Evaluation Of A High Performance Qr Decomposition-Based Matrix Inversion Technique For Embedded Mimo Receivers

P.G.C. VIKRAM
M.Tech Student Scholar, DECS,
Dept of Electronics and Communication Engineering,
Nalanda Institute of Engineering and technology,
Sattenapalli (M); Guntur (Dt); A.P, India

N.GOPI CHAND
M.Tech, Asst Professor
Dept of Electronics and Communication Engineering,
Nalanda Institute of Engineering and technology,
Sattenapalli (M); Guntur (Dt); A.P, India

Abstract

Matrix inversion is an essential computation for various algorithms which are employed in multi antenna wireless communication systems. FPGAs are ideal platforms for wireless communication; however, the need for vast amounts of customization throughout the design process of a matrix inversion core can overwhelm the designer. Decomposition methods provide the analytic simplicity and computational convenience necessary for computationally intensive matrix inversion. Real-time matrix inversion is a key enabling technology in multiple-input multiple-output (MIMO) communications systems, such as 802.11n. To date, however, no matrix inversion implementation has been devised which supports real-time operation for these standards. In this paper, we overcome this barrier by presenting a novel matrix inversion algorithm which is ideally suited to high performance floating-point implementation. We show how the resulting architecture offers fundamentally higher performance than currently published matrix inversion approaches and we use it to create the first reported architecture capable of supporting real-time 802.11n operation. Specifically, we present a matrix inversion approach based on modified squared Givens rotations (MSGR). This is a new QR decomposition algorithm which overcomes critical limitations in other QR algorithms that prohibits their application to MIMO systems. In addition, we present a novel modification that further reduces the complexity of MSGR by almost 20%. This enables real-time implementation with negligible reduction in the accuracy of the inversion operation, or the BER of a MIMO receiver based on this.

***Keywords*: BLAST, matrix inversion, multiple-input multiple-output (MIMO) , QR decomposition.**

I. INTRODUCTION

Matrix inversion is a common function found in many algorithms used in wireless communication systems. For example MIMO-OFDM systems use matrix inversion in equalization algorithms to remove the effect of the channel on the signal [1], minimum mean square error algorithms for pre-coding in spatial multiplexing [2] and detection-estimation algorithms in space-time coding [3]. These systems often use a small number of antennas (2 to 8) which results in small matrices to be inverted and/or decomposed. For example the 802.11n standard specifies a maximum of 4 antennas on the transmit/receive sides and the 802.16 standard specifies a maximum of 16 antennas at a base station and 2 antennas at a remote station. Matrix inversion is a computationally intensive calculation. Decomposition methods provide a means to simplify this computation. There are different decomposition methods, such as QR, LU and Cholesky, that solve matrix inversion. The selection of the decomposition method depends on the characteristics of the given matrix. For non-square matrices or when simple inversion to recover the data performs poorly, the QR decomposition is used to generate an equivalent upper triangular system, allowing for detection using the sphere decomposition or M-algorithm. For simpler detection via inversion of square channel matrices, the LU and Cholesky decompositions are compatible with positive definite and nonsingular diagonally dominant square matrices, respectively[4].

FPGAs are an ideal platform for wireless communication due to their high processing power, flexibility and non recurring engineering (NRE) cost. However, FPGAs require vast amounts of customization throughout the design process and few tools exist which can aid the designer with the many system, architectural and logic design choices. Designing a high level tool for fast prototyping matrix inversion architectures is crucial. Multi Input – Multi Output (MIMO) technology, such as BLAST [1]–[3] for WiFi or WiMAX offers the potential to exploit spatial diversity in a communications channel to increase its bandwidth without sacrificing larger portions of the radio spectrum. The general form of a MIMO system composed of transmit and receive antennas is outlined in Figure. 1. Implementation of these systems involves satisfying the real-time performance requirements of the application (in terms of metrics such as throughput, latency, etc.), in a manner which efficiently exploits the embedded device(s) to implement such systems.

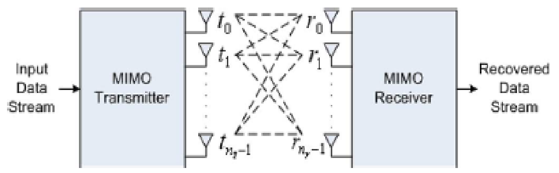


Figure 1: MIMO system Overview

A key feature of MIMO receivers is their reliance on matrix computations such as addition, multiplication and inversion— for instance, to enable operations such as minimum mean square error (MMSE) equalization during channel detection [3]. In particular, complex matrix inversion has proven so difficult to implement that, to the best of the authors' knowledge, there are no reported implementations capable of sustaining the 14.4 MInversions/s (MInv/s) and 4 latency required for modern IEEE 802.11n MIMO systems [4]–[7]. As a result, to implement algorithms such as MMSE designers currently have to develop custom algorithms which avoid explicit matrix inversion [8]. This severely complicates the implementation process. This paper presents a new matrix inversion approach which overcomes this real-time performance barrier. We develop a general purpose complex-valued matrix inversion algorithm and study its application to and integration in MIMO receiver algorithms and embedded architectures. Specifically, we make three main contributions.

We derive a new QR decomposition (QRD)-based algorithm known as modified squared Givens' rotations (MSGR), which overcomes key limitations with other QRD-based approaches which hinder their adoption in MIMO receiver architectures. We show how the complexity of MSGR-based matrix inversion may be further reduced by almost 20% by removing a scale factor term, with little effect on its numerical stability, or the perceived BER of a MIMO receiver in which it is integrated.

II. BACKGROUND

The need to detect 52 data subcarriers within both the OFDM symbol and short inter frame spacing periods of 802.11n places extreme real-time constraints on embedded signal processing architecture for these systems. For instance, [8] shows that there is no current matrix inversion implementation capable of sustaining the 14.4 MInvs/s with 4 latency for MMSE in 802. 11n. This necessitates complicated redesign of algorithms as fundamental as MMSE specifically to avoid explicit matrix inversion and clearly complicates the design process in a way which is not required for less complex matrix operations such as addition or multiplication. It also does not solve the long term problem, since algorithms such as the MMSE in [8] exhibit the same O( ) complexity matrix inversion algorithms [4]–[7]. As a result, as MIMO systems grow larger to incorporate more antennas, both solutions will tend to the same complexity. However, given that the matrix inversion approaches in [4]–[7] are all either throughput or latency deficient by a factor of at least 2, there appear to be substantial issues to be overcome before real-time matrix inversion in systems such as 802.11n is feasible.

Matrix inversion techniques are, generally, either *iterative* or *direct* [9]. Iterative methods, such as the Jacobi or Gauss-Seidel methods, start with an estimate of the solution and iteratively update the estimate based on calculation of the error in the previous estimate, until a sufficiently accurate solution is derived. The sequential nature of this process can limit the amount of available parallelism and make high throughput implementation problematic [9]. On the other hand, direct methods such as Gaussian elimination (GE), Cholesky decomposition (CD), and QRD typically compute the solution in a known, finite number of operations and exhibit plentiful data and task-level parallelism.The complexity of a number of direct matrix inversion algorithms in MIMO communications systems composed of antennas, as well as an absolute complexity measure for .

All these approaches exhibit similar complexity scaling: O( ) additions and multiplications and O( ) divisions. Whilst relatively low complexity when compared to the others, CD suffers from the drawback of requiring symmetric matrices, a condition not guaranteed to occur in MIMO systems, limiting its applicability. Despite their relatively high complexity as compared with CD, QRD approaches are an attractive alternative not only because of their ability to overcome the symmetric restriction, but also because of their innate numerical stability [10]. Furthermore, the plentiful data and task level parallelism available in these algorithms has previously been comprehensively exploited in a range of algorithms and architectures for recursive least squares (RLS) in adaptive beam forming and RADAR.

As regards QRD algorithms, Givens' rotations [14] QRD algorithms are more easily parallelized than Householder Transformations [15], but the methods for implementing Givens' rotations, i.e., conventional Givens' rotations (CGR) [14], squared Givens' rotations (SGR) [16], and CORDIC [17] all place different constraints on implementations. The authors in [18] show that fixed-point CORDIC-based QR algorithms are more accurate for linear MMSE detection of practical MIMO-OFDM channels than SGR employing conventional arithmetic. However this comes at an excessively high area cost due to the use of CORDIC operators; indeed [8] and [19] report 3.5:1 and 3:1 area efficiency advantages when employing conventional mathematical operators as opposed to CORDIC; [19] also describes a 25:1 sample rate advantage associated with employing conventional arithmetic and demonstrates that floating-point arithmetic can be employed to overcome the precision issues outlined in [18] at no area cost. These factors seem to favour SGR-based implementation over CORDIC. Whilst CGR does not fundamentally require CORDIC for implementation, it does require widespread use of costly square-root operations, and is generally more computationally demanding than SGR As such, the ability of SGR to avoid the use of CORDIC and square-root operations, reduce overall complexity and exploit floating-point arithmetic at no area cost promises an appealing blend of numerical stability and computational complexity. There are, however, a number of critical deficiencies which restrict its adoption in MIMO systems[10].

1) As described in Section III-B, SGR produces erroneous results when zeros occur on the diagonal elements of either the input matrix or the partially decomposed matrices generated during the triangularization.
2) Complex-valued SGR is highly computationally demanding and there are no reported variants which exhibit
significantly lower complexity whilst maintaining accuracy.
3) There is no currently reported implementation of SGR based matrix inversion which can meet the high real-time performance demands of modern MIMO receivers. In this paper, we resolve these issues in a new SGR approach, known as MSGR. Section III derives MSGR specifically to overcome the issues in 1), before Section IV describes how the complexity of this algorithm may be reduced by a further 20% with little reduction in accuracy, addressing 2).

III. MODIFIED SGR FOR COMPLEX MATRIX INVERSION

A. Modified Squared Givens Rotations

Inversion of a matrix A can be performed by firstly decomposing this into an upper triangular form, which can be more easily inverted. QRD is one such approach to perform this triangularization by decomposing A into two resulting matrices Q and R, such that

A= QR

(1)

Q is a unitary matrix and R is an upper triangular matrix. From (1) it can be implied that the inverse of A is the inverse of the QR product and since Q is unitary, is simply its Hermitian transpose . Given this, is given by (2). After QRD, inversion is much simpler because the inversion of the upper triangular matrix R can be derived using back substitution, as in

= (Q =

(2)

$$W_{ij} = \begin{cases} -\sum_{k=1}^{j-1} w_{ik} r_{kj} / r_{jj} & i < j \\ 1/r_{jj} & i = j \\ 0 & \text{otherwise.} \end{cases}$$

(3)

Using SGR, the matrix A is decomposed as given in (4)–(8), where is in general not a unitary matrix, and U is an upper triangular matrix

$$\mathbf{A} = \mathbf{Q}_A \mathbf{D}_U^{-1} \mathbf{U} \quad (4)$$
$$\mathbf{Q}_A = \mathbf{Q}\mathbf{D}_R \quad (5)$$
$$\mathbf{D}_R = \mathrm{diag}(\mathbf{R}) \quad (6)$$
$$\mathbf{U} = \mathbf{D}_R \mathbf{R} \quad (7)$$
$$\mathbf{D}_U = \mathrm{diag}(\mathbf{D}_R \mathbf{R}) = \mathbf{D}_R^2. \quad (8)$$

The inverse of the matrix A is then given by (9). As U is an upper triangular matrix, its inverse can be found by back substitution, as in (3). In addition, since (10) holds, inversion of this component is simply a Hermitian transpose operation.

$$\mathbf{A}^{-1} = \left(\mathbf{Q}_A \mathbf{D}_U^{-1} \mathbf{U}\right)^{-1} = \mathbf{U}^{-1}\left(\mathbf{Q}_A \mathbf{D}_U^{-1}\right)^{-1} \quad (9)$$
$$\left(\mathbf{Q}_A \mathbf{D}_U^{-1}\right)^{-1} = \left(\mathbf{Q}\mathbf{D}_R^{-1}\right)^{-1} = \mathbf{D}_R \mathbf{Q}^H = (\mathbf{Q}\mathbf{D}_R)^H = \mathbf{Q}_A^H. \quad (10)$$

To illustrate how MSGR extends SGR for complex-valued data, we use an example. Consider a 3× 4 matrix of complex
values as in (11). MSGR generates an upper triangular matrix, eliminating , and in a three-stage approach

$$\begin{bmatrix} r \\ a \\ b \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \end{bmatrix}. \quad (11)$$

Stage 1: Rotate rows r and a to eliminate element a_1
(i.e., update a_1 to $_1$ such that $_1$ =0). To achieve this, the MSGR updating quations can be written as (12)–(14), where is the updated value of r

$$q = (r_1^* r_1 + a_1^* a_1)^{\frac{1}{2}} \quad (12)$$
$$\overline{\mathbf{r}} = q^{-1}(r_1^* \mathbf{r} + a_1^* \mathbf{a}) \quad (13)$$
$$\overline{\mathbf{a}} = q^{-1}(-a_1 \mathbf{r} + r_1 \mathbf{a}). \quad (14)$$

From (13), (15) follows and therefore

$$\overline{r}_1^* = \overline{r}_1 = q^{-1}(r_1^* r_1 + a_1^* a_1) = q \quad (15)$$
$$\overline{r}_1^* \overline{\mathbf{r}} = r_1^* \mathbf{r} + a_1^* \mathbf{a}. \quad (16)$$

Introducing , defined by (17), then combining (16)–(18) allows the update process for to be expressed as

$$\mathbf{u} = r_1^* \mathbf{r} \quad (17)$$

$$\bar{\mathbf{u}} = \bar{r}_1^* \bar{\mathbf{r}} \quad (18)$$

$$\bar{\mathbf{u}} = \mathbf{u} + a_1^* \mathbf{a} \quad (19)$$

Similarly, introducing as defined by (20), where is a scale factor, then (14) can be written as [16]

$$\mathbf{a} = w_a^{\frac{1}{2}} \mathbf{v} \quad (20)$$

$$\bar{\mathbf{a}} = w_a^{\frac{1}{2}} \frac{r_1}{\bar{r}_1} \left(\mathbf{v} - \frac{v_1}{r_1} \mathbf{r} \right) = w_a^{\frac{1}{2}} \frac{r_1}{\bar{r}_1} \left(\mathbf{v} - \frac{v_1}{u_1} \mathbf{u} \right). \quad (21)$$

Effectively, and have been translated to U and V -space, respectively. Further, if (22) holds, then the MSGR updating
equations from (12)–(14) can be rearranged as

$$\overline{w}_a = w_a^{\frac{1}{2}} \frac{r_1}{\bar{r}_1} \left(w_a^{\frac{1}{2}} \frac{r_1}{\bar{r}_1} \right)^* = w_a \frac{r_1^* r_1}{\bar{r}_1^* \bar{r}_1} = w_a \frac{u_1}{\bar{u}_1} \quad (22)$$

$$\bar{\mathbf{u}} = \mathbf{u} + w_a v_1^* \mathbf{v} \quad (23)$$

$$\bar{\mathbf{v}} = \mathbf{v} - \frac{v_1}{u_1} \mathbf{u} \quad (24)$$

$$\overline{w}_a = w_a \frac{u_1}{\bar{u}_1}. \quad (25)$$

Stage 2: Rotate and TO ELIMINATE . Since is in -space [given by (17)], row must now be translated to -space for rotation according to

$$\mathbf{b} = w_b^{\frac{1}{2}} \mathbf{v}_b. \quad (26)$$

Equation (27) defines , where is real as indicated by (15). Accordingly, updating of row b can be written as (28)
(where indicates the second update of r)

$$\overline{w}_b = w_b^{\frac{1}{2}} \frac{\bar{r}_1}{\bar{\bar{r}}_1} \left(w_b^{\frac{1}{2}} \frac{\bar{r}_1}{\bar{\bar{r}}_1} \right)^* = w_b \frac{\bar{r}_1^2}{\bar{\bar{r}}_1^2} = w_b \frac{\bar{u}_1}{\bar{\bar{u}}_1} \quad (27)$$

$$\bar{\mathbf{b}} = w_b^{\frac{1}{2}} \frac{\bar{r}_1}{\bar{\bar{r}}_1} \left(\mathbf{v}_b - \frac{v_{b1}}{\bar{r}_1} \bar{\mathbf{r}} \right)$$

$$= w_b^{\frac{1}{2}} \frac{\bar{r}_1}{\bar{\bar{r}}_1} \left(\mathbf{v}_b - \frac{v_{b1}}{\bar{u}_1} \bar{\mathbf{u}} \right) = \overline{w}_b^{\frac{1}{2}} \bar{\mathbf{v}}_b. \quad (28)$$

Stage 3: Rotate and to eliminate . To carry out these rotations, must be translated from V-space to U-space and must be translated to V-space, as described in

$$\mathbf{u}_a = \bar{a}_2^* \bar{\mathbf{a}}$$

$$= \left(w_a^{\frac{1}{2}} \frac{r_1}{\bar{r}_1} \left(v_2 - \frac{v_1}{u_1} u_2 \right) \right)^* \left(w_a^{\frac{1}{2}} \frac{r_1}{\bar{r}_1} \left(\mathbf{v} - \frac{v_1}{u_1} \mathbf{u} \right) \right)$$

$$= \overline{w}_a \bar{v}_2^* \bar{\mathbf{v}} \quad (29)$$

\
$$\mathbf{v}_b = w^{-\frac{1}{2}} \bar{\mathbf{b}} = w^{-\frac{1}{2}} \overline{w}_b^{\frac{1}{2}} \bar{\mathbf{v}}_b. \quad (30)$$

Therefore, if we let the updated *w*= , then the updated value of V= . This implies that the updated scale factor must be used to translate in order to use as the new V-space vector for further processing. The MSGR sequence of operations is illustrated in Figure. 2.

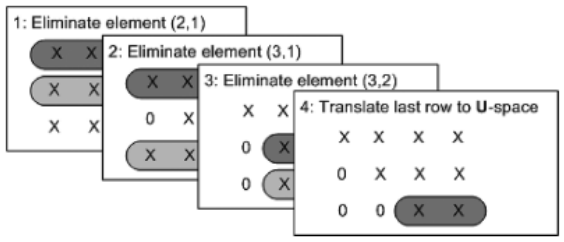

1: Eliminate element (2,1)
2: Eliminate element (3,1)
3: Eliminate element (3,2)
4: Translate last row to U-space

Figure 2: MSGR operation sequence.

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 (31)–(33) where is the th column being processed.

$$\bar{\mathbf{u}} = \mathbf{u} + w v_k^* \mathbf{v} \quad (31)$$

$$\bar{\mathbf{v}} = \mathbf{v} - \left(\frac{v_k}{u_k} \right) \mathbf{u} \quad (32)$$

$$\overline{w} = w \frac{u_k}{\bar{u}_k}. \quad (33)$$

These three stages define the sequence of operations of MSGR. There are, however, a number of specific conditions under which SGR cannot operate properly and produces erroneous results; this situation is maintained in MSGR. The nature of these cases and a resolution to the resulting operational issues is presented in Section III-B.

B. Processing Zero Values Incurred on the Matrix Diagonal

To this point, it has been assumed that . However, zeros can arise either in the input matrix or during phases of rotations, as illustrated in Figure 3. 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 the undesirable occurrence of zeros in the diagonal position of the lower rotated row. During the next phase of rotations, (i.e., rotations to zero the lower diagonal elements in the next column) the first element of the -space is zero, leading to incorrect operation of (31)–(33).

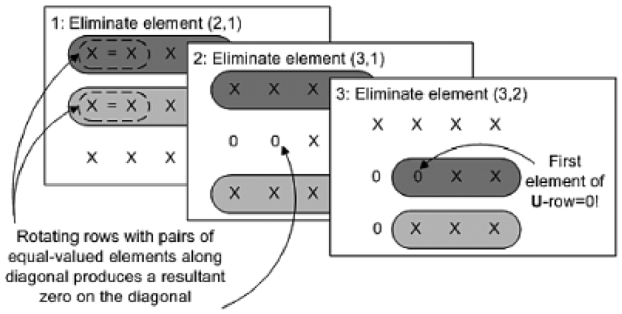

1: Eliminate element (2,1)
2: Eliminate element (3,1)
3: Eliminate element (3,2)
First element of U-row=0!
Rotating rows with pairs of equal-valued elements along diagonal produces a resultant zero on the diagonal

Figure 3: Diagonal zero element arisal.

Due to this, an operational caveat must be applied before further rotations can be carried out. In [16], Dohler proposed a solution for dealing with $u_k = 0$, as given in

$$\text{for } u_k = 0,\ v_k \neq 0 \begin{cases} \bar{\mathbf{u}} = w v_k^* \mathbf{v} \\ \bar{\mathbf{v}} = \text{arbitrary} \\ \overline{w} = 0 \end{cases}. \quad (34)$$

Despite this, the condition of both has not $u_k = v_k = 0$ previously been considered, and the solutions proposed in (34) prohibit further processing based on these updated values. For example, translation of a row from V-space to U -space (such as that described in (29)) is not possible using Dohler's suggested updated scale factor =0 . To overcome this problem, a novel solution $u_k = 0$ for

u_k 0 , which permits subsequent processing, as well as a solution for $u_k = v_k = 0$, which has not been previously considered, is given in

$$\text{for } u_k = 0,\ v_k \neq 0 \begin{cases} \overline{\mathbf{u}} = w v_k^* \mathbf{v} \\ \overline{\mathbf{v}} = -\mathbf{u} \\ \overline{w} = w \end{cases}$$
$$\text{for } u_k = 0,\ v_k = 0 \begin{cases} \overline{\mathbf{u}} = w \mathbf{v} \\ \overline{\mathbf{v}} = -\mathbf{u}\,. \\ \overline{w} = w \end{cases} \qquad (35)$$

This caveat fully defines MSGR-based matrix triangularization and has resolved the outstanding barriers to employing SGR for complex-valued matrix inversion. It remains to convert the triangularized matrix to the inverse, which requires a suitable back-substitution operation. This back-substitution process is outlined in Section III-C.
\

C. **MSGR-Based Matrix Inversion**

Recalling (4) and (9), MSGR decomposes the input matrix as A= Q_A , which may be inverted as A^{-1}= $U^{-1}(Q_A$ $^{-1}$. Inversion of the upper triangular matrix can be computed using back-substitution on the result of the MSGR operation, as in (36) where G= U^{-1}

$$G_{ij} = \begin{cases} -\frac{1}{u_{jj}}\left(\sum_{ik}^{j-1} G_{ik} u_{kj}\right) & i < j \\ \frac{1}{u_{ij}} & i = j \\ 0 & i > j. \end{cases} \qquad (36)$$

Given this back-substitution operation, according to (9), MSGR-based matrix inversion may be split into three sub-operations: decomposition of the input matrix A into the upper triangular matrix U, formation of $U^{-1}$from U via back-substitution, and finally multiplication by (Q_A $^{-1}$ to form the product $U^{-1}(Q_A$ $^{-1}$. We can use this to formulate an operational model of MSGR-based matrix inversion to complement the mathematical model described thus far.

If we reformulate (9) as U= (Q_A $^{-1}$A , then (Q_A $^{-1}$ can be considered to be the factor by which A is multiplied to produce U, an operation which is achieved in the MSGR algorithm using rotations. Therefore, this factor can be isolated by multiplying by the identity matrix I, which is equivalent to rotating I in the same manner as A. Therefore, processing immediately after produces the output (Q_A $^{-1}$. These two factors (U and (Q_A $^{-1}$) are produced by an *MSGR array.*

The formation of U^{-1} from U via back-substitution, and subsequent multiplication by (Q_A $^{-1}$ is then performed by an *Invert and Multiply (IAM) Array*. Dependence graphs of the MSGR and IAM arrays and the dataflow. The MSGR array produces an upper triangular matrix U given an input matrix A. It is also used to produce the component (Q_A $^{-1}$given an input identity matrix . It consists of two classes of operational unit: boundary cells which translate the input data vectors to U-space and internal cells for calculation of rotation angles and subsequent rotation of the input data. Note the bi-modal operation of these cells, which depends on whether the cell processes diagonal or off-diagonal elements—diagonal elements of the input matrix are tagged with a flag (e.g. ,) as shown in Figure 4. Furthermore, the zero-value operational caveat described in Section III-B (i.e., u_k =0) is evident in these cells. Similarly, the IAM array comprises boundary cells and internal cells which carry out both the inversion of U by back-substitution and subsequent multiplication by (Q_A $^{-1}$. These finalize the MSGR-based matrix inversion algorithm.

Whilst our primary motivating application for developing this new matrix inversion approach is MIMO systems, it is evident from the algorithm description that MSGR-based matrix inversion can be used for inversion of complex matrices in any application. Since it is derived from SGR, MSGR benefits from SGR's inherent numerical stability. To maximise the performance of embedded implementations of this algorithm, this complexity should be reduced so far as is possible without significantly reducing the accuracy of the result. In Section IV we describe a technique to achieve this and analyze the effect of this simplification on the accuracy of MSGR-based matrix inversion with respect to a number of other standard approaches and on the performance of a state-of-the-art MIMO receiver system.

IV. REDUCED COMPLEXITY MSGR-BASED MATRIXINVERSION FOR BLAST MIMO

A. **Reduced Complexity MSGR-Based Matrix Inversion***:*

In this section, we propose to remove the scale factor and its associated computations (as described in Section III-C) from the MSGR cells to reduce the complexity of MSGR-based matrix inversion. Using MSGR without the -factor, the matrix **A** is decomposed as in

$$\mathbf{A} = \mathbf{Q}_W \mathbf{U}_W$$

(37)

$\mathbf{Q}_W$ is not normally an unitary matrix. Since multiplication of the matrix rows by the scale factor ω in (20) and (31) does not affect the matrix properties, then the decomposition of using (37) compared to standard MSGR (4), does not influence the properties of **U** and **Uw** i.e., they are both invertible upper triangular matrices. Therefore, for MSGR without the ω factor, **Uw** is an invertible upper triangular matrix and the inversion of **A** can be given as in

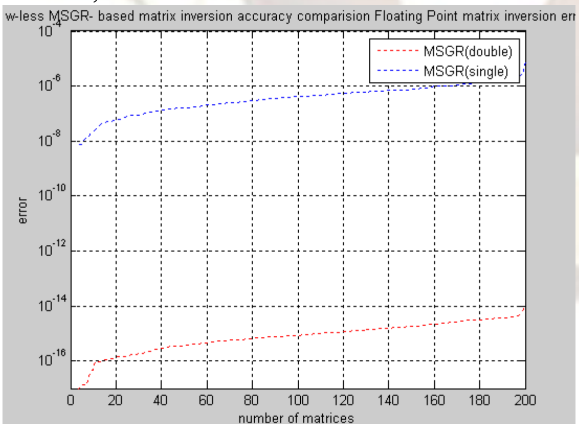
$$\mathbf{A}^{-1} = (\mathbf{Q}_W\mathbf{U}_W)^{-1} = \mathbf{U}_W^{-1}\mathbf{Q}_W^{-1}$$

(38)

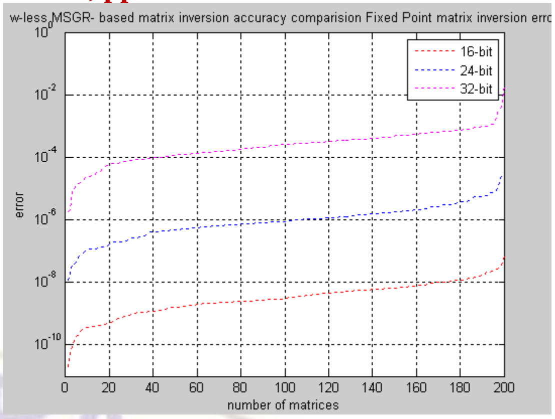
Once all ω -related computations have been removed, the MSGR array now produces an upper triangular matrix **Uw** given an input matrix **A** It also produces the component $\mathbf{Q}_W^{-1}$ given an input identity matrix . As **Uw** is an upper triangular matrix, its inversion can be found by back-substitution. This inversion and multiplication by $\mathbf{Q}_W^{-1}$ is carried out in the IAM array, as for the MSGR algorithm with -factor included. It compares the number of operations required for MSGR-based inversion of a 4 x 4 matrix when the –factor has been included and excluded. As this shows, removing the ω factor has the significant effect of reducing the number of multiplications and divisions by 19% and 18%, respectively. This represents a considerable performance advantage for real-time implementation, in particular the reduction in the number of complex division operations, since these are relatively expensive on implementation. However, whilst the complexity reduction offered by removing the ω factor is clearly attractive, it is only advantageous if it does not significantly reduce the accuracy of the resulting inverted matrices, or the accuracy of any MIMO receiver algorithm built upon MSGR-based matrix inversion. These two factors are analyzed in Section IV-B and -C, respectively

B. Reduced Precision MSGR Accuracy Analysis:

The effect of removing the -factor on the accuracy of the resulting inverted matrix is described in the graphs of Figure. 4. These two graphs measure the deviation of the product of the original matrix and its inverse from the matrix (**y** axis) for each of 200 4 x4 MIMO rich scattering Rayleigh-fading channel matrices, which are enumerated on the axis.



(a)



(b)

Figure 4. ω-less MSGR-based matrix inversion accuracy comparison. (a)Floating-point matrix inversion error. (b) Fixed-point matrix inversion error.

To help gauge the accuracy of these results, the errors encountered by alternative schemes are also included. For floating-point data, we compare with the results of complex matrix inversion as implemented using the LAPACK linear algebra package (in this case implemented in the NAG Matlab toolkit)in Figure 4(a), while Figure 4(b) compares the accuracy of fixed-point MSGR with CORDIC-based matrix inversion. The results are shown in order of ascending error in Figure 4, for easier interpretation. Figure 4(a) shows that the errors encountered in MSGR- based matrix inversion are, more or less, the same as those encountered using LAPACK, and also show the expected increase in absolute error when single precision data is used. Figure 4(b) shows a similar relationship between fixed-point MSGR and CORDIC, for various word size data. The dramatically larger errors encountered for a small number of matrices [represented by the spikes at the end of the distributions in Figure4(b)] are due to the occurrence of divide by zero operations, and occur consistently in both MSGR and CORDIC-based algorithms. However, the major trend to note from Figure 4 is that reduced complexity MSGR-based matrix inversion, without the factor, has minimal impact on the accuracy of the final solution, and provides solutions as accurate as the comparable techniques.

C. MSGR-Based Matrix Inversion in BLAST MIMO Receivers*:*

Real-time matrix inversion is vital in MIMO systems, for operations such as MMSE-based channel detection, as described by (39). Here we study the effect of removing the -factor on the BER of an MMSE-based BLAST receiver system, specifically a Turbo-BLAST (T-BLAST) scheme [3], [12]

$$\mathbf{x} = (\mathbf{H}^H\mathbf{H} + \sigma^2\mathbf{I})^{-1}\mathbf{H}^H\mathbf{r} \qquad (39)$$

Here **r** is the received signal, **H** is the channel matrix and x is the estimate of the transmitted signal. To illustrate the effect of removing the -factor, Figure 5 shows the BER performance of a 4 x 4 T-BLAST receiver on a variable signal-to-noise-ratio (SNR) rich scattering Rayleigh-fading channel, under a variety of word size constraints with **w** either included or excluded. There are a number of trends evident in Figure 5. The close proximity of the curves for the variants including and excluding the ω factor in Figure 5 (which in the single precision, 20 bit floating-point and 32 bit fixed-point cases make the curves indistinguishable) is that there is little degradation in the performance of the T-BLAST receiver whether the ω factor is included or excluded. Indeed, for 24 and 16 bit fixed-point, excluding the -factor actually lowers the BER at high SNR in this simulation. In addition, the figure shows the benefit of adopting reduced word size floating-point arithmetic—not only due to the resource and throughput advantages described in Section II, but also due to the clear accuracy advantage that 20 bit floating-point holds over even 32 bit fixed-point.

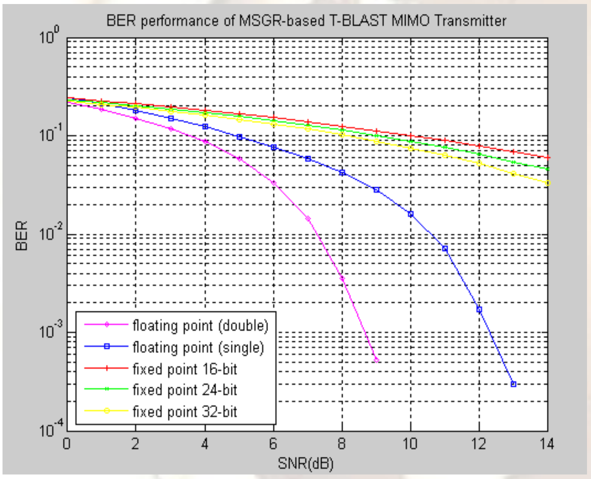


Figure 5: BER performance of MSGR-based T-BLAST MIMO transceiver

Finally, Figure. 5 shows that when 16 bit fixed-point data is employed, the receiver performance degrades to the point where reliable data transmission is not possible. Thus, MSGR-based matrix inversion exhibits considerable numerical stability with or without the -factor, and whilst not the least computationally complex option for matrix inversion, the reduction in computational complexity enabled by excluding the -factor, coupled with the high levels of data and task parallelism inherent in the algorithm, suggests considerable potential for high performance implementation of MSGR-based matrix inversion.

V CONCLUSION

Explicit matrix inversion is a major bottleneck in the design of embedded MIMO transceiver architectures. Until now, there has been no appropriate solution to this problem for state-of the- art MIMO systems, such as those in 802.11n systems, with the large disparities between required and actual performance indicating the need for a thorough review of both algorithms and architectures employed. The work presented in this paper has solved this problem. We have derived Modified Squared Givens' rotations (MSGR), an algorithm for QR-based matrix triangularization and inversion which overcomes deficiencies in the standard SGR algorithms. This provides a complex-valued matrix inversion method which not only overcomes key factors for integration in MIMO systems, but also enables a suitable mechanism for complex matrix inversion more generally. Moreover, we have shown that the computational complexity of the algorithm may be further reduced by almost 20% with minimal impact on the accuracy of the inverted matrix or the perceived BER of a BLAST MIMO receiver based upon it.

References

[1] Lei Ma, , Kevin Dickson,John McAllister, and John McCanny" QR Decomposition-Based Matrix Inversion for High Performance Embedded MIMO Receivers" IEEE Trans On Signal Processing, Vol. 59, No. 4, APRIL 2011

[2] P. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. URSI Int. Symp. Signals, Syst., Electron.*, 1998, pp. 295–300.

[3] M. Sellathurai and S. Haykin, "Turbo-blast for wireless communications: Theory and experiments," *Bell Lab. Tech. J.*, vol. 50, pp. 2538–2546, 2002.

[4] F. Echman and V. Owall, "A scalable pipelined complex valued matrix inversion architecture," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS 2005)*, May 2005, pp. 4489–4492.

[5] I. LaRoche and S. Roy, "An efficient regular matrix inversion circuit architecture for MIMO processing," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS 2006)*, May 2006, pp. 4819–4822.

[6] M. Myllyla, J.-H. Hintikka, J. Cavallaro, M. Juntti, M. Limingoja, and A. Byman, "Complexity analysis of MMSE detector architectures for MIMO OFDM systems," in *Conf. Rec. 39th Asilomar Conf. Signals, Syst., Comput. (2005)*, vol. 1, pp. 75–81.

[7] M. Karkooti, J. Cavallaro, and C. Dick, "FPGA implementation of matrix inversion using QRD-RLS algorithm," in *Conf. Rec. Asilomar Conf. Signals, Syst. Comput.*, 2005, pp. 1625–1629.

[8] H. S. Kim, W. Zhu, J. Bhatia, K. Mohammed, A. Shah, and B. Daneshrad, "A practical, hardware friendly MMSE detector for MIMO-OFDM-based systems," *EURASIP J. Adv. Signal Process.*, vol. 2008, pp. 1–14, Jan. 2008.

[9] M. Ylinen, A. Burian, and J. Takala, "Updating matrix inverse in fixed-point representation: Direct versus iterative methods," in *Proc. Int. Symp. System-on-Chip*, 2003, pp. 45–48.

[10] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[11] G. Lightbody, R.Walke, R.Woods, and J. McCanny, "Linear QR architecture for a single chip adaptive beamformer," *J. VLSI Signal Process. Syst. Signal, Image, and Video Technol.*, vol. 24, pp. 67–81, 2000.

[12] Z. Liu, J. McCanny, and R. Walke, "Generic Soc QR array processor for adaptive beamforming," *IEEE Trans. Circuits Syst., Part II—Analog Digit. Signal Process.*, vol. 50, no. 4, pp. 169–175, Apr. 2003.

[13] R. Hamill, J. McCanny, and R. Walke, "On-line CORDIC algorithm and VLSI architecture for implementing QR-array processors," *IEEE Trans. Signal Process.*, vol. 48, no. 2, pp. 592–598, Feb. 2000.

[14] 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.

[15] G. Golub, "Numerical methods for solving least-squares problems,"*Num. Math.*, vol. 7, pp. 206–216, 1965.

[16] R. Dohler, "Squared givens rotations," *IMA J. Numer. Anal.*, vol. 11,pp. 1–5, 1991.

[17] J. E. Volder, "The cordic trigonometric computing technique," *Instit.Radio Eng. Trans. Electron. Comput.*, vol. EC-8, pp. 330–334, 1959.

[18] M. Myllyla, M. Juntti, M. Limingoja, A. Byman, and J. Cavallaro, "Performance evaluation of two LMMSE detectors in aMIMO-OFDM hardware testbed," in *Conf. Rec. Asilomar Conf. Signals, Syst. Comput.*, October 2006, pp. 1161–1165.

[19] R. Walke, "High sample rate givens rotations for recursive least squares," Ph.D. dissertation, The Univ. Warwick, Warwick, 1997.

[20] G. Golub and C. van Loan, *Matrix Computations*. Baltimore, MD:The Johns Hopkins Univ. Press, 1996.

[21] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proc. IEEE Conf. Acoust., Speech, Signal Process.*, 2000, vol. 2, pp. 737–740.

[22] M. Sellathurai and S. Haykin, "T-BLAST for wireless communications: First experimental results," *IEEE Trans. Veh. Technol.*, vol. 52, no. 3, pp. 530–535, 2003.

[23] R. Walke, R. Smith, and G. Lightbody, "Architectures for adaptiveweight calculation on ASIC and FPGA," in *Conf. Rec. Asilomar Conf. Signals, Syst. Comput.*, 1999, pp. 1375–1380.