A polynomial matrix QR decomposition with application to MIMO channel equalisation

A Polynomial Matrix QR Decomposition with Application to MIMO Channel Equalisation

Joanne Foster[1] and John McWhirter
Centre of Digital Signal Processing
School of Engineering
Cardiff University, UK
[1] Email: fosterj1@Cardiff.ac.uk

Jonathon Chambers
Advanced Signal Processing Group
Department of Electronic and Electrical Engineering
Loughborough University, UK

***Abstract*—An algorithm for computing the QR decomposition of a polynomial matrix is introduced. The algorithm proceeds to perform the decomposition by following the same strategy in eliminating entries of the matrix as is used in the Givens method for a QR decomposition of a scalar matrix, however polynomial Givens rotations are now required. A possible application of the decomposition is in MIMO communications, where it is often required to reconstruct data sequences that have been distorted due to the effects of co-channel interference and multipath propagation, leading to intersymbol interference. If the channel matrix for the system is known, its QR decomposition can be calculated and used to transform the MIMO channel equalisation problem into a set of single channel problems, which can then be solved using a maximum likelihood sequence estimator. Some simulated average bit error rate results are presented to support the potential application to MIMO channel equalisation.**

I. Introduction

Polynomial matrices have many applications in the field of control, but in recent years they have also been used extensively in the areas of digital signal processing and communications. Examples of their applications include broadband adaptive sensor array processing, MIMO communication channels, broadband subspace decomposition and also digital filter banks for subband coding or data compression [1], [2]. In the context of this paper, polynomial matrices arise when a set of signals are received at an array of sensors over multiple paths and with different time delays. This is referred to as convolutive mixing and the mixing or channel matrix required to express this takes the form of a polynomial matrix, where each element is a finite impulse response (FIR) filter.

If, instead, the received signals are instantaneously mixed, then there are no time delays in the propagation of the signals from the sources to the sensors and a matrix of scalar entries is sufficient to describe the mixing. In this situation, provided the channel matrix is known and is of full column rank, its QR decomposition can be formulated. Decomposing the channel matrix and exploiting the upper triangular structure of the resulting matrix, the set of source signals can easily be determined from the received signals using back substitution. This technique can be extended to broadband signal processing, where polynomial matrices are now observed, by using a suitable algorithm for computing the QR decomposition of a polynomial matrix. Currently, no other techniques of achieving this decomposition of a polynomial matrix exist.

In communication applications it is often necessary to estimate a set of source signals from a set of received signals, where the channel matrix for the system has previously been estimated by passing a training sequence through the system. This problem is termed as multi-input multi-output (MIMO) channel equalisation. The QR decomposition of the polynomial channel matrix could be calculated to transform the MIMO channel equalisation problem into a set of single-input single-output (SISO) channel equalisation problems, which can then be solved using a maximum likelihood sequence estimator.

This paper firstly discusses polynomial matrices and the QR decomposition of a scalar matrix. Subsequently, the concept of a polynomial Givens Rotation is introduced, before detailing an algorithm for computing the polynomial QR decomposition (PQRD) of a matrix. Finally, the problem of MIMO channel equalisation is discussed and through simulations the possible application of the PQRD algorithm to this problem is demonstrated.

II. Polynomial Matrices and Notation

A polynomial matrix is simply a matrix with polynomial elements. However, it can alternatively be thought of as a polynomial with matrix coefficients and so a polynomial matrix $\underline{\mathbf{A}}(z)$, where the indeterminate variable of the polynomial is z^{-1}, used to represent a unit delay, can be expressed as

$$\begin{aligned}\underline{\mathbf{A}}(z) &= \sum_{\tau=t_1}^{t_2} \mathbf{A}(\tau) z^{-\tau} \\ &= \begin{bmatrix} \underline{a}_{11}(z) & \underline{a}_{12}(z) & \cdots & \underline{a}_{1q}(z) \\ \underline{a}_{21}(z) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \underline{a}_{p1}(z) & \cdots & \cdots & \underline{a}_{pq}(z) \end{bmatrix} \end{aligned} \quad (1)$$

where $\tau \in \mathbb{Z}$, $t_1 \leq t_2$ and $\mathbf{A}(\tau) \in \mathbb{C}^{p \times q}$ is the matrix of coefficients of $z^{-\tau}$. The polynomial coefficient in the $(i,j)^{th}$ element of $\underline{\mathbf{A}}(z)$ corresponding to a delay of z^{-t} will be denoted as $a_{ij}(t)$ and the order of the polynomial matrix can be calculated as $(t_2 - t_1)$, where the values of the parameters t_1 and t_2 are not necessarily positive. The underline notation

978-1-4244-2110-7/08/$25.00 ©2007 IEEE

in (1) is used to denote a polynomial, whether it is a matrix, vector or scalar, to avoid confusion with the notation used for the z-transform of a variable. Let the set of polynomial matrices, with complex coefficients, be denoted by $\underline{\mathbb{C}}^{a\times b}$ where a and b denote the number of rows and columns in the matrix respectively.

The paraconjugate of the polynomial matrix $\underline{\mathbf{A}}(z)$ is defined to be

$$\widetilde{\underline{\mathbf{A}}}(z) = \underline{\mathbf{A}}_*^T(1/z) \tag{2}$$

where * denotes the complex conjugation of the coefficients of each polynomial element and ^{T} denotes matrix transposition. The tilde notation, as demonstrated in (2), will be used to denote paraconjugation. A polynomial matrix $\underline{\mathbf{A}}(z)$ is said to be paraunitary if the following statement is true

$$\underline{\mathbf{A}}(z)\widetilde{\underline{\mathbf{A}}}(z) = \widetilde{\underline{\mathbf{A}}}(z)\underline{\mathbf{A}}(z) = \mathbf{I}. \tag{3}$$

Finally, the Frobenius norm, or F-norm, of the polynomial matrix $\underline{\mathbf{A}}(z) \in \underline{\mathbb{C}}^{p\times q}$ is defined as

$$\left\|\underline{\mathbf{A}}(z)\right\|_F = \sqrt{\sum_{\tau=t_1}^{t_2}\sum_{i=1}^{p}\sum_{j=1}^{q}\left|a_{ij}(\tau)\right|^2}. \tag{4}$$

III. The QR Decomposition of a Scalar Matrix

The QR decomposition of a matrix, $\mathbf{A} \in \mathbb{C}^{p\times q}$, whose elements are complex scalars is defined as

$$\mathbf{A} = \mathbf{QR} \tag{5}$$

where $\mathbf{Q} \in \mathbb{C}^{p\times p}$ is a unitary matrix and $\mathbf{R} \in \mathbb{C}^{p\times q}$ is an upper triangular matrix. One method for computing the unitary matrix $\mathbf{Q}$ is by calculating a series of plane rotations, where each rotation will drive one of the elements beneath the diagonal of the matrix $\mathbf{A}$ to zero, [3]. The elements of the matrix below the diagonal are rotated to equal zero in a particular order to ensure that each element need only be eliminated once. There are several different orderings that can be implemented. However, for the purposes of this paper the elements are eliminated starting with the uppermost left element and then moving across all elements beneath the diagonal in each row from left to right, before moving to the next row down.

This technique of eliminating the elements beneath the diagonal in a specified order can be directly applied to polynomial matrices, even though each element now consists of a series of polynomial coefficients. All coefficients from the series need to be eliminated to ensure that the polynomial element is zero. Unfortunately, this can no longer be achieved with a scalar Givens rotation matrix; instead a polynomial Givens rotation is required.

IV. Polynomial Givens Rotations

A. Elementary Polynomial Givens Rotation

An elementary polynomial Givens rotation (EPGR) is a polynomial matrix that can be applied to either a polynomial vector or matrix to selectively zero one coefficient of a polynomial element. It takes the form of a Givens rotation preceded by an elementary time shift matrix as follows

$$\begin{aligned}\widehat{\underline{\mathbf{G}}}^{(\alpha,\theta,\phi,t)}(z) &= \begin{pmatrix} ce^{i\alpha} & se^{i\phi} \\ -se^{-i\phi} & ce^{-i\alpha}\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & z^t\end{pmatrix}\\ &= \begin{pmatrix} ce^{i\alpha} & se^{i\phi}z^t \\ -se^{-i\phi} & ce^{-i\alpha}z^t\end{pmatrix}\end{aligned} \tag{6}$$

where c and s define the cosine and sine of the angle θ respectively. The aim of this matrix, when applied to a polynomial vector $\underline{\mathbf{a}}(z) \in \underline{\mathbb{C}}^{2\times 1}$ as demonstrated

$$\begin{pmatrix} ce^{i\alpha} & se^{i\phi}z^t \\ -se^{-i\phi} & ce^{-i\alpha}z^t\end{pmatrix}\begin{pmatrix}\underline{a}_1(z)\\ \underline{a}_2(z)\end{pmatrix} = \begin{pmatrix}\underline{a}_1'(z)\\ \underline{a}_2'(z)\end{pmatrix} \tag{7}$$

is to drive a specified coefficient from the polynomial element $\underline{a}_2(z)$ to zero. For example, to zero the coefficient $a_2(\tau)$, then the lag parameter in the EPGR matrix is set as $t = \tau$ and the rotation angles are chosen such that

$$\tan(\theta) = \frac{|a_2(\tau)|}{|a_1(0)|},\ \phi = -arg(a_2(\tau)),\ \alpha = -arg(a_1(0)) \tag{8}$$

thus resulting in $a_2'(0) = 0$. Furthermore, following the application of the EPGR the coefficient $a_1'(0)$ is real and $|a_1'(0)|^2 = |a_1(0)|^2 + |a_2(\tau)|^2$. Note that an EPGR matrix is paraunitary by construction as each component of the matrix, i.e. the Givens rotation and the elementary time shift matrix, are both paraunitary. Furthermore this matrix is norm preserving and so

$$\left\|\widehat{\underline{\mathbf{G}}}^{(\alpha,\theta,\phi,t)}(z)\underline{\mathbf{a}}(z)\right\|_F = \left\|\underline{\mathbf{a}}(z)\right\|_F. \tag{9}$$

B. Complete Polynomial Givens Rotation

A series of EPGRs can be applied iteratively to the polynomial vector $\underline{\mathbf{a}}(z) \in \underline{\mathbb{C}}^{2\times 1}$, as demonstrated by (7), to drive all coefficients of the polynomial element $\underline{a}_2(z)$ arbitrarily close to zero. At each iteration the rotation angles θ, ϕ and α and the lag parameter t are chosen to zero the coefficient within $\underline{a}_2(z)$ with maximum magnitude, this coefficient will be referred to as the dominant coefficient. If this coefficient is not unique, then any of the dominant coefficients from the element may be chosen. The complete series of EPGRs required, constitutes a complete polynomial Givens rotation (CPGR), which will be denoted by the matrix $\underline{\mathbf{G}}^{(2,1)}(z)$, where the superscripts denote the position of the polynomial element that the matrix is attempting to annihilate. A matrix of this form can be applied to a polynomial vector $\underline{\mathbf{a}}(z) \in \underline{\mathbb{C}}^{2\times 1}$ such that

$$\underline{\mathbf{G}}^{(2,1)}(z)\begin{bmatrix}\underline{a}_1(z)\\ \underline{a}_2(z)\end{bmatrix} \cong \begin{bmatrix}\underline{a}_1'(z)\\ 0\end{bmatrix} \tag{10}$$

where all coefficients of the polynomial element $\underline{a}_2(z)$ have been driven arbitrarily close to zero over a series of EPGRs.

In practice it is often not feasible to zero all coefficients of a

polynomial element, instead the coefficients are driven to zero until the magnitude of all coefficients in $\underline{a}_2(z)$ are sufficiently small and the following stopping condition is satisfied

$$|a_2(t)| < \epsilon \tag{11}$$

$\forall t \in \mathbb{Z}$ where $\epsilon > 0$ is a pre-specified small value. Convergence of a CPGR can easily be proved and is detailed in [4]. Note also that each of the EPGRs is paraunitary and so the complete polynomial Givens rotation will also be paraunitary. Furthermore, as the transformation is paraunitary, it is also norm preserving and so $\|\underline{\mathbf{a}}(z)\|_F = \left\|\underline{\mathbf{G}}^{(2,1)}(z)\underline{\mathbf{a}}(z)\right\|_F$.

A CPGR can similarly be applied to a polynomial matrix to drive one of the polynomial elements beneath the diagonal of the matrix to zero. This technique forms the basis of the algorithm for calculating the QR decomposition of a polynomial matrix.

V. The QR Decomposition of a Polynomial Matrix

The PQRD by Steps algorithm is a technique for factorising a polynomial matrix into an upper triangular and a paraunitary polynomial matrix. Let $\underline{\mathbf{A}}(z) \in \underline{\mathbb{C}}^{p\times q}$, then the objective of the algorithm is to calculate a paraunitary matrix $\underline{\mathbf{Q}}(z) \in \underline{\mathbb{C}}^{p\times p}$ such that

$$\underline{\mathbf{Q}}(z)\underline{\mathbf{A}}(z) = \underline{\mathbf{R}}(z) \tag{12}$$

where $\underline{\mathbf{R}}(z) \in \underline{\mathbb{C}}^{p\times q}$ is an upper triangular polynomial matrix. The polynomial matrix $\underline{\mathbf{Q}}(z)$ is computed as a series of complete polynomial Givens rotations, each one designed to drive all coefficients of one of the polynomial elements situated beneath the diagonal to be sufficiently small. The order in which the polynomial elements are eliminated is the same as that of the scalar Givens QR method explained in section III. This algorithm can therefore be thought of as an extension of the conventional Givens QR method for factorising a scalar matrix by applying a series of unitary rotation matrices.

A. The PQRD by Steps Algorithm

The algorithm operates as a series of ordered steps, where at each step all coefficients of one polynomial element situated beneath the diagonal of the matrix $\underline{\mathbf{A}}(z)$ are driven sufficiently small by applying the appropriate CPGR. The transformation observed, to drive all coefficients of the polynomial element $\underline{a}_{jk}(z)$ sufficiently small, is of the form

$$\underline{\mathbf{A}}^{'}(z) = \underline{\mathbf{G}}^{(j,k)}(z)\underline{\mathbf{A}}(z) \tag{13}$$

where $\underline{\mathbf{G}}^{(j,k)}(z)$ is the complete polynomial Givens rotation designed to drive all coefficients of the polynomial element $\underline{a}_{jk}(z)$ sufficiently small.

Each step however, operates as an iterative process, where at each iteration an EPGR matrix is applied to the polynomial matrix $\underline{\mathbf{A}}(z)$ to zero the dominant coefficient of the $(j,k)^{th}$ polynomial element. At each iteration the EPGR is formulated as a $p \times p$ identity matrix with the exception of the four elements positioned at the intersection of rows j and k with columns j and k. These elements are given by the 2×2 submatrix $\widehat{\underline{\mathbf{G}}}^{(\alpha,\theta,\phi,t)}(z)$ demonstrated in (6), where, if for example the dominant coefficient is $a_{jk}(\tau)$, then the lag parameter is set as $t = \tau$ and the coefficients required for calculating the rotation angles in (8) now correspond to $a_2(\tau) = a_{jk}(\tau)$ and $a_1(0) = a_{kk}(0)$. The effect of this transformation is to firstly shift the dominant coefficient so that it becomes the coefficient of z^0 and then apply the appropriate rotation so that the coefficient becomes equal to zero. Also as a result of this transformation $|a'_{kk}(0)|^2 = |a_{jk}(\tau)|^2 + |a_{kk}(0)|^2$ and $a'_{kk}(0)$ is real.

The iterative process is repeated until all coefficients from the polynomial element $\underline{a}_{jk}(z)$ are sufficiently small and satisfy the stopping condition defined by (11). To begin the subsequent step of the algorithm, $\underline{\mathbf{A}}(z)$ is replaced with $\underline{\mathbf{A}}^{'}(z)$ and the indices j and k are amended appropriately, moving to the next polynomial element in the ordering.

Following i steps of the algorithm, the transformation is of the form

$$\underline{\mathbf{A}}_i(z) = \underline{\mathbf{Q}}_i(z)\underline{\mathbf{A}}(z) \tag{14}$$

where $\underline{\mathbf{Q}}_i(z)$ is the product of i CPGR matrices and will be paraunitary by construction.

Once all elements beneath the diagonal in the matrix have been visited, this completes one sweep of the algorithm. The QR decomposition for scalar matrices drives all elements below the diagonal to zero. However, the Polynomial QR decomposition algorithm, although driving the dominant coefficient at each iteration to zero, only ensures that all coefficients of an element are suitably small before moving on to the next polynomial element in the ordering. Therefore, through future rotations of the algorithm, these small coefficients could be rotated with other suitably small coefficients, forcing them to increase in magnitude and so multiple sweeps of the algorithm may be required. Convergence of the PQRD by Steps algorithm can be easily deduced from the proof of convergence for the SBR2 algorithm, [1].

B. Truncation Method

With the application of each elementary delay matrix, at each iteration of each step of the algorithm, the order of the polynomial matrices $\underline{\mathbf{Q}}_i(z)$ and $\underline{\mathbf{A}}_i(z)$ increases, often after a series of iterations becoming unnecessarily large. Therefore throughout the algorithm the polynomial matrices are truncated, to stop their orders from becoming unnecessarily large and the algorithm becoming slow to implement. The method of truncation has been previously used in [4] and is similar to that developed in [1], [5]. A more detailed explanation as to why the order can become unnecessarily large is also found in these papers. A suitable truncation method for a polynomial matrix $\underline{\mathbf{A}}(z) \in \underline{\mathbb{C}}^{p\times q}$, with coefficient matrices $\mathbf{A}(t) \in \mathbb{C}^{p\times q}$ for $t = t_1, \ldots, t_2$ can be implemented as follows: find a maximum value for T_1 and a minimum value for T_2 such that

$$\frac{\sum_{\tau=t_1}^{T_1}\sum_{l=1}^{p}\sum_{m=1}^{q}|a_{lm}(\tau)|^2}{\|\underline{\mathbf{A}}(z)\|_F^2} \leq \frac{\mu}{2} \tag{15}$$

and

$$\frac{\sum_{\tau=T_2}^{t_2}\sum_{l=1}^{p}\sum_{m=1}^{q}|a_{lm}(\tau)|^2}{\|\underline{\mathbf{A}}(z)\|_F^2} \leq \frac{\mu}{2} \tag{16}$$

where μ defines the proportion of $\|\underline{\mathbf{A}}(z)\|_F^2$ permitted to be truncated from the polynomial matrix $\underline{\mathbf{A}}(z)$, with one implementation of the truncation method. The coefficient matrices $\mathbf{A}(\tau)$ for $\tau = t_1, \ldots, T_1$ and $\tau = T_2, \ldots, t_2$ can subsequently be trimmed from the matrix.

VI. Application of the Algorithm to MIMO Channel Equalisation

It is assumed that a set of source signals $\mathbf{s}(t) \in \mathbb{C}^{q\times 1}$ for $t \in \{0, \ldots, T-1\}$ are emitted from q independent sources through a convolutive channel, to be received at an array of p sensors, where it is assumed that $p \geq q$. In communications source signals are generally drawn from a finite constellation, such as binary or quaternary phase shift keying, BPSK/QPSK. The mixing model for the set of convolutively mixed signals, $\mathbf{x}(t) \in \mathbb{C}^{p\times 1}$ where $t \in \{0, \ldots, T-1\}$, can be expressed as

$$\mathbf{x}(t) = \sum_{k=0}^{N} \mathbf{C}(k)\mathbf{s}(t-k) + \mathbf{n}(t) \tag{17}$$

where $\underline{\mathbf{C}}(z) = \sum_{k=0}^{N} \mathbf{C}(k)z^{-k}$ denotes the polynomial channel matrix where each of the coefficient matrices $\mathbf{C}(k) \in \mathbb{C}^{p\times q}$ for $k \in \{0, \ldots, N\}$ and $\mathbf{n}(t) \in \mathbb{C}^{p\times 1}$ denotes an additive Gaussian noise process with variance $\sigma^2\mathbf{I}$. The mixing model of (17) can also be written in the form

$$\underline{\mathbf{x}}(z) = \underline{\mathbf{C}}(z)\underline{\mathbf{s}}(z) + \underline{\mathbf{n}}(z). \tag{18}$$

The QR decomposition of the channel matrix, $\underline{\mathbf{C}}(z) \in \underline{\mathbb{C}}^{p\times q}$, can be calculated so $\underline{\mathbf{Q}}(z)\underline{\mathbf{C}}(z) = \underline{\mathbf{R}}(z)$ and the convolutive mixing model expressed in (18) can be rewritten as

$$\underline{\mathbf{x}}'(z) = \underline{\mathbf{R}}(z)\underline{\mathbf{s}}(z) + \underline{\mathbf{n}}'(z) \tag{19}$$

where $\underline{\mathbf{x}}'(z) = \underline{\mathbf{Q}}(z)\underline{\mathbf{x}}(z)$ and $\underline{\mathbf{n}}'(z) = \underline{\mathbf{Q}}(z)\underline{\mathbf{n}}(z)$. Note that as $\underline{\mathbf{Q}}(z)$ is paraunitary, $\underline{\mathbf{n}}'(z)$ is also a Gaussian noise process with identical spectral properties. Now provided the channel matrix is of full column rank, the MIMO channel equalisation problem can be transformed into a set of q single channel equalisation problems using back substitution. Starting with the q^{th} element of $\underline{\mathbf{x}}'(z)$, this can be expressed as

$$\underline{x}_q'(z) = \underline{r}_{qq}(z)\underline{s}_q(z) + \underline{n}_q'(z), \tag{20}$$

which is a single channel equalisation problem. This can now be solved, to obtain an estimate of the q^{th} source signal, $\hat{s}_q(t)$, using a Maximum Likelihood Sequence Estimator (MLSE) based on the Viterbi algorithm, [6]. Furthermore, the i^{th} single channel equalisation problem can be formulated as

$$\underline{x}_i'(z) - \sum_{j=i+1}^{q} \underline{r}_{ij}(z)\underline{s}_j(z) = \underline{r}_{ii}(z)\underline{s}_i(z) + \underline{n}_i'(z), \tag{21}$$

which, provided the set of signals are estimated according to the ordering $i = q, q-1, \ldots, 1$, then each is a single channel equalisation problem. Each SISO equalisation problem can then be solved to obtain an estimate of the i^{th} source signal $\hat{s}_i(t)$ using the previously estimated source signals $\hat{s}_j(t)$ for $j = i+1, \ldots, q$.

VII. Results

In this section, the proposed application of the algorithm is illustrated, but firstly the PQRD by Steps algorithm is demonstrated. A polynomial channel matrix $\underline{\mathbf{C}}(z) \in \underline{\mathbb{C}}^{4\times 3}$ was generated, specifically designed to demonstrate the propagation of three signals onto four sensors. Each of the polynomial elements of the matrix was a fourth order FIR filter, where both the real and imaginary parts of the filter coefficients were drawn from a uniform distribution in the range [-1,1]. The matrix was then normalised so that $\|\underline{\mathbf{C}}(z)\|_F = 1$.

Firstly, the QR decomposition of the polynomial channel matrix $\underline{\mathbf{C}}(z)$ was obtained using the PQRD by Steps algorithm, where the truncation parameter and the stopping criterion were set as $\mu = 10^{-3}$ and $\epsilon = 10^{-3}$ respectively. Only a single sweep of the algorithm was required to ensure the stopping condition demonstrated by (11) was satisfied, requiring a total of 218 iterations over six steps. The upper triangular matrix $\underline{\mathbf{R}}(z)$ and the paraunitary transformation matrix $\underline{\mathbf{Q}}(z)$ obtained from the algorithm can be seen in Figures 1 and 2 respectively, where a stem plot has been used to demonstrate the magnitude of the series of coefficients for each of the polynomial elements. The position of the stem plot in each figure corresponds to the position of the polynomial element, which it represents within the matrix.

The relative error for the decomposition can then be calculated to ensure that by using the truncation method and by setting all non-zero coefficients beneath the diagonal of $\underline{\mathbf{R}}(z)$ to zero, required for the MIMO channel equalisation application of the algorithm, the accuracy of the decomposition obtained has not been compromised. The relative error is defined to be

$$E_{rel} = \frac{\left\|\underline{\mathbf{C}}(z) - \widetilde{\underline{\mathbf{Q}}}(z)\widehat{\underline{\mathbf{R}}}(z)\right\|_F}{\|\underline{\mathbf{C}}(z)\|_F} \tag{22}$$

where $\widehat{\underline{\mathbf{R}}}(z)$ is our upper triangular matrix $\underline{\mathbf{R}}(z)$ with all elements beneath the diagonal set equal to zero. This measure was found to be 0.168.

Three independent BPSK source signals of length 1000 were then generated and convolutively mixed following the mixing model demonstrated by (17) using the channel matrix $\underline{\mathbf{C}}(z)$, where N defines the order of polynomial matrix and for this example is equal to four. Randomly generated Gaussian noise representative of thermal noise, with spatial covariance matrix $\sigma^2\mathbf{I}$, was then added to each of the receive sensors, to give a desired received Signal-to-Noise Ratio (RSNR). The

RSNR for the experiment can be calculated as

$$\text{RSNR} = 10\log_{10}\left(\frac{Tr\left\{\underline{\mathbf{C}}(z)\widetilde{\underline{\mathbf{C}}}(z)\right\}\Big|_{t=0}}{4\sigma^2}\right) \qquad (23)$$

where 4 in the denominator defines the number of receivers and $|_{t=0}$ denotes the matrix containing the coefficients of z^0 of the polynomial matrix. The average bit error rate (BER) for each of the estimated source signals was calculated, where the variance of the additive noise was chosen to give varying levels of RSNR. This was carried out for 100 independent realisations, using the same channel matrix $\underline{\mathbf{C}}(z)$, but generating new source signals and noise terms for each realisation. The average of these results can be seen in Table I.

VIII. Conclusion

A QR algorithm suitable for polynomial matrices has been introduced. In a similar approach to the Givens QR method for scalar matrices, the algorithm operates as a series of ordered steps. However, unlike the scalar method, each step in the polynomial QR algorithm must now operate as an iterative process. This algorithm has been proven to converge and other techniques for calculating this decomposition have been developed. The potential application of the decomposition in multi-channel equalisation has been explored and for a quasi-static channel shown to yield good error rate performance.

References

[1] J. G. McWhirter, P. D. Baxter, T. Cooper, S. Redif and J. Foster, *An EVD Algorithm for Para-Hermitian Polynomial Matrices* IEEE Transactions on Signal Processing, 55(6), pp. 2158-2169, 2007.

[2] P. P. Vaidyanathan, *Multirate Systems and Filter Banks* Prentice Hall, 1993.

[3] G. H. Golub and C. F. Van Loan, *Matrix Computations (Third Edition)* The John Hopkins University Press, 1996.

[4] J. Foster, J. G. McWhirter and J. Chambers, *An Algorithm for Computing the QR Decomposition of a Polynomial Matrix* 15th International Conference on Digital Signal Processing, Cardiff, 2007.

[5] J. Foster, J. G. McWhirter and J. Chambers, *Limiting the Order of Polynomial Matrices Within the SBR2 Algorithm* IMA Conference Mathematics in Signal Processing, Cirencester, 2006.

[6] J. G. Proakis, *Digital Communications (Fourth Edition)* McGraw-Hill Book Co., 2001.

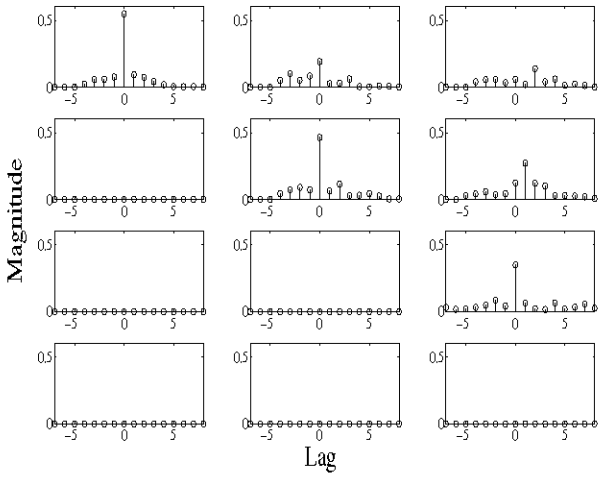

Fig. 1. The polynomial elements of the upper triangular polynomial matrix $\underline{\mathbf{R}}(z)$, obtained when the PQRD by Steps algorithm was applied to the channel matrix $\underline{\mathbf{C}}(z)$.

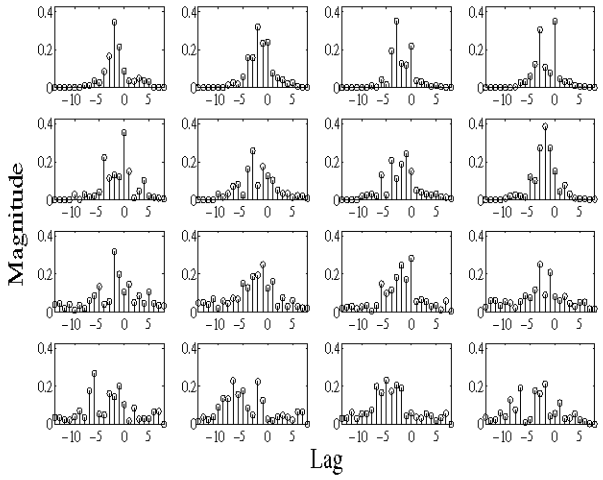

Fig. 2. The polynomial elements of the paraunitary transformation matrix $\underline{\mathbf{Q}}(z)$ obtained using the PQRD by Steps algorithm with input matrix $\underline{\mathbf{C}}(z)$.

TABLE I
AVERAGE BERS FOR THE THREE ESTIMATED SOURCES FOR THE MIMO CHANNEL EQUALISATION PROBLEM FOR VARYING LEVELS OF RSNR.

RSNR (dB)	Average BER		
	Source 1	**Source 2**	**Source 3**
-5	0.1914	0.1938	0.2693
0	0.0644	0.0729	0.1470
5	0.0032	0.0056	0.0319
10	0	0	0.0007
15	0	0	0
20	0	0	0