A NOVEL ALGORITHM FOR CALCULATING THE SINGULAR VALUE DECOMPOSITION OF A POLYNOMIAL MATRIX

Joanne Foster[1]*, *John McWhirter*[2] *and Jonathon Chambers*[1]

[1]Advanced Signal Processing Group,
Dept. of Electronic Electrical Engineering,
Loughborough University, LE11 3TU, UK.
*Email:J.A.Foster@lboro.ac.uk

[2]Centre of Digital Signal Processing,
School of Engineering, Cardiff University,
CF24 3AA, UK.

ABSTRACT

A novel algorithm for calculating the singular value decomposition (SVD) of a polynomial matrix is proposed. The algorithm operates by iteratively calculating the QR decomposition (QRD) of the matrix to transform it to a diagonal polynomial matrix. Alternatively, this decomposition can be calculated using the polynomial matrix eigenvalue decomposition (EVD) routine known as the second-order sequential best rotation (SBR2) algorithm. However, this method does not operate directly on the polynomial matrix. Instead, it formulates the left and right singular vectors in turn, by calculating the EVD of two para-Hermitian matrices formed from the polynomial matrix. The advantage of the algorithm proposed in this paper, is that it does operate directly upon the polynomial matrix and can therefore allow significantly more control over the level of decomposition performed. The two approaches are compared by means of computer simulations which demonstrates that the method based on the polynomial matrix QRD algorithm is numerically superior.

Index Terms— Convolutive mixing, paraunitary matrix, polynomial matrix singular value decomposition, MIMO channel equalisation.

1. INTRODUCTION

A polynomial matrix is simply a matrix with polynomial elements. Matrices of this form have previously been applied in the field of control, but in recent years they have also been used extensively in the areas of digital signal processing and communications [1, 2]. In this situation, a polynomial matrix can be 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. The received signals, obtained at the sensors, will then consist of weighted and delayed versions of the transmitted signals. The channel matrix required to express this mixing process takes the form of a polynomial matrix, where each element is a finite impulse response (FIR) filter. For example, 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 $\tau \in \mathbb{Z}$ and $t_1 \leq t_2$, where the values of the parameters t_1 and t_2 in the above equation are not necessarily positive. $\mathbf{A}(\tau) \in \mathbb{R}^{p \times q}$ is the matrix containing the coefficients of $z^{-\tau}$ from each of the polynomial elements. Each element of the mixing matrix in equation (1) will be an FIR filter and so the indeterminate variable of each of the polynomial elements of the matrix is z^{-1}, as this is commonly used to represent a unit delay. For example, the $(j,k)^{th}$ polynomial element of $\underline{\mathbf{A}}(z)$ can be expressed as

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

where $a_{jk}(\tau)$ will define the polynomial coefficient of $\underline{a}_{jk}(z)$ corresponding to a delay of $z^{-\tau}$. This element represents the impulse response of the propagation channel from the k^{th} transmitter to the j^{th} sensor. For clarity of development, it is assumed that all polynomial matrices in this paper have real coefficients, however, extension to complex coefficients is straight forward.

If instead the received signals are instantaneously mixed then a matrix with scalar elements is sufficient to describe the mixing process. In this situation, conventional matrix decompositions, such as the SVD, have proven to be extremely useful for manipulating the channel matrix into a simpler form [3]. For example, the SVD can be used to transform the channel matrix into a diagonal matrix, which can then be exploited to easily retrieve estimates of the transmitted signals [4]. This decomposition can therefore be used in a narrowband MIMO communication system, to split a MIMO channel into a number of independent spatial subchannels [4]. Extensions of these conventional scalar matrix decomposition techniques to polynomial matrices would be extremely useful in the convolutive mixing scenario where polynomial matrices now arise.

The SBR2 algorithm has previously been applied to this problem in [5, 6], where it has been used to formulate the SVD of a polynomial matrix. In the work reported here an alternative method for calculating this decomposition is proposed, which operates by iteratively applying an algorithm proposed in [7] for calculating the QRD of a polynomial matrix. This algorithm can also be applied to broadband MIMO channel equalisation problems as discussed in [8], but this is beyond the scope of this paper.

The remainder of the paper is organised as follows. Firstly, the terminology and notation are defined. An algorithm for calculating the QRD of a polynomial matrix (PQRD) is then detailed, as this forms the basis of the proposed polynomial matrix SVD (PSVD) algorithm. The PSVD algorithm is then introduced. Finally, the capability and improved performance of the PSVD by PQRD algorithm over the existing method, which operates using the SBR2 algorithm, is confirmed by simulation for a particular polynomial matrix.

1.1. Notation and Definitions

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

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

where $\mathbf{I}_p$ denotes the $p \times p$ identity matrix. Finally, the Frobenius norm, or F-norm, of the polynomial matrix $\underline{\mathbf{A}}(z)$ is defined as

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

2. THE QR DECOMPOSITION OF A POLYNOMIAL MATRIX

The PQRD By Columns (PQRD-BC) algorithm is a technique for factorising a polynomial matrix into an upper triangular and a paraunitary polynomial matrix [7]. 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

$$\underline{\mathbf{Q}}(z)\underline{\mathbf{A}}(z) = \underline{\mathbf{R}}(z) \tag{5}$$

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 elementary delay and rotation matrices. These matrices can be used to formulate a polynomial Givens rotation, which are used within the PQRD-BC algorithm and are therefore firstly introduced.

2.1. An Elementary Polynomial Givens Rotations

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 one of the polynomial elements. For example, suppose for a polynomial matrix $\underline{\mathbf{A}}(z) \in \underline{\mathbb{R}}^{p\times q}$ we wish to drive the coefficient of $z^{-\tau}$ in the polynomial element $\underline{a}_{jk}(z)$, i.e. the coefficient $a_{jk}(t)$, to zero. The EPGR required to do this is referred to as $\underline{\mathbf{G}}^{(j,k,t,\theta)}(z)$ and is formed of a Givens rotation preceded by an elementary time shift matrix. For this example, it will take 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 four elements are denoted as $\underline{g}_{kk}(z)$, $\underline{g}_{jk}(z)$, $\underline{g}_{kj}(z)$ and $\underline{g}_{jj}(z)$ and are calculated such that

$$\begin{bmatrix} \underline{g}_{kk}(z) & \underline{g}_{kj}(z) \\ \underline{g}_{jk}(z) & \underline{g}_{jj}(z) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & z^t \end{bmatrix} \tag{6}$$

$$= \begin{bmatrix} \cos(\theta) & \sin(\theta)z^t \\ -\sin(\theta) & \cos(\theta)z^t \end{bmatrix}, \tag{7}$$

where the values of the rotation angle θ is chosen such that $a_{jk}(t)$ is driven to zero under application of $\underline{\mathbf{G}}^{(j,k,t,\theta)}(z)$. This is satisfied when

$$\tan(\theta) = \frac{a_{jk}(\tau)}{a_{kk}(0)} \tag{8}$$

provided $a_{kk}(0) \neq 0$. Note that if $a_{kk}(0) = 0$, then the rotation angle is set as $\theta = \pi/2$. Following the application of the EPGR upon $\underline{\mathbf{A}}(z)$ as demonstrated by

$$\underline{\mathbf{G}}^{(j,k,t,\theta)}(z)\underline{\mathbf{A}}(z) = \underline{\mathbf{A}}^{'}(z), \tag{9}$$

will result in $a'_{jk}(0) = 0$. Furthermore, following the application of the EPGR, the coefficient $a'_{kk}(0)$ has increased in magnitude squared such that $(a'_{kk}(0))^2 = (a_{kk}(0))^2 + (a_{jk}(\tau))^2$. Note that the order of the matrix $\underline{\mathbf{A}}(z)$ will increase by $|t|$ under this transformation. This is due to the elementary delay matrix incorporated in the EPGR, which will apply a t-fold delay upon all elements in the j^{th} row of the matrix. The order must therefore increase to accommodate all of the shifted coefficients. Note also that the polynomial matrix $\underline{\mathbf{G}}^{(j,k,t,\theta)}(z)$ is paraunitary and as a consequence, the transformation demonstrated in equation (9) satisfies $\left\|\underline{\mathbf{A}}'(z)\right\|_F = \|\underline{\mathbf{A}}(z)\|_F$. These matrices now form the basis of the proposed algorithm for calculating the PQRD.

2.2. The PQRD by Columns Algorithm

The PQRD-BC algorithm operates as a series of ordered steps, where at each step (referred to as a column-step) all coefficients associated with the polynomial elements beneath the diagonal of one column of the matrix are driven sufficiently small according to a suitable stopping condition.

The algorithm begins the first step with the first column of the matrix. The first iteration of the first column-step begins by locating the coefficient with maximum magnitude from any of the elements situated beneath the diagonal of the first column, i.e. the coefficient with maximum magnitude from the series of elements $\underline{a}_{21}(z),\ldots,\underline{a}_{p1}(z)$. Suppose this coefficient is found to be $a_{j1}(t)$ - 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.

Once this element has been located, the rotation angle θ and the EPGR matrix $\underline{\mathbf{G}}^{(j,1,t,\theta)}(z)$ are calculated according to Section 2.1, such that when this matrix is applied to the polynomial matrix $\underline{\mathbf{A}}(z)$ as follows

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

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

$$\underline{\mathbf{A}}''(z) = \underline{\mathbf{B}}^{(j,t)}(z)\underline{\mathbf{A}}'(z). \tag{11}$$

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 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 erratic behaviour that has been observed in an earlier algorithm proposed for calculating the PQRD [7].

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

$$\underline{\mathbf{A}}''(z) = \underline{\mathbf{B}}^{(j,t)}(z)\underline{\mathbf{G}}^{(j,1,t,\theta)}(z)\underline{\mathbf{A}}(z). \tag{12}$$

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

In practice it is often not feasible to zero all coefficients of the polynomial elements beneath the diagonal of 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

$$\left|a_{j1}(t)\right| < \varepsilon \tag{13}$$

for $j = 2,\ldots,p$ and $\forall t \in \mathbb{Z}$ where $\varepsilon > 0$ 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

$$\underline{\mathbf{A}}^1(z) = \underline{\mathbf{Q}}^1(z)\underline{\mathbf{A}}(z), \tag{14}$$

where $\underline{\mathbf{Q}}^1(z) \in \mathbb{R}^{p\times p}$ is formed of a series of EPGRs interspersed with inverse delay matrices and is therefore paraunitary by construction.

To begin a subsequent column-step, the iterative process outlined above is repeated moving to the next column of the matrix. The columns are visited according to the ordering $k = 1,2\ldots,\min\{(p-1),q\}$, which is important for convergence of the algorithm. Following i column-steps of the algorithm, the transformation is of the form

$$\underline{\mathbf{A}}^i(z) = \underline{\mathbf{Q}}^i(z)\underline{\mathbf{A}}(z) \tag{15}$$

where $\underline{\mathbf{Q}}^i(z)$ is formed of a series of EPGRs interspersed with inverse delay matrices and is therefore paraunitary by construction.

Once all columns of the matrix have been visited, this completes one sweep of the algorithm. The PQRD-BC algorithm, although driving the dominant coefficient at each iteration to zero, only ensures that all coefficients of the series of elements beneath the diagonal of one column are suitably small before moving to the next column in the ordering. Therefore, through future column-steps 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. Despite this, the algorithm is guaranteed to converge and in practice generally only a couple of sweeps are ever required.

2.2.1. Polynomial Matrix Truncation Method

The orders of the polynomial matrices $\underline{\mathbf{A}}^i(z)$ and $\underline{\mathbf{Q}}^i(z)$ from equation (15) 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 of these matrices 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. For this reason, both these polynomial matrices can be truncated at each iteration of the algorithm according to the following criteria.

For a polynomial matrix $\underline{\mathbf{A}}(z) \in \mathbb{R}^{p\times q}$, 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

$$\frac{\sum_{\tau=t_1}^{T_1}\sum_{l=1}^{p}\sum_{m=1}^{q}(a_{lm}(\tau))^2}{\|\underline{\mathbf{A}}(z)\|_F^2} \leq \frac{\mu}{2} \tag{16}$$

and

$$\frac{\sum_{\tau=T_2}^{t_2}\sum_{l=1}^{p}\sum_{m=1}^{q}(a_{lm}(\tau))^2}{\|\underline{\mathbf{A}}(z)\|_F^2} \leq \frac{\mu}{2} \tag{17}$$

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

$$E_{rel} = \left\|\underline{\mathbf{A}}(z) - \underline{\widetilde{\mathbf{Q}}}(z)\underline{\mathbf{R}}(z)\right\|_F / \|\underline{\mathbf{A}}(z)\|_F . \tag{18}$$

Note that this truncation method is not essential to the algorithm. However, using it can significantly reduce the computational time taken to implement the algorithm. Furthermore, large orders are generally undesirable when applying the algorithm to MIMO equalisation problems, where the computational complexity of the equaliser will be directly proportional to the order of the upper triangular matrix $\underline{\mathbf{R}}(z)$. In this application, the truncation method can be used to prevent this problem. Moreover, if this truncation method is used, it will not affect convergence of the algorithm. This PQRD algorithm, can now be used to formulate an algorithm for calculating the SVD of a polynomial matrix.

3. THE SINGULAR VALUE DECOMPOSITION OF A POLYNOMIAL MATRIX

Let $\underline{\mathbf{A}}(z) \in \mathbb{R}^{p\times q}$, the objective of the PSVD algorithm is to calculate the paraunitary matrices $\underline{\mathbf{U}}(z) \in \mathbb{R}^{p\times p}$ and $\underline{\mathbf{V}}(z) \in \mathbb{R}^{q\times q}$ such that

$$\underline{\mathbf{U}}(z)\underline{\mathbf{A}}(z)\underline{\widetilde{\mathbf{V}}}(z) = \underline{\mathbf{S}}(z) \tag{19}$$

where $\underline{\mathbf{S}}(z) \in \mathbb{R}^{p\times q}$ is a diagonal polynomial matrix. A method of formulating the PSVD has already been proposed in [1]. This method operates by using the second-order sequential best rotation (SBR2) algorithm to calculate the EVD of the para-Hermitian polynomial matrices $\underline{\mathbf{A}}(z)\underline{\widetilde{\mathbf{A}}}(z)$ and $\underline{\widetilde{\mathbf{A}}}(z)\underline{\mathbf{A}}(z)$ in turn, to calculate the left and right singular vectors of $\underline{\mathbf{A}}(z)$ [5, 6]. Note that the SBR2 algorithm only generates an approximate decomposition, resulting in an approximately diagonal polynomial matrix $\underline{\mathbf{S}}(z)$. Furthermore, as this method does not operate directly on the polynomial matrix $\underline{\mathbf{A}}(z)$, it is not possible to specify the level of decomposition in advance. The PSVD by PQRD algorithm proposed in this paper does operate directly on the polynomial matrix. This algorithm is now discussed.

3.1. The PSVD by PQRD Algorithm

The algorithm begins the first iteration by calculating the PQRD of the matrix $\underline{\mathbf{A}}(z) \in \mathbb{R}^{p\times q}$ such that

$$\underline{\mathbf{U}}_1(z)\underline{\mathbf{A}}(z) = \underline{\mathbf{R}}_1(z) \tag{20}$$

where $\underline{\mathbf{R}}_1(z) \in \mathbb{R}^{p\times q}$ is an approximately upper-triangular polynomial matrix and $\underline{\mathbf{U}}_1(z) \in \mathbb{R}^{p\times p}$ is paraunitary. Next the QRD of the polynomial matrix $\underline{\mathbf{A}}'(z) = \underline{\widetilde{\mathbf{R}}}_1(z) \in \mathbb{R}^{q\times p}$ is calculated such that

$$\underline{\mathbf{V}}_1(z)\underline{\mathbf{A}}'(z) = \underline{\mathbf{R}}_2(z) \tag{21}$$

where $\underline{\mathbf{R}}_2(z) \in \mathbb{R}^{q\times p}$ is an upper-triangular polynomial matrix and $\underline{\mathbf{V}}_1(z) \in \mathbb{R}^{q\times q}$ is a paraunitary polynomial matrix. The overall decomposition following the first iteration is therefore of the form

$$\underline{\mathbf{U}}_1(z)\underline{\mathbf{A}}(z)\underline{\widetilde{\mathbf{V}}}_1(z) = \underline{\mathbf{A}}_1(z) \tag{22}$$

where $\underline{\mathbf{A}}_1(z) = \underline{\widetilde{\mathbf{R}}}_2(z)$. This matrix is guaranteed to be approximately lower triangular. This iterative process is then repeated replacing $\underline{\mathbf{A}}(z)$ with $\underline{\widetilde{\mathbf{R}}}_2(z)$ until all off-diagonal coefficients of this matrix are deemed sufficiently small according to the stopping condition

$$\left|a_{jk}(t)\right| < \varepsilon \tag{23}$$

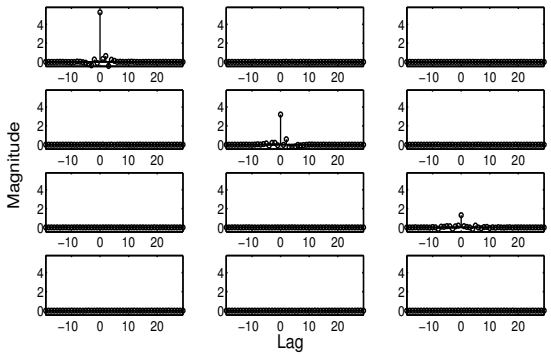

Fig. 1. The polynomial elements of the diagonal matrix $\underline{\mathbf{S}}(z)$, obtained when the PSVD by PQRD algorithm was applied to the polynomial matrix $\underline{\mathbf{A}}(z)$.

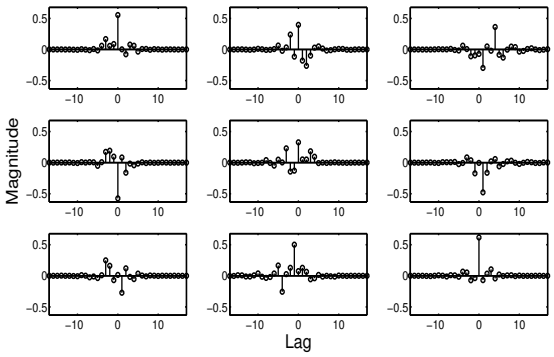

Fig. 3. The polynomial elements of the paraunitary matrix $\underline{\mathbf{V}}(z)$, obtained when the PSVD by PQRD algorithm was applied to the polynomial matrix $\underline{\mathbf{A}}(z)$.

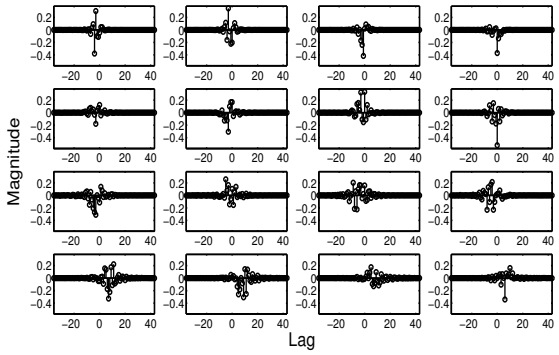

Fig. 2. The polynomial elements of the paraunitary matrix $\underline{\mathbf{U}}(z)$, obtained when the PSVD by PQRD algorithm was applied to the polynomial matrix $\underline{\mathbf{A}}(z)$.

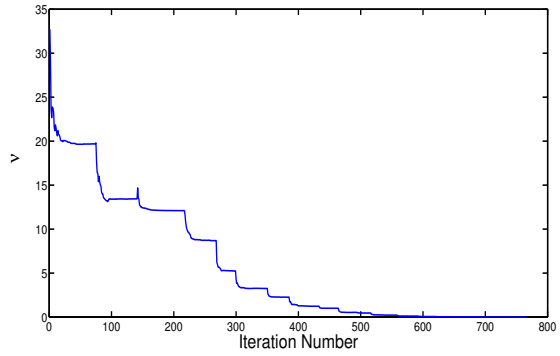

Fig. 4. The F-norm of the off-diagonal polynomial elements of $\underline{\mathbf{A}}(z)$ (ν) over the series of iterations of the proposed PSVD by PQRD algorithm.

$\forall t \in \mathbb{Z}$, $j = 1,\ldots,p$, $k = 1,\ldots,q$ such that $j \neq k$ and where $\varepsilon > 0$ is a prespecified small value.

Following i iterations of the algorithm, the decomposition calculated is of the form

$$\underline{\mathbf{U}}(z)\underline{\mathbf{A}}(z)\widetilde{\underline{\mathbf{V}}}(z) = \underline{\mathbf{A}}_i(z) \tag{24}$$

where $\underline{\mathbf{U}}(z) = \underline{\mathbf{U}}_i(z)\ldots\underline{\mathbf{U}}_1(z)$ and $\widetilde{\underline{\mathbf{V}}}(z) = \widetilde{\underline{\mathbf{V}}}_1(z)\ldots\widetilde{\underline{\mathbf{V}}}_i(z)$. Both of these matrices are clearly paraunitary by construction. Furthermore, the matrix $\underline{\mathbf{A}}_i(z)$ will converge to a diagonal matrix provided a sufficient number of iterations to satisfy the stopping condition demonstrated by equation (23).

4. A SIMPLE EXAMPLE

A polynomial matrix $\underline{\mathbf{A}}(z) \in \mathbb{R}^{4\times 3}$ was generated, where each of the elements was chosen to be a fourth order FIR filter with coefficients drawn from a Gaussian distribution with mean zero and unit variance. The PSVD of this matrix was obtained using the PSVD by PQRD algorithm, where the truncation parameter and the stopping criterion were set as $\mu = 10^{-6}$ and $\varepsilon = 10^{-2}$ respectively. Only 10 iterations of the algorithm were required to ensure the stopping condition given by equation (13) was satisfied and this required a total of 765 EPGRs over all iterations. The diagonal matrix $\underline{\mathbf{S}}(z)$ (of order 48) 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. The two paraunitary matrices $\underline{\mathbf{U}}(z)$ (of order 79) and $\underline{\mathbf{V}}(z)$ (of order 34) can be seen in Figures 2 and 3 respectively. To ensure that an accurate decomposition was performed despite truncating the polynomial matrices, the relative error was calculated as

$$E_{rel} = \left\| \underline{\mathbf{A}}(z) - \widetilde{\underline{\mathbf{U}}}(z)\underline{\mathbf{S}}(z)\underline{\mathbf{V}}(z) \right\|_F / \left\| \underline{\mathbf{A}}(z) \right\|_F . \tag{25}$$

This measure was found to be 0.0087 demonstrating that a good approximation has been achieved. Convergence of the algorithm is demonstrated in Figure 4, which demonstrates the F-norm, ν, of all off-diagonal polynomial elements of the matrix $\underline{\mathbf{A}}(z)$ over the series of EPGRs. Note that this example only aims to demonstrate how the PSVD by PQRD algorithm operates. For application purposes it may be desirable to further truncate the order of the diagonal matrix $\underline{\mathbf{S}}(z)$. From inspection of Figure 1, it can be seen that the outer coefficients of this matrix are very small, with the largest coefficients positioned in the central lags, i.e. for this matrix the larger coefficients appear to be positioned in lags $-5,\ldots,5$. If all coefficients outside of these lags are truncated from this diagonal matrix, then the relative error for this decomposition can again be calculated according to equation (25) and was now found to be 0.0433.

The SBR2 algorithm was then used to obtain the PSVD of $\underline{\mathbf{A}}(z)$. However, to generate approximately the same level of decomposi-

tion, i.e. in terms of the magnitude of the off-diagonal coefficients of $\underline{\mathbf{S}}(z)$, as that obtained when using the PSVD by PQRD algorithm is not a simple task. An empirical process of trial and error must be undertaken to find suitable values for both the stopping condition and truncation parameter of the algorithm that will generate a diagonal matrix, whose off-diagonal coefficients are approximately of the same magnitude as those obtained using the PSVD by PQRD algorithm. For this example, the appropriate values of the stopping condition and the truncation parameter were found to be 10^{-3} and 10^{-8} respectively. With these parameters the SBR2 algorithm required 6.70 seconds to formulate the decomposition, demonstrating that this method is considerably slower than the PSVD by PQRD approach, which required only 2.71 seconds[1]. Furthermore, the orders of the matrices $\underline{\mathbf{S}}(z)$ (order 178), $\underline{\mathbf{U}}(z)$ (order 182) and $\underline{\mathbf{V}}(z)$ (order 58) are all considerably larger than those obtained using the PQRD approach. This last point is particularly important 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 diagonal matrix $\underline{\mathbf{S}}(z)$.

5. CONCLUSIONS

We have presented a novel algorithm for calculating the SVD of a polynomial matrix. The method has been compared to an existing algorithm and has shown improved performance in terms of the computational time taken for the algorithm to converge. Moreover, the orders of the polynomial matrices generated by the algorithm are typically shorter, which is an advantage if the decomposition is to be applied within a MIMO communication system. However, the most significant advantage of the algorithm proposed in this paper, is that it operates directly on the polynomial matrix, which is something that the SBR2 algorithm does not do. As as result, the PSVD by PQRD method allows more control over the level of decomposition performed and is also likely to be more numerically robust. Future work aims at exploring the potential application of the PSVD by PQRD algorithm to broadband MIMO channel equalisation. In particular, this work aims to compare this approach to previous results obtained when using the SBR2 algorithm and those obtained using a MIMO orthogonal frequency division multiplexing system [5].

[1]Computations undertaken on a *Intel Centrino Duo* processor with 1GB of RAM.

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, pp. 2158–2169, May 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] A.J. Paulraj, R. Nabar and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge University Press, 2003.

[5] M. Davies, S. Lambotharan, J.A. Chambers and J.G. McWhirter, "Broadband MIMO Beamforming For Frequency Selective Channels Using the Sequential Best Rotation Algorithm," 67^{th} *Vehicular Technology Conference (VTC), Singapore*, Spring, 2008.

[6] C.H. Ta and S. Weiss, "A Design of Precoding and Equalisation for Broadband MIMO Systems," 41^{st} *Asilomar Conference on Signals, Systems and Computers, California*, 2007.

[7] J. A. Foster, J. G. McWhirter and J. A. Chambers, "A Novel Algorithm for Calculating the QR Decomposition of a Polynomial Matrix," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Taipei*, 2009.

[8] M. Davies, S. Lambotharan, J. Foster, J. Chambers and J. McWhirter, "Polynomial Matrix QR Decomposition and Iterative Decoding of Frequency Selective MIMO Channels," *Accepted to IEEE Wireless Communications & Networking Conference (WCNC), Budapest*, 2009.