^{-1}(r_1^* r_1 + a_1^* a_1) = q \tag{25}$$

and therefore,

$$r_1^* \overline{\mathbf{r}} = r_1^* \mathbf{r} + a_1^* \mathbf{a} \tag{26}.$$

If we first assume that

$$\mathbf{u} = r_1^* \mathbf{r} \tag{27}$$

$$\overline{\mathbf{u}} = \overline{r}_1^* \overline{\mathbf{r}} \tag{28}$$

then, combining (26)-(28), the updating of row **r** can be written as

$$\overline{\mathbf{u}} = \mathbf{u} + a_k^* \mathbf{a} \tag{29}.$$

In a similar way, if we assume

$$\mathbf{a} = w_a^{1/2} \mathbf{v} \tag{30}$$

where, w_a>0 is a scale factor, then (24) can be rewritten as

$$\overline{\mathbf{a}} = w_a^{1/2} \frac{r_1}{\overline{r}_1} \left(\mathbf{v} - \frac{v_1}{r_1} \mathbf{r} \right) = w_a^{1/2} \frac{r_1}{\overline{r}_1} \left(\mathbf{v} - \frac{v_1}{u_1} \mathbf{u} \right) \tag{31}.$$

Effectively, the rows **r** and **a** have been translated to **U** and **V**-space respectively. If we further assume that

$$\overline{w}_a = w_a^{1/2} \frac{r_1}{\overline{r}_1} \left(w_a^{1/2} \frac{r_1}{\overline{r}_1} \right)^* = w_a \frac{r_1^* r_1}{\overline{r}_1^* \overline{r}_1} = w_a u_1 / \overline{u}_1 \tag{32}$$

then the MSGR updating can be rearranged as

$$\overline{\mathbf{u}} = \mathbf{u} + w_a v_1^* \mathbf{v} \tag{33}$$

$$\overline{\mathbf{v}} = \mathbf{v} - (v_1 / u_1)\mathbf{u} \tag{34}$$

$$\overline{w}_a = w_a u_1 / \overline{u}_1 \tag{35}.$$

Stage 2: Rotate $\overline{\mathbf{r}}$ and **b** to eliminate b_1. Since, $\overline{\mathbf{r}}$ is in **U**-space (i.e. $\overline{\mathbf{u}} = \overline{r}_1^* \overline{\mathbf{r}}$), row **b** must now be translated to **V**-space for rotation. This is achieved by setting

$$\mathbf{b} = w_b^{1/2} \mathbf{v}_\mathbf{b} \tag{36}.$$

As before, we find that

$$\overline{w}_b = w_b^{1/2} \frac{\overline{r}_1}{\overline{\overline{r}}_1} \left(w_b^{1/2} \frac{\overline{r}_1}{\overline{\overline{r}}_1} \right)^* = w_b \frac{\overline{r}_1^2}{\overline{\overline{r}}_1^2} = w_b \overline{u}_1 / \overline{\overline{u}}_1 \tag{37}.$$

(Note that after rows **a** and **b** have been processed, $\overline{r}_1$ will be a real value). Therefore, the updating of row **b** can be written as

$$\begin{aligned} \overline{\mathbf{b}} &= w_b^{1/2} \frac{\overline{r}_1}{\overline{\overline{r}}_1} \left(\mathbf{v}_\mathbf{b} - \frac{v_{b1}}{\overline{r}_1} \overline{\mathbf{r}} \right) = w_b^{1/2} \frac{\overline{r}_1}{\overline{\overline{r}}_1} \left(\mathbf{v}_\mathbf{b} - \frac{v_{b1}}{\overline{u}_1} \overline{\mathbf{u}} \right) \\ &= \overline{w}_b^{1/2} \overline{\mathbf{v}}_\mathbf{b} \end{aligned} \tag{38}.$$

Stage 3: Rotate $\overline{\mathbf{a}}$ and $\overline{\mathbf{b}}$ to eliminate $\overline{b}_2$. In order to carry out the rotations, $\overline{\mathbf{a}}$ must be translated from **V**-space to **U**-space and $\overline{\mathbf{b}}$ must be translated to **V**-space, as described in the following

$$\begin{aligned} \mathbf{u}_\mathbf{a} &= \overline{a}_2^* \overline{\mathbf{a}} \\ &= \left(w_a^{1/2} \frac{r_1}{\overline{r}_1} \left(v_2 - \frac{v_1}{u_1} u_2 \right) \right)^* \left(w_a^{1/2} \frac{r_1}{\overline{r}_1} \left(\mathbf{v} - \frac{v_1}{u_1} \mathbf{u} \right) \right) = \overline{w}_a \overline{v}_2^* \overline{\mathbf{v}} \end{aligned} \tag{39}$$

$$\mathbf{v}_\mathbf{bnew} = w_{new}^{-1/2} \overline{\mathbf{b}} = w_{new}^{-1/2} \overline{w}_b^{1/2} \overline{\mathbf{v}}_\mathbf{b} \tag{40}.$$

Therefore, if we let $w_{new} = \overline{w}_b$, then $\mathbf{v}_\mathbf{bnew} = \overline{\mathbf{v}}_\mathbf{b}$. This implies that the updated scale factor $\overline{w}_b$ must be used to translate $\overline{\mathbf{b}}$ in order to use $\overline{\mathbf{v}}_\mathbf{b}$ as the new **V**-space vector for further processing. The MSGR sequence of operations is illustrated in Fig. 1.

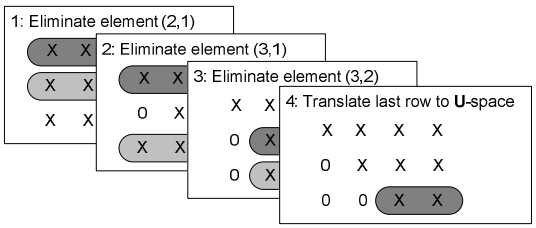

Fig. 1. MSGR sequence of operations

The final phase of translating the last row to **U**-space is necessary in order to make the diagonal element in this row

a real value. The MSGR method for the general case is described using (41)-(43)

$$\overline{\mathbf{u}} = \mathbf{u} + wv_k^*\mathbf{v} \tag{41}$$

$$\overline{\mathbf{v}} = \mathbf{v} - (v_k / u_k)\mathbf{u} \tag{42}$$

$$\overline{w} = wu_k / \overline{u}_k \tag{43}$$

where, k is the k^{th} column currently being processed.

5. HANDLING ZEROS ON THE DIAGONAL

The analysis in Section 4 assumes that $u_k \neq 0$, however, zeros can occur on the diagonal either in the input matrix or during phases of rotations (Fig. 2). Rotating two rows which have pairs of adjacent equal-valued elements on the diagonal will result in zeros on the first element (as desired) but also on the diagonal position (undesired) of the lower rotated row. During the next phase of rotations, (i.e. rotations to zero lower diagonal elements in the next column) the first element of the **U**-space is zero and must be dealt with before further rotations can be carried out.

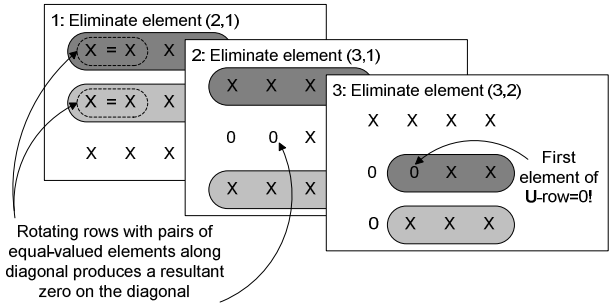

Fig. 2. Occurrence of zeros on diagonal

In [7], Dohler proposed a solution for dealing with $u_k=0$. However, the condition of both $u_k=v_k=0$ has not previously been considered. Dohler proposed that,

$$\text{for } u_k = 0, \; v_k \neq 0 \begin{cases} \overline{\mathbf{u}} = wv_k^*\mathbf{v} \\ \overline{\mathbf{v}} = \textit{arbitrary} \\ \overline{w} = 0 \end{cases}$$

This solution prohibits further processing based on these updated values. For example, translation of a row from **V**-space to **U**-space (such as that described in (39)) would not be possible using Dohler's suggested updated scale factor $\overline{w} = 0$. To overcome this problem, a novel solution for the condition $u_k=0$, $v_k\neq 0$ which permits subsequent processing, as well as a solution for $u_k=v_k=0$, which has not been previously considered, is proposed, i.e.

$$\text{for } u_k = 0, \; v_k \neq 0 \begin{cases} \overline{\mathbf{u}} = wv_k^*\mathbf{v} \\ \overline{\mathbf{v}} = -\mathbf{u} \\ \overline{w} = w \end{cases} \qquad \text{for } u_k = 0, \; v_k = 0 \begin{cases} \overline{\mathbf{u}} = w\mathbf{v} \\ \overline{\mathbf{v}} = -\mathbf{u} \\ \overline{w} = w \end{cases}$$

Thus, the proposed MSGR method provides a comprehensive and accurate solution for QRD-based complex matrix inversion, such as that required in high-performance MIMO systems.

6. SUMMARY

In this paper, modified Givens rotations methods have been proposed for QRD-based complex matrix inversion suitable for high-performance applications such as MIMO systems.

Firstly, a novel modified conventional Givens rotations (MCGR) method has been derived which offers high-performance implementation since it avoids high-latency angle-based architectures, such as CORDIC. Secondly, a novel modified squared Givens rotations (MSGR) method has been proposed which extends the original SGR method for complex valued data, and also corrects erroneous results in the original SGR method when zeros occur on the diagonal of the matrix either initially or during processing. In addition, both of the proposed methods avoid complex dividers in the matrix inversion, thus minimising the complexity of potential real-time implementations.

7. REFERENCES

[1] M. Ylinen, A. Burian and J. Takala, "Updating matrix inverse in fixed-point representation: Direct versus iterative methods," in *International Symposium on System-on-Chip,* pp. 45-8, 2003.

[2] S. Haykin, *Adaptive Filter Theory*, 2nd ed. Prentice Hall, 1991.

[3] G. Lightbody, R. Walke, R. Woods and J. McCanny, "Linear QR architecture for a single chip adaptive beamformer," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology,* vol. 24, pp. 67-81, 2000.

[4] M. Sellathurai and S. Haykin, "TURBO-BLAST for wireless communications: Theory and experiments," *IEEE Trans. Signal Processing,* vol. 50, pp. 2538-2546, 2002.

[5] W. Givens, "Computation of plane unitary rotations transforming a general matrix to triangular form," *J. Soc. Indust. Appl. Math,* vol. 6, pp. 26-50, 1958.

[6] G. Golub, "Numerical methods for solving least-squares problems," *Num. Math*., no.7, pp. 206-216, 1965.

[7] R. Dohler, "Squared Givens rotation," *IMA Journal of Numerical Analysis,* vol. 11, pp. 1-5, 01. 1991.

[8] J. E. Volder, "The cordic trigonometric computing technique," *Institute of Radio Engineers Trans. Electronic Computing,* vol. EC-8, pp. 330-334, 1959.

[9] R. L. Walke, R. W. M. Smith and G. Lightbody, "Architectures for adaptive weight calculation on ASIC and FPGA," *Conference Record of the Asilomar Conference on Signals, Systems and Computers,* vol. 2, pp. 1375-1380, 1999.

[10] T. F. Coleman and C. F. Van Loan, *Handbook for Matrix Computations*, SIAM, 1988.

[11] A. Maltsev, V. Pestretsov, R. Maslennikov and A. Khoryaev, "Triangular systolic array with reduced latency for QR-decomposition of complex matrices," in *2006 IEEE International Symposium on Circuits and Systems,* pp.385-388, 2006.