Complexity and Search Space Reduction in Cyclic-by-Row PEVD Algorithms

Fraser K. Coutts*, Jamie Corr*, Keith Thompson*, Stephan Weiss*, Ian K. Proudler*,†, John G. McWhirter‡
* Department of Electronic & Electrical Engineering, University of Strathclyde, Glasgow, Scotland
*,† School of Electrical, Electronics & Systems Engineering, Loughborough Univ., Loughborough, UK
‡ School of Engineering, Cardiff University, Cardiff, Wales, UK
{fraser.coutts,jamie.corr,keith.thompson,stephan.weiss}@strath.ac.uk, i.k.proudler@lboro.ac.uk, mcwhirterjg@cardiff.ac.uk

Abstract—**In recent years, several algorithms for the iterative calculation of a polynomial matrix eigenvalue decomposition (PEVD) have been introduced. The PEVD is a generalisation of the ordinary EVD and uses paraunitary operations to diagonalise a parahermitian matrix. This paper addresses potential computational savings that can be applied to existing cyclic-by-row approaches for the PEVD. These savings are found during the search and rotation stages, and do not significantly impact on algorithm accuracy. We demonstrate that with the proposed techniques, computations can be significantly reduced. The benefits of this are important for a number of broadband multichannel problems.**

I. Introduction

Polynomial matrix representations can elegantly express broadband multichannel problems. The use of such formulations can be found in multichannel factorisation [1], broadband MIMO precoding and equalisation [2], polyphase analysis and synthesis matrices for filter banks [3], broadband angle of arrival estimation [4], broadband beamforming [5], optimal subband coding [6], and channel coding [7] to name but a few. These problems generally involve parahermitian polynomial matrices, which are identical to their parahermitian conjugate, i.e., $\boldsymbol{R}(z) = \tilde{\boldsymbol{R}}(z) = \boldsymbol{R}^{\mathrm{H}}(z^{-1})$ [3].

While the eigenvalue decomposition (EVD) presents an optimal tool for many narrowband problems involving covariance matrices, the broadband case necessitates a factorisation for parahermitian polynomial matrices. Therefore as an extension of the EVD, a polynomial matrix EVD (PEVD) has been defined in [8], [9]. The PEVD uses finite impulse response (FIR) paraunitary matrices [11] to approximately diagonalise a space-time covariance matrix. Note that the PEVD also produces spectrally majorised polynomial eigenvalues [10].

Algorithms to compute the PEVD include the original second order sequential best rotation (SBR2) algorithm [9], sequential matrix diagonalisation (SMD) [12] and various evolutions of the algorithm families [13]–[16]. All of these algorithms employ an iterative approach to approximately diagonalise the parahermitian matrix, stopping when some suitable threshold is reached. Both SBR2 and SMD are computationally costly to compute, therefore any cost savings that can be applied to these algorithms will be advantageous for applications.

The focus of algorithmic cost reduction in PEVD algorithms has typically been the trimming of polynomial matrix factors to curb growth in order [9], [17]–[20], which translates directly into a growth of memory storage requirements and computational complexity. Besides trimming, storage and cost reductions have also been accomplished by considering the symmetry of the parahermitian matrix [21]. Further, an SMD cyclic-by-row (SMDCbR) approach [13] has been introduced as a low-cost variant of SMD, which approximates the EVD step inside the SMD algorithm by a cyclic-by-row implementation of the Jacobi algorithm [22].

Here we describe a method to concatenate the Jacobi rotations in SMDCbR to reduce the computational cost of the algorithm, and introduce thresholding to the rotation process to eliminate the rotation of near-zero elements. In addition, we provide another source of complexity reduction by demonstrating that the search space of the algorithm can be reduced without significant accuracy loss.

Below, Sec. II will provide a brief overview of the SMDCbR method. The proposed approaches for complexity reduction when implementing this algorithm are outlined in Sec. III. Simulation results demonstrating the savings are presented in Sec. IV, with conclusions drawn in Sec. V.

II. Sequential Matrix Diagonalisation Cyclic-by-Row

This section reviews aspects of the iterative PEVD algorithm SMDCbR [13] in Sec. II-A, with an assessment of the main algorithmic cost in Sec. II-B.

A. Algorithm Overview

The SMDCbR algorithm approximates the PEVD using a series of elementary paraunitary operations to iteratively diagonalise a parahermitian matrix $\boldsymbol{R}(z)$. Note that $\boldsymbol{R}(z)$ is the z-transform of a set of coefficient matrices relating to different lags, $\mathbf{R}[\tau]$. Each elementary paraunitary operation consists of two steps: first a delay operation is used to move the column with the largest energy in its off-diagonal elements to the zero lag; then an approximate EVD diagonalises the zero lag matrix, transferring the shifted off-diagonal energy onto the diagonal.

The SMDCbR algorithm is initialised with an approximate diagonalisation of the lag-zero coefficient matrix $\mathbf{R}[0]$ by means of its modal matrix $\mathbf{Q}^{(0)}$ from $\boldsymbol{S}^{(0)}(z) = \mathbf{Q}^{(0)}\boldsymbol{R}(z)\mathbf{Q}^{(0)\mathrm{H}}$. Note that the unitary $\mathbf{Q}^{(0)}$ — obtained from a cyclic-by-row approximation to the EVD of the lag-zero slice $\mathbf{R}[0]$ — is applied to all coefficient matrices $\mathbf{R}[\tau]\ \forall\ \tau$.

In the ith step, $i = 1, 2, \ldots I$, the SMDCbR algorithm performs a paraunitary transform such that

$$\boldsymbol{S}^{(i)}(z) = \boldsymbol{U}^{(i)}(z)\boldsymbol{S}^{(i-1)}(z)\tilde{\boldsymbol{U}}^{(i)}(z)\ , \tag{1}$$

whereby

$$\boldsymbol{U}^{(i)}(z) = \mathbf{Q}^{(i)}\boldsymbol{\Lambda}^{(i)}(z)\ . \tag{2}$$

The product (2) comprises of a delay matrix

$$\boldsymbol{\Lambda}^{(i)}(z) = \mathrm{diag}\{\underbrace{1\ \ldots\ 1}_{k^{(i)}-1}\ z^{-\tau^{(i)}}\ \underbrace{1\ \ldots\ 1}_{M-k^{(i)}}\}\quad , \tag{3}$$

and a unitary matrix $\mathbf{Q}^{(i)}$, with the result that $\boldsymbol{U}^{(i)}(z)$ in (2) is paraunitary by construction.

It is convenient for subsequent discussion to define an intermediate variable $\boldsymbol{S}^{(i)\prime}(z)$ where

$$\boldsymbol{S}^{(i)\prime}(z) = \boldsymbol{\Lambda}^{(i)}(z)\boldsymbol{S}^{(i-1)}(z)\tilde{\boldsymbol{\Lambda}}^{(i)}(z)\quad , \tag{4}$$

followed by a rotation

$$\boldsymbol{S}^{(i)}(z) = \mathbf{Q}^{(i)}\boldsymbol{S}^{(i)\prime}(z)\mathbf{Q}^{(i)\mathrm{H}}\quad . \tag{5}$$

The parameters of $\boldsymbol{\Lambda}^{(i)}(z)$ and $\mathbf{Q}^{(i)}$ in the ith iteration are determined by the position of the dominant off-diagonal column in $\boldsymbol{S}^{(i-1)}(z)\ \bullet\!\!-\!\!\circ\ \mathbf{S}^{(i-1)}[\tau]$,

$$\{k^{(i)}, \tau^{(i)}\} = \arg\max_{k,\tau} \|\hat{\mathbf{s}}_k^{(i-1)}[\tau]\|_2\quad , \tag{6}$$

where

$$\|\hat{\mathbf{s}}_k^{(i-1)}[\tau]\|_2 = \Big(\sum_{m=1, m\neq k}^{M} |\mathbf{s}_{m,k}^{(i-1)}[\tau]|^2\Big)^{\frac{1}{2}} \tag{7}$$

and $\mathbf{s}_{m,k}^{(i-1)}[\tau]$ represents the element in the mth row and kth column of the coefficient matrix at lag τ, $\mathbf{S}^{(i-1)}[\tau]$.

Due to the parahermitian symmetry in $\mathbf{S}^{(i-1)}[\tau]$, the shifting process in (4) moves both the dominant off-diagonal row and column into the zero-lag coefficient matrix; therefore, the modified norm in (7) serves to measure half of the total energy moved into the zero-lag matrix $\mathbf{S}^{(i)\prime}[0]$. This energy is transferred onto the diagonal by the unitary matrix $\mathbf{Q}^{(i)}$ in (5) that approximately diagonalises $\mathbf{S}^{(i)\prime}[0]$ by means of an approximate EVD.

At the ith iteration, SMDCbR calculates an approximation to the EVD of the zero-lag slice using a sequence of $n = 1 \ldots P$ Jacobi rotations $\mathbf{Q}^{(i,n)} \in \mathbb{C}^{M\times M}$. Each rotation takes the form of an $M \times M$ identity matrix, but with the four elements at the intersections of rows and columns $\{m^{(i,n)}, k^{(i,n)}\}$ defined by

$$\begin{bmatrix} \cos\phi^{(i,n)} & e^{j\theta^{(i,n)}}\sin\phi^{(i,n)} \\ -e^{-j\theta^{(i,n)}}\sin\phi^{(i,n)} & \cos\phi^{(i,n)} \end{bmatrix}\ . \tag{8}$$

The rotation angles $\phi^{(i,n)}$ and $\theta^{(i,n)}$ in (8) are determined by the target element at row $m^{(i,n)}$ and column $k^{(i,n)}$ in the slice.

Each Jacobi rotation transfers the energy of an off-diagonal element onto the diagonal. The process used in SMDCbR can

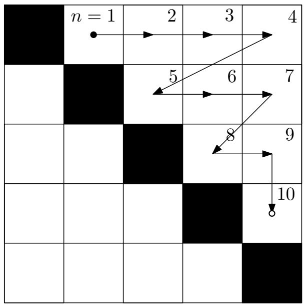

Fig. 1. Values of n used for cyclic-by-row sequence of Jacobi rotations in one Jacobi sweep for a 5×5 matrix with start ● and end point ○ .

be described as a cyclic-by-row implementation of the Jacobi algorithm [22]. The term CbR refers to the ordering of the rotations in a sequence like that of Fig. 1, referred to as a Jacobi sweep. While in a standard EVD these are repeated until e.g. off-diagonal elements are suppressed below a given threshold, SMDCbR employs only one Jacobi sweep at each iteration. This equates to a fixed number of $P = M(M-1)/2$ Jacobi rotations for a matrix of size $M \times M$ to provide the unitary $\mathbf{Q}^{(i)}$ in (5),

$$\mathbf{Q}^{(i)} = \mathbf{Q}^{(i,P)}\mathbf{Q}^{(i,P-1)}\cdots\mathbf{Q}^{(i,2)}\mathbf{Q}^{(i,1)} = \prod_{n=1}^{P}\mathbf{Q}^{(i,n)}\ . \tag{9}$$

The iterative process continues for I steps, say, until $\boldsymbol{S}^{(I)}(z)$ is sufficiently diagonalised with the dominant off-diagonal column norm

$$\max_{k,\tau}\|\hat{\mathbf{s}}_k^{(I)}[\tau]\|_2 \leq \rho\quad , \tag{10}$$

below an arbitrarily small threshold ρ. This completes the SMDCbR algorithm and generates an approximate PEVD given by

$$\boldsymbol{S}^{(I)}(z) = \boldsymbol{H}^{(I)}(z)\boldsymbol{R}(z)\tilde{\boldsymbol{H}}^{(I)}(z)\ , \tag{11}$$

with

$$\boldsymbol{H}^{(I)}(z) = \prod_{i=0}^{I}\boldsymbol{U}^{(i)}(z) \tag{12}$$

based on the product defined in (9).

B. Algorithm Complexity

The search step of the SMDCbR algorithm described by (6) uses all lags of the coefficient matrix $\mathbf{S}^{(i-1)}[\tau] \in \mathbb{C}^{M\times M}$. If at the ith iteration $\mathbf{S}^{(i-1)}[\tau] = \mathbf{0}\ \forall\ |\tau| > N^{(i-1)}$, for some $N \in \mathbb{N}$, the search space encompasses $ML^{(i-1)}$ elements, where $L^{(i-1)} = (2N^{(i-1)} + 1)$ is the length of the parahermitian matrix.

For each Jacobi rotation during iteration i of SMDCbR, of which there are $M(M-1)/2$, every matrix-valued coefficient in $\boldsymbol{S}^{(i)\prime}(z)$ must be left- and right-multiplied with a sparse unitary matrix. Accounting for the multiplication of a 4-sparse $M\times M$ matrix with a non-sparse $M\times M$ matrix by $4M$ MACs, a total of $4L^{(i)}M^2(M-1)$ MACs arise to generate $\boldsymbol{S}^{(i)}(z)$ from $\boldsymbol{S}^{(i)\prime}(z)$, where $L^{(i)}$ is the length of $\boldsymbol{S}^{(i)\prime}(z)$.

Every matrix-valued coefficient in $\boldsymbol{H}^{(i)\prime}(z)$ must also be left-multiplied with a sparse unitary matrix for each rotation. A total of $2L_H^{(i)}M^2(M-1)$ MACs arise to generate $\boldsymbol{H}^{(i)}(z)$ from $\boldsymbol{H}^{(i)\prime}(z)$, where $L_H^{(i)}$ is the length of $\boldsymbol{H}^{(i)\prime}(z)$.

The total number of MACs dedicated towards the rotation step of the algorithm at each iteration is therefore given by $4L^{(i)}M^2(M-1) + 2L_H^{(i)}M^2(M-1)$.

III. Complexity and Search Space Reduction in SMDCbR

To reduce the complexity of the SMDCbR algorithm, Sec. III-A and Sec. III-B outline modification to the rotation step of the algorithm to concatenate and threshold the Jacobi rotations, respectively. Sec. III-C then details the steps to decrease complexity further by shrinking the algorithm's available search space at any given iteration.

A. Reduction in Cost of Jacobi Rotations

In the approach detailed here, Jacobi rotations are performed on the zero-lag only, before a concatenated unitary matrix is applied to the entire parahermitian and paraunitary matrices. This unitary matrix is equal to the product of all the sparse rotation matrices applied to the zero-lag. The direct application of sparse rotations to the entire polynomial matrices is therefore avoided, which results in a reduction in complexity.

For each of the $M(M-1)/2$ sparse rotations, the zero-lag and unitary matrix must be updated. Updating the zero-lag for each rotation involves left- and right-multiplication with a sparse unitary matrix, costing $8M$ MACs; thus, a full sweep requires $4M^2(M-1)$ MACs. Left-multiplying the unitary matrix for each rotation requires $4M$ MACs, therefore a full sweep encompasses $2M^2(M-1)$ MACs. Both steps combined therefore require $6M^2(M-1)$ MACs.

At each iteration, every matrix-valued coefficient in $\boldsymbol{S}^{(i)\prime}(z)$ must be left- and right-multiplied with a non-sparse unitary matrix. Accounting for the multiplication of 2 $M \times M$ matrices by M^3 MACs, a total of $2L^{(i)}M^3$ MACs arise to generate $\boldsymbol{S}^{(i)}(z)$ from $\boldsymbol{S}^{(i)\prime}(z)$.

In addition, every matrix-valued coefficient in $\boldsymbol{H}^{(i)\prime}(z)$ must be left-multiplied with a non-sparse unitary matrix at each iteration. A total of $L_H^{(i)}M^3$ MACs arise to generate $\boldsymbol{H}^{(i)}(z)$ from $\boldsymbol{H}^{(i)\prime}(z)$.

The total number of MACs dedicated to the rotation stage per iteration is therefore $2L^{(i)}M^3 + L_H^{(i)}M^3 + 6M^2(M-1) \approx (2L^{(i)} + L_H^{(i)} + 6)M^3 \approx 2L^{(i)}M^3 + L_H^{(i)}M^3$ if $\max\{L^{(i)}, L_H^{(i)}\} \gg 6$. This is approximately equal to half of the MACs required for the standard SMDCbR algorithm.

B. Thresholding of Jacobi Rotations

As the zero-lag matrix is approximately diagonalised at each iteration of SMDCbR, the off-diagonal elements of $\mathbf{S}^{(i)\prime}[0]$ eventually only contain significant values from the shifted dominant row and column found via (6), with the remaining elements being approximately zero. As these elements possess very small values, there is little merit in applying a Jacobi rotation to transfer their energy onto the diagonal.

By incorporating a threshold ν during execution of the cyclic-by-row Jacobi algorithm in SMDCbR, elements with absolute value that fall below this threshold can be ignored.

It should be noted that ignoring Jacobi rotations in this way reduces the accuracy of the algorithm; however, if ν is kept sufficiently low, this approach can reduce computation time with only a minor impact on algorithm accuracy. Furthermore, the approach described in Sec. III-A does not benefit as significantly as the original SMDCbR algorithm would when employing a threshold for Jacobi rotations, as savings in the former would only be made during the zero-lag update step; that is, only $12M$ MACs would be avoided for each missed rotation. The latter would instead experience significant complexity reduction, as each skipped Jacobi rotation would equate to the avoidance of $8ML^{(i)} + 4ML_H^{(i)}$ MACs.

C. Limited Search Strategy

Based on previous work [20], [23] to reduce the parameter search space in (6), a further cost reduction for the proposed SMDCbR versions is possible by limiting the search of maximum off-diagonal columns to a particular range of lags surrounding lag-zero. The size of this search segment can be determined by estimating the energy distribution in the parahermitian matrix in current or previous executions of the algorithm; using estimations from previous executions can be useful when the data input to the algorithm remains statistically similar between multiple executions of SMDCbR.

A narrow distribution of energy around the zero-lag can therefore lead to a large decrease in search space, and vice-versa. As the algorithm progresses, the lags closest to the zero-lag become increasingly diagonalised, therefore the average delay applied at each iteration increases; this can be accounted for by gradually widening the search space around the zero-lag as the number of iterations increases.

The search step at each iteration of the modified SMDCbR algorithm described by (6) uses the column norms of the off-diagonal elements for a reduced set of the lags of the coefficient matrix $\mathbf{S}^{(i-1)}[\tau] \in \mathbb{C}^{M\times M}$. At the ith iteration, the search step is applied to a secondary matrix $\mathbf{F}^{(i-1)}[\tau] = \mathbf{0}\ \forall\ |\tau| > \delta^{(i-1)}$, which is generated using the $L' = (2\delta^{(i-1)} + 1)$ centre lags of $\mathbf{S}^{(i-1)}[\tau]$; thus, the search space encompasses only ML' elements. If this reduced search space can adequately contain most of the energy from the original parahermitian matrix, then searching for a maximum within this search space can reduce computation time with little impact on algorithm performance. To accommodate the widening of the search space as the algorithm progresses, $\delta^{(i)}$ can be described as an increasing linear function of i.

IV. Results

To benchmark the proposed approaches, this section first defines the performance metric for evaluating differently implemented SMDCbR algorithms before setting out a simulation scenario, over which an ensemble of simulations will be performed.

A. Performance Metric

Since SMDCbR iteratively minimises off-diagonal energy, a suitable normalised metric defined in [12] is

$$E_{\text{norm}}^{(i)} = \frac{\sum_\tau \sum_{k=1}^{M} \|\hat{\mathbf{s}}_k^{(i)}[\tau]\|_2^2}{\sum_\tau \|\mathbf{R}[\tau]\|_{\text{F}}^2} \; , \tag{13}$$

with $\hat{\mathbf{s}}_k^{(i)}[\tau]$ as defined in (7). The metric $E_{\text{norm}}^{(i)}$ divides the off-diagonal energy at the ith iteration by the total energy, which remains unaltered under paraunitary operations; therefore the normalisation is performed using $\boldsymbol{R}(z)$, which is only calculated once. For a logarithmic metric, the notation $5\log_{10} E_{\text{norm}}^{(i)}$ reflects that quadratic covariance terms are squared once more for the norm calculations in (13).

B. Simulation Scenario

The simulations below have been performed over an ensemble of 10^3 instantiations of $\boldsymbol{R}(z) \in \mathbb{C}^{M\times M}$, $M = 5$, based on the randomised source model in [12]. This source model generates $\boldsymbol{R}(z) = \tilde{\boldsymbol{U}}(z)\boldsymbol{D}(z)\boldsymbol{U}(z)$, whereby the diagonal $\boldsymbol{D}(z) \in \mathbb{C}^{M\times M}$ contains the power spectral densities (PSDs) of M independent sources. These sources are spectrally shaped by innovation filters such that $\boldsymbol{D}(z)$ has an order of 120, with a restriction on the placement of zeros to limit the dynamic range of the PSDs to about 30dB. Random paraunitary matrices $\boldsymbol{U}(z) \in \mathbb{C}^{M\times M}$ of order 60 perform a convolutive mixing of these sources, such that $\boldsymbol{R}(z)$ has a full polynomial rank and an order of 240.

The SMDCbR is below referred to as the **standard**, compared to the following four proposed variations:

- **method 1**: SMDCbR with thresholding of Jacobi rotations;
- **method 2**: SMDCbR with concatenation of Jacobi rotations;
- **method 3**: SMDCbR with concatenation and thresholding of Jacobi rotations;
- **method 4**: as in method 3 but with limiting of search space.

During iterations, a stopping threshold of $\rho = 10^{-6}$ was used. The standard and proposed SMDCbR implementations are run over $I = 200$ iterations, and at every iteration step the metric defined in Sec. IV-A is recorded together with the elapsed execution time.

C. Diagonalisation

The ensemble-averaged diagonalisation according to (13) was calculated for the standard and proposed implementations.

The algorithms incorporating a threshold of $\nu = 10^{-3}$ during diagonalisation (methods 1, 3, and 4) are functionally different to those without this step, but very similar diagonalisation performance over algorithm iterations can be seen in Fig. 2. The diagonalisation versus algorithm execution time for all methods is shown in Fig. 3. The curves demonstrate that for $M = 5$, the lower complexity associated with the reduced implementations translates to a faster diagonalisation than observed for the standard realisation.

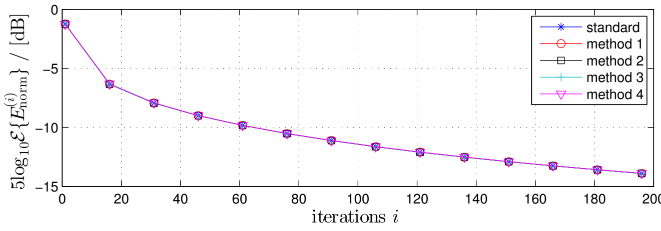

Fig. 2. Diagonalisation metric vs. algorithm iterations for the proposed and standard implementations for $M = 5$.

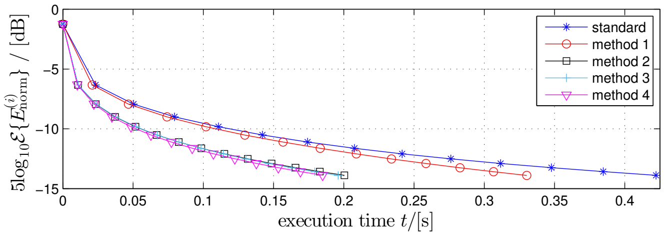

Fig. 3. Diagonalisation metric vs. algorithm execution time for the proposed and standard implementations for $M = 5$.

Thresholding of the Jacobi rotations in method 1 has a significant impact on the performance of the algorithm versus the standard algorithm; however, thresholding the same rotations in method 3 does not have such a high impact versus method 2. This is as a result of the thresholding in method 1 eliminating sparse rotations which would have been applied to all lags, while the thresholding in method 3 only eliminates sparse rotations from the zero-lag update step. Despite the small relative impact of thresholding in method 3, this algorithm performs marginally better than all algorithms with a full search space.

Fig. 4 demonstrates that the method 3 becomes more effective relative to method 1 as the spatial dimension M increases, for various values of threshold ν. It can be seen that the benefits of the proposed rotation concatenation approach exceed what could be expected from the calculated complexity reduction, owing to the well-suited nature of the utilised Matlab software for matrix multiplication. Increasing the value of ν decreases the accuracy of both methods, but also reduces the total execution time of method 1 such that it approaches the execution time of method 3. It should be noted that using a threshold of $\nu = 0$ in method 1 is equivalent to using the standard SMDCbR algorithm.

Fig. 3 also indicates the potential performance gain when limiting the search space to those lags deemed likely to contain high energy. As the algorithm's search step is a relatively inexpensive process compared with the shifting and rotation stages, the performance gain observed is small but significant. Time profiling in Matlab has shown that the use of the limited search strategy reduces the search time by 50.6% for the simulation used to generate Fig. 3. In this simulation, the search space in the proposed method has been reduced to approximate the 95% confidence interval of absolute delays applied in method 3 in a previous ensemble of 10^3, following

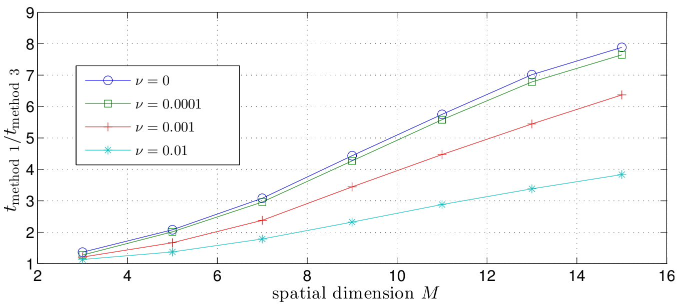

Fig. 4. Ratio of method 1 to method 3 total execution time for different spatial dimension M.

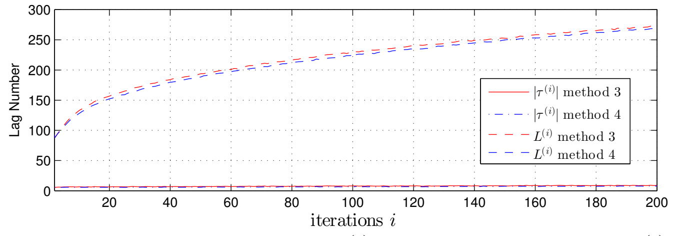

Fig. 5. Parahermitian matrix length $L^{(i)}$ and absolute applied delay $|\tau^{(i)}|$ versus iteration number for methods 3 and 4.

the assumption that these values were normally distributed. Using this information, the search space reduction parameter evolution was identified as $\delta^{(i)} = 0.15i + 24$.

D. Impact on Order of Parahermitian Matrix

The impact of search space reduction on algorithm operation was investigated for method 4, which was shown to perform the best in Sec. IV-C. Fig. 5 shows the ensemble average absolute delay $|\tau^{(i)}|$ applied to a column in iteration i of methods 3 and 4, alongside the length $L^{(i)}$ of the parahermitian matrix in both algorithms. From this figure, it is clear that the reduction in search space does not significantly impact the average delay applied — and thus the order of the parahermitian matrix — at each iteration of the proposed algorithm.

V. Conclusion

In this paper, we have proposed a series of steps to reduce the complexity of an existing cyclic-by-row SMD algorithm. We have shown through simulation that this reduction in complexity translates to an increase in diagonalisation speed of the algorithm with minimal impact on its accuracy. Furthermore, it has also been demonstrated that a reduction in the search space does not significantly impact the rate of growth of the parahermitian matrix.

When designing PEVD implementations for real applications, the potential for the proposed techniques to reduce complexity and memory requirements offers benefits. In addition, the complexity reductions proposed here can be extended to any cyclic-by-row PEVD algorithm by adapting the search and rotation operations accordingly.

Acknowledgement

Fraser Coutts is the recipient of a Caledonian Scholarship; we would like to thank the Carnegie Trust for their support.

References

[1] Z. Wang, J. G. McWhirter, and S. Weiss. Multichannel spectral factorization algorithm using polynomial matrix eigenvalue decomposition. In *Asilomar SSC*, Pacific Grove, CA, Nov. 2015.

[2] C. H. Ta and S. Weiss. A design of precoding and equalisation for broadband MIMO systems. In *Asilomar SSC*, pp. 1616–1620, Pacific Grove, CA, Nov. 2007.

[3] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, Englewood Cliffs, 1993.

[4] M. Alrmah, S. Weiss, and S. Lambotharan. An extension of the MUSIC algorithm to broadband scenarios using polynomial eigenvalue decomposition. In *EUSIPCO*, pp. 629–633, Barcelona, Spain, Aug. 2011.

[5] S. Weiss, S. Bendoukha, A. Alzin, F. Coutts, I. Proudler, and J. Chambers. MVDR broadband beamforming using polynomial matrix techniques. In *EUSIPCO*, pp. 839–843, Nice, France, Sep. 2015.

[6] S. Redif, J. McWhirter, and S. Weiss. Design of FIR paraunitary filter banks for subband coding using a polynomial eigenvalue decomposition. *IEEE Trans. SP*, 59(11):5253–5264, Nov. 2011.

[7] S. Weiss, S. Redif, T. Cooper, C. Liu, P. D. Baxter, and J. G. McWhirter. Paraunitary Oversampled Filter Bank Design for Channel Coding. *EURASIP J. Applied SP*, 2006.

[8] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix Polynomials*. Academic Press, New York, 1982.

[9] J. G. McWhirter, P. D. Baxter, T. Cooper, S. Redif, and J. Foster. An EVD Algorithm for Para-Hermitian Polynomial Matrices. *IEEE Trans. SP*, 55(5):2158–2169, May 2007.

[10] P. Vaidyanathan. Theory of optimal orthonormal subband coders. *IEEE Trans. SP*, 46(6):1528–1543, June 1998.

[11] S. Icart and P. Comon. Some properties of Laurent polynomial matrices. In *IMA Maths in Signal Proc.*, Birmingham, UK, Dec. 2012.

[12] S. Redif, S. Weiss, and J. McWhirter. Sequential matrix diagonalization algorithms for polynomial EVD of parahermitian matrices. *IEEE Trans. SP*, 63(1):81–89, Jan. 2015.

[13] J. Corr, K. Thompson, S. Weiss, J. McWhirter, and I. Proudler. Cyclic-by-row approximation of iterative polynomial EVD algorithms. In *SSPD*, pp. 1–5, Edinburgh, Scotland, Sep. 2014.

[14] J. Corr, K. Thompson, S. Weiss, J. McWhirter, S. Redif, and I. Proudler. Multiple shift maximum element sequential matrix diagonalisation for parahermitian matrices. In *IEEE SSP*, pp. 312–315, Gold Coast, Australia, June 2014.

[15] J. Corr, K. Thompson, S. Weiss, J. G. McWhirter, and I. K. Proudler. Causality-Constrained multiple shift sequential matrix diagonalisation for parahermitian matrices. In *EUSIPCO*, pp. 1277–1281, Lisbon, Portugal, Sep. 2014.

[16] Z. Wang, J. G. McWhirter, J. Corr, and S. Weiss. Multiple shift second order sequential best rotation algorithm for polynomial matrix EVD. In *EUSIPCO*, pp. 844–848, Nice, France, Sep. 2015.

[17] J. Corr, K. Thompson, S. Weiss, I. Proudler, and J. McWhirter. Row-shift corrected truncation of paraunitary matrices for PEVD algorithms. In *EUSIPCO*, pp. 849–853, Nice, France, Aug. 2015.

[18] J. Foster, J. G. McWhirter, and J. Chambers. Limiting the order of polynomial matrices within the SBR2 algorithm. In *IMA Maths in Signal Proc.*, Cirencester, UK, Dec. 2006.

[19] C. H. Ta and S. Weiss. Shortening the order of paraunitary matrices in SBR2 algorithm. In *6th International Conference on Information, Communications & Signal Processing*, pages 1–5, Singapore, Dec. 2007.

[20] J. Corr, K. Thompson, S. Weiss, I. Proudler, and J. McWhirter. Shortening of paraunitary matrices obtained by polynomial eigenvalue decomposition algorithms. In *SSPD*, Edinburgh, Scotland, Sep. 2015.

[21] F. Coutts, J. Corr, J. Thompson, S. Weiss, J. Proudler, and J. McWhirter. Memory and Complexity Reduction in Parahermitian Matrix Manipulations of PEVD Algorithms. Submitted to *EUSIPCO*, Budapest, Hungary, August 2016.

[22] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, Maryland, 3rd ed., 1996.

[23] Z. Wang, J. McWhirter, J. Corr, and S. Weiss. Order-controlled multiple shift sbr2 algorithm for para-hermitian polynomial matrices. In *IEEE*, Rio de Janeiro, Brazil, July 2016.