A NOVEL ALGORITHM FOR CALCULATING THE QR DECOMPOSITION OF A POLYNOMIAL MATRIX

Joanne Foster and Jonathon Chambers

Advanced Signal Processing Group,
Loughborough University, UK.

John McWhirter

Centre of Digital Signal Processing,
Cardiff University, UK.

ABSTRACT

A novel algorithm for calculating the QR decomposition (QRD) of polynomial matrix is proposed. The algorithm operates by applying a series of polynomial Givens rotations to transform a polynomial matrix into an upper-triangular polynomial matrix and, therefore, amounts to a generalisation of the conventional Givens method for formulating the QRD of a scalar matrix. A simple example is given to demonstrate the algorithm, but also illustrates two clear advantages of this algorithm when compared to an existing method for formulating the decomposition. Firstly, it does not demonstrate the same unstable behaviour that is sometimes observed with the existing algorithm and secondly, it typically requires less iterations to converge. The potential application of the decomposition is highlighted in terms of broadband multi-input multi-output (MIMO) channel equalisation.

***Index Terms*—** Paraunitary matrix, polynomial matrix QR decomposition, broadband MIMO channel equalisation.

1. INTRODUCTION

Polynomial matrices have many potential 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, the description of 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 are used to describe a convolutive mixing process, which occurs, for example, when a set of signals arrive at an array of sensors via multiple paths. This will result in the received signals consisting of weighted and delayed versions of the transmitted (source) signals. The 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 a matrix of scalars is sufficient to describe the mixing. In this situation, provided the channel matrix of the system is known and of full rank, its QRD can be calculated, thus transforming the channel matrix into an upper triangular matrix by means of a unitary transformation [3]. Decomposing the matrix in this way and exploiting the upper triangular structure of the transformed matrix, the set of source signals can be easily retrieved from the received signals using back substitution [3]. This technique can be extended to broadband signal processing, where polynomial matrices are now observed using the novel algorithm, introduced in this paper, for calculating the QRD of a polynomial matrix (PQRD). In this case the received signals will have been distorted due to the effects of co-channel interference (CCI) and multipath propagation, which then leads to intersymbol interference (ISI). By using a suitable algorithm for calculating the PQRD the observed CCI can be removed. The remaining ISI can then be removed using a standard single-input single-output equalisation scheme [4]. Note that conventional methods for calculating the QRD of a scalar matrix, such as Givens rotation [3], cannot directly be applied to determine the decomposition of a polynomial matrix, where each element now consists of a series of coefficients.

The authors have previously proposed an algorithm for calculating this decomposition known as the the PQRD by steps (PQRD-BS) algorithm [4, 5]. However, the algorithm proposed in this paper typically requires fewer iterations to converge than the existing algorithm. Furthermore, the existing PQRD-BS algorithm has demonstrated unstable behaviour as the algorithm converges [6]. This behaviour does not occur with the algorithm proposed in this paper due to the addition of an inverse delay matrix.

The remainder of this paper is organised as follows. Firstly, the notation and terminology surrounding polynomial matrices are introduced. A polynomial Givens rotation is then introduced, before a detailed description of the PQRD by columns (PQRD-BC) algorithm. Finally, through a simple example the capability and improved performance of the PQRD-BC over the existing PQRD-BS algorithm is confirmed by simulation for a particular polynomial matrix.

1.1. Polynomial Matrix Notation

A polynomial matrix is simply a matrix with polynomial elements or can alternatively be thought of as a polynomial with matrix coefficients. In the context of this paper, the indeterminate variable of each of the polynomial elements is z^{-1} used to represent a unit delay and so a $p \times q$ polynomial matrix $\underline{\mathbf{A}}(z)$ can be expressed as

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

where $\tau \in \mathbb{Z}$, $t_1 \leq t_2$ and $\mathbf{A}(\tau) \in \mathbb{R}^{p\times q}$ is the matrix containing the coefficients of $z^{-\tau}$. It is assumed that all polynomial matrices in this paper have real coefficients for simplicity. The polynomial coefficient in the $(j,k)^{th}$ polynomial element of $\underline{\mathbf{A}}(z)$ corresponding to a delay of $z^{-\tau}$ will be denoted as $a_{jk}(\tau)$ and so the $(j,k)^{th}$ polynomial element of the matrix can be expressed as

$$\underline{a}_{jk}(z) = \sum_{\tau=t_1}^{t_2} a_{jk}(\tau)z^{-\tau}. \quad (2)$$

This represents the impulse response of the propagation channel

from the k^{th} transmitter to the j^{th} sensor. The order of the polynomial matrix in equation (1) is by definition the quantity $(t_2 - t_1)$. The underline notation, for example in $\underline{\mathbf{A}}(z)$, is used to denote a polynomial, in this case, a matrix. This notation is also used to represent a polynomial vector or scalar to avoid confusion with the notation used for the z-transform of a variable. The set of polynomial matrices with real coefficients is denoted by $\underline{\mathbb{R}}^{p\times q}$, where p and q define the number of rows and columns of the matrix respectively.

The paraconjugate of the polynomial matrix $\underline{\mathbf{A}}(z)$ is defined to be $\widetilde{\underline{\mathbf{A}}}(z) = \underline{\mathbf{A}}^T(1/z)$ where $(\cdot)^T$ denotes matrix transposition. The tilde notation $\widetilde{(\cdot)}$ will be used throughout this paper to denote paraconjugation. A polynomial matrix $\underline{\mathbf{A}}(z) \in \underline{\mathbb{R}}^{p\times p}$ is said to be paraunitary if the following statement is true $\underline{\mathbf{A}}(z)\widetilde{\underline{\mathbf{A}}}(z) = \widetilde{\underline{\mathbf{A}}}(z)\underline{\mathbf{A}}(z) = \mathbf{I}_p$ where $\mathbf{I}_p$ denotes the $p \times p$ identity matrix. Finally, the Frobenius norm, or F-norm, of the polynomial matrix $\underline{\mathbf{A}}(z) \in \underline{\mathbb{R}}^{p\times q}$ is defined as $\|\underline{\mathbf{A}}(z)\|_F = \sqrt{\sum_{\tau=t_1}^{t_2}\sum_{i=1}^{p}\sum_{j=1}^{q}(a_{ij}(\tau))^2}$.

2. AN 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. For simplicity, a 2×2 EPGR is firstly introduced. An EPGR takes the form of a Givens rotation preceded by an elementary time shift matrix as follows

$$\underline{\mathbf{G}}^{(t,\theta)}(z) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & z^t \end{bmatrix} \quad (3)$$

$$= \begin{bmatrix} \cos(\theta) & \sin(\theta)z^t \\ -\sin(\theta) & \cos(\theta)z^t \end{bmatrix}. \quad (4)$$

The aim of this matrix, when applied to a polynomial vector $\underline{\mathbf{a}}(z) = [\underline{a}_1(z), \underline{a}_2(z)]^T \in \underline{\mathbb{R}}^{2\times 1}$ as demonstrated by

$$\begin{bmatrix} \cos(\theta) & \sin(\theta)z^t \\ -\sin(\theta) & \cos(\theta)z^t \end{bmatrix}\begin{bmatrix} \underline{a}_1(z) \\ \underline{a}_2(z) \end{bmatrix} = \begin{bmatrix} \underline{a}'_1(z) \\ \underline{a}'_2(z) \end{bmatrix} \quad (5)$$

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

$$\tan(\theta) = \frac{a_2(\tau)}{a_1(0)}, \quad (6)$$

thus resulting in $a'_2(0) = 0$. Furthermore, following the application of the EPGR, the coefficient $a'_1(0)$ has increased in magnitude squared such that $(a'_1(0))^2 = (a_1(0))^2 + (a_2(\tau))^2$. Note that the order of the vector $\underline{\mathbf{a}}(z)$ will increase by $|t|$ under the transformation. This is due to the elementary delay matrix incorporated in the EPGR, which will apply a t-fold delay upon $\underline{a}_2(z)$. The order of the vector must therefore increase to accommodate all of the shifted coefficients. The polynomial matrix $\underline{\mathbf{G}}^{(t,\theta)}(z)$ is paraunitary and as a consequence, the transformation demonstrated in equation (5) satisfies $\left\|\underline{\mathbf{G}}^{(t,\theta)}(z)\underline{\mathbf{a}}(z)\right\|_F = \|\underline{\mathbf{a}}(z)\|_F$.

An EPGR can easily be extended so that it can be applied to a polynomial matrix $\underline{\mathbf{A}}(z) \in \underline{\mathbb{R}}^{p\times q}$ to drive a single coefficient of one of the polynomial elements to zero. For example, suppose we wish to drive the polynomial coefficient $a_{jk}(t)$ to zero. The appropriate EPGR required to do this, takes the form of 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 four elements of the EPGR $\underline{\mathbf{G}}^{(t,\theta)}(z)$ demonstrated in equation (4). This $p \times p$ matrix will be defined as $\underline{\mathbf{G}}^{(j,k,t,\theta)}(z)$ where the superscripts j and k have been added to denote the position of the coefficient, which will be driven to zero under the application of the EPGR. The coefficients required for calculating the rotation angle θ in equation (6) then correspond to $a_2(\tau) = a_{jk}(t)$ and $a_1(0) = a_{kk}(0)$. Note that the scalar Givens rotation in (3) can easily be extended to be applicable to polynomial matrices with complex coefficients as demonstrated in [4, 5]. These matrices now form the basis of the proposed algorithm for calculating the PQRD.

3. THE QR DECOMPOSITION OF A POLYNOMIAL MATRIX

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

$$\underline{\mathbf{Q}}(z)\underline{\mathbf{A}}(z) = \underline{\mathbf{R}}(z) \quad (7)$$

where $\underline{\mathbf{R}}(z) \in \underline{\mathbb{R}}^{p\times q}$ is an upper triangular polynomial matrix. The polynomial matrix $\underline{\mathbf{Q}}(z)$ is computed as a series of EPGR matrices, each one designed to drive a single coefficient of one of the polynomial elements situated beneath the diagonal to be sufficiently small.

3.1. THE PQRD BY COLUMNS ALGORITHM

The algorithm operates as a series of ordered steps where at each step, all coefficients relating to all polynomial elements beneath the diagonal in one column of the matrix, are driven sufficiently small by applying a series of EPGR matrices interspersed with inverse delay matrices. The step process will be referred to as a column-step to avoid confusion with the terminology used to describe the PQRD-BS algorithm [4, 5]. The algorithm begins the first column-step with the first column of the matrix. The process to implement this first step is now explained.

3.1.1. A Single Step of the Algorithm

The first column-step operates as an iterative process, where at each iteration an EPGR is applied to $\underline{\mathbf{A}}(z)$ to zero the coefficient with maximum magnitude within any of the polynomial elements situated beneath the diagonal in the first column of the matrix. The first iteration begins by locating the coefficient with maximum magnitude from the elements $\underline{a}_{21}(z), \ldots, \underline{a}_{p1}(z)$. Suppose this coefficient is found to be $a_{j1}(t)$, i.e. the coefficient of z^{-t} in the polynomial element $\underline{a}_{j1}(z)$. This coefficient will be referred to as the dominant coefficient and if it is not unique then any of the dominant coefficients may be chosen.

Firstly, the rotation angle θ and the EPGR matrix $\underline{\mathbf{G}}^{(j,1,t,\theta)}(z)$ are calculated according to Section 2, such that when this matrix is applied to the polynomial matrix $\underline{\mathbf{A}}(z)$ as follows

$$\underline{\mathbf{A}}'(z) = \underline{\mathbf{G}}^{(j,1,t,\theta)}(z)\underline{\mathbf{A}}(z), \quad (8)$$

the dominant coefficient $a_{j1}(t)$ will have been driven to zero. Following the transformation $a'_{j1}(0) = 0$ and also $(a'_{11}(0))^2 = (a_{11}(0))^2 + (a_{j1}(t))^2$.

Next a paraunitary inverse delay matrix $\underline{\mathbf{B}}^{(j,t)}(z)$ is applied to $\underline{\mathbf{A}}'(z)$ as demonstrated by

$$\underline{\mathbf{A}}''(z) = \underline{\mathbf{B}}^{(j,t)}(z)\underline{\mathbf{A}}'(z). \tag{9}$$

The matrix $\underline{\mathbf{B}}^{(j,t)}(z) \in \underline{\mathbb{R}}^{p\times p}$ takes the form of an identity matrix with the exception of the j^{th} diagonal element which is z^{-t}. The application of this matrix will apply a t-fold delay to all elements in the j^{th} row of $\underline{\mathbf{A}}'(z)$ such that $a''_{jk}(\tau+t) = a'_{jk}(\tau)$ for $k = 1, \ldots, q$ and $\forall\tau \in \mathbb{Z}$. Note that the order of the matrix $\underline{\mathbf{A}}'(z)$ must increase to accommodate the delayed coefficients. In particular, the purpose of this matrix is to ensure that the coefficients of z^0 in $\underline{\mathbf{A}}(z)$ are returned to their original positions following the application of the EPGR. As a result, this will stop the unstable behaviour that has been observed in the PQRD-BS algorithm as it converges. The advantage of using this additional inverse delay matrix is demonstrated in Section 4 by means of a simple example.

This completes the first iteration of the first column-step of the algorithm. Over this iteration the complete transformation performed to zero the polynomial coefficient $a_{j1}(t)$ is of the form

$$\underline{\mathbf{A}}''(z) = \underline{\mathbf{B}}^{(j,t)}(z)\underline{\mathbf{G}}^{(j,1,t,\theta)}(z)\underline{\mathbf{A}}(z). \tag{10}$$

This iterative process is repeated replacing $\underline{\mathbf{A}}(z)$ with $\underline{\mathbf{A}}''(z)$ until all coefficients associated with polynomial elements beneath the diagonal in the first column of the matrix are sufficiently small.

In practice it is often not feasible to zero all coefficients of the polynomial elements beneath the diagonal in the column. Instead the coefficients are driven to zero until the magnitude of all coefficients associated with these elements are sufficiently small and the following stopping condition is satisfied

$$|a_{j1}(t)| < \epsilon \tag{11}$$

for $j = 2, \ldots, p$ and $\forall t \in \mathbb{Z}$ where $\epsilon > 0$ is a pre-specified small value. Once this stopping condition has been satisfied, the overall transformation performed in the first column-step of the algorithm is of the form

$$\underline{\mathbf{A}}^1(z) = \underline{\mathbf{Q}}^1(z)\underline{\mathbf{A}}(z), \tag{12}$$

where $\underline{\mathbf{Q}}^1(z) \in \underline{\mathbb{R}}^{p\times p}$ is formed of a series of EPGRs interspersed with inverse delay matrices and is therefore paraunitary by construction. Furthermore, following this column-step the first column of the matrix $\underline{\mathbf{A}}(z)$ will satisfy

$$\underline{\mathbf{Q}}^1(z)\begin{bmatrix} \underline{a}_{11}(z) \\ \underline{a}_{21}(z) \\ \vdots \\ \underline{a}_{p1}(z) \end{bmatrix} \cong \begin{bmatrix} \underline{a}^1_{11}(z) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{13}$$

where $\underline{a}^1_{11}(z)$ is the first diagonal element of $\underline{\mathbf{A}}^1(z)$.

Convergence of a single column-step can now be easily deduced. With every application of an EPGR, to zero the dominant coefficient say $a_{k1}(t)$, the quantity $(a_{11}(0))^2$ will increase by the magnitude squared of the largest coefficient beneath the diagonal in the first column. Furthermore, this quantity is bounded above by the squared F-norm of the first column of $\underline{\mathbf{A}}(z)$, i.e. the F-norm of the vector $\left[\underline{a}_{11}(z), \ldots, \underline{a}_{p1}(z)\right]^T$, which remains constant throughout all iterations of the algorithm. As $(a_{11}(0))^2$ is monotonically increasing and bounded above, over a series of EPGRs the condition set by equation (11) is guaranteed and the column-step converges in this respect.

3.1.2. The Algorithm

To begin a subsequent column-step, the iterative process outlined in Section 3.1.1 is repeated, moving to the next column of the matrix, i.e. the column positioned to the right of the column from the previous column-step. Following i column-steps, 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 formed of a series of EPGRs interspersed with inverse delay matrices and is therefore paraunitary by construction. Note that over each iteration of each column-step of the algorithm, the order of the matrices $\underline{\mathbf{A}}^i(z)$ and $\underline{\mathbf{Q}}^i(z)$ will increase due to the application of both the EPGR and inverse delay matrices.

Once all columns of the matrix have been visited, one sweep of the algorithm has been completed. Although the dominant coefficient at each iteration is driven to zero, the algorithm only ensures that all coefficients of the series of elements beneath the diagonal in one column are suitably small before starting the next column-step. Therefore, through future column-steps of the algorithm, these small coefficients can be rotated with other suitably small coefficients, allowing them to increase in magnitude. As a result, multiple sweeps of the algorithm may be required to ensure that the magnitude of all coefficients relating to elements beneath the diagonal of the matrix are less than ϵ. Despite this, the algorithm is guaranteed to converge and generally only a couple of sweeps are required. Note that this is not a problem when calculating the QRD of a scalar matrix, where all elements beneath the diagonal are driven to zero. Convergence of the PQRD-BC algorithm can be easily deduced from the proof of convergence for the SBR2 algorithm [1] and is outlined in [6].

3.1.3. Truncation of the Polynomial Matrices

The orders of the polynomial matrices $\underline{\mathbf{A}}^i(z)$ and $\underline{\mathbf{Q}}^i(z)$ from equation (14) will increase with the application of every elementary delay matrix at each iteration, of each column-step of the algorithm. Often after a series of iterations the orders can become very large with many of the coefficients positioned at outer lags of the matrix equal to a small proportion of the F-norm of the matrix. This is undesirable for the potential application of the algorithm to MIMO equalisation, where the computational complexity of the equaliser will be directly proportional to the order of the upper triangular matrix $\underline{\mathbf{R}}(z)$. Furthermore, large orders will also slow down the iterative computational procedure of the algorithm. Therefore, throughout the algorithm both polynomial matrices are truncated [4, 5]. For a polynomial matrix $\underline{\mathbf{A}}(z) \in \underline{\mathbb{R}}^{p\times q}$, with coefficient matrices $\mathbf{A}(t) \in \mathbb{R}^{p\times q}$ for $t = t_1, \ldots, t_2$, a suitable truncation method 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. To ensure that an accurate decomposition has been performed for a particular choice of μ, the relative error of the decomposition can be calculated as

$$E_{rel} = \left\| \underline{\mathbf{A}}(z) - \underline{\widetilde{\mathbf{Q}}}(z)\underline{\mathbf{R}}(z) \right\|_F / \left\| \underline{\mathbf{A}}(z) \right\|_F . \quad (17)$$

Using this truncation method does not affect the proof of convergence for the algorithm [6].

4. EXAMPLE OF THE PQRD

A polynomial matrix $\underline{\mathbf{A}}(z) \in \mathbb{R}^{4\times 3}$ was generated. Each of the polynomial elements was chosen to be a fourth order FIR filter with coefficients drawn from a Gaussian distribution with mean zero and unit variance. The QRD of $\underline{\mathbf{A}}(z)$ was obtained using the PQRD-BC algorithm, where the truncation parameter and the stopping criterion were set as $\mu = 10^{-6}$ and $\epsilon = 10^{-2}$ respectively. Only two sweeps of the algorithm were required to ensure the stopping condition demonstrated by (11) was satisfied, requiring a total of 233 iterations over six column-steps. The upper triangular matrix $\underline{\mathbf{R}}(z)$ obtained from the algorithm can be seen in Figure 1, where a stem plot has been used to demonstrate the series of coefficients for each of the polynomial elements. The position of the stem plot in the figure corresponds to the position of the polynomial element, which it represents within the matrix. To ensure that an accurate decomposition was performed despite truncating the polynomial matrices, the relative error was calculated according to equation (17). This measure was found to be 0.0057 demonstrating that a good approximation has been achieved. Note that if this measure if too large, then the value of μ can of course be decreased.

The existing PQRD-BS algorithm, as described in [4], was then applied to $\underline{\mathbf{A}}(z)$ using the same values of μ and ϵ. This algorithm required a further 50 iterations to converge than the PQRD-BC algorithm, but also obtained a larger relative error of 0.0079. Furthermore, from Figure 2 the F-norm, ν, of all polynomial elements beneath the diagonal of the matrix $\underline{\mathbf{A}}(z)$ over the series of iterations of the algorithm can be seen to suddenly increase in two places. By inspection of the same figure, the measure ν obtained using the proposed PQRD-BC algorithm can be seen to not exhibit this erratic behaviour. This is due to the inverse time shift matrix applied at each iteration of the algorithm. If this inverse time shift is also applied at each iteration of the PQRD-BS algorithm, then the erratic behaviour is no longer observed, but again the convergence is slower. This measure is also demonstrated in Figure 2.

5. CONCLUSIONS

We have presented a novel algorithm for calculating the QRD of a polynomial matrix. The method has been compared to an existing algorithm and has shown improved performance in terms of the number of iterations for the algorithm to converge. Moreover, by addition of the application of an inverse time shift matrix, which stops off-diagonal coefficients within the matrix suddenly increasing, erratic behaviour in the convergence of the earlier algorithm is removed. Future works aim at exploring the potential application of the decomposition algorithm to broadband MIMO channel equalisation and a comparison of this method with a MIMO orthogonal frequency division multiplexing QRD approach.

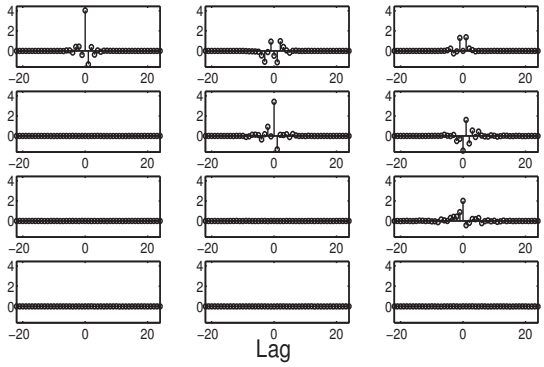

Fig. 1. The polynomial elements of the upper triangular polynomial matrix $\underline{\mathbf{R}}(z)$, obtained when the PQRD-BC algorithm was applied to the polynomial matrix $\underline{\mathbf{A}}(z)$.

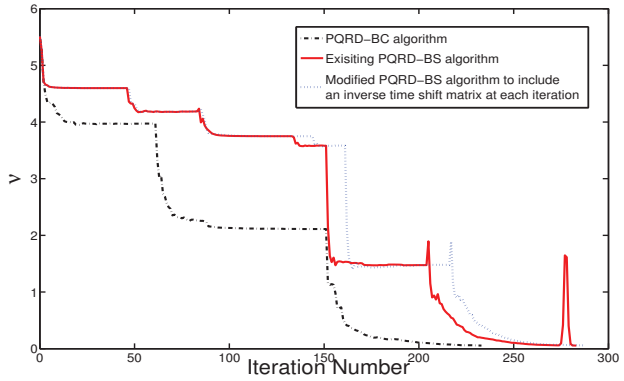

Fig. 2. The F-norm of all of the polynomial elements beneath the diagonal, ν, over the series of iterations of i) the proposed PQRD-BC algorithm, ii) the existing PQRD-BS algorithm and iii) the PQRD-BS algorithm with an inverse delay matrix at each iteration.

6. 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*, vol. 55, no. 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.A. Foster, J.G. McWhirter and J. Chambers, "A Polynomial Matrix QR Decomposition with Application to MIMO Channel Equalisation," *Proc. 41^{st} Asilomar Conference on Signals, Systems and Computers*, 2007.

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

[6] J.A. Foster, *Algorithms and Techniques for Polynomial Matrix Decompositions*, Ph.D. thesis, School of Engineering, Cardiff University, UK, 2008.