Memory and Complexity Reduction in Parahermitian Matrix Manipulations of 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*—A number of 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 will diagonalise a parahermitian matrix via paraunitary operations. This paper addresses savings — both computationally and in terms of memory use — that exploit the parahermitian structure of the matrix being decomposed, and also suggests an implicit trimming approach to efficiently curb the polynomial order growth usually observed during iterations of the PEVD algorithms. We demonstrate that with the proposed techniques, both storage and computations can be significantly reduced, impacting on a number of broadband multichannel problems.**

I. Introduction

Broadband multichannel problems can be elegantly expressed using polynomial matrix formulations. This includes polyphase analysis and synthesis matrices for filter banks [1], channel coding [2], [3], broadband MIMO precoding and equalisation [4], optimal subband coding [5], broadband angle of arrival estimation [6], [7], broadband beamforming [8], [9], and multichannel factorisation [10] to name but a few. These problems generally involve parahermitian polynomial matrices, where $\boldsymbol{R}(z)$ is identical to its parahermitian $\tilde{\boldsymbol{R}}(z) = \boldsymbol{R}^{\mathrm{H}}(z^{-1})$, i.e. a Hermitian transposed and time-reversed version of itself [1].

Similar to the way the eigenvalue decomposition (EVD) presents an optimal tool for many narrowband problems involving covariance matrices, a factorisation for parahermitian polynomial matrices is required for the broadband case. Therefore as an extension of the EVD, a polynomial matrix EVD (PEVD) has been defined in [11], [12] for space-time covariance matrices which can be approximately diagonalised and spectrally majorised [13] by finite impulse response (FIR) paraunitary matrices [14].

PEVD algorithms include the original second order sequential best rotation (SBR2) algorithm [12], sequential matrix diagonalisation (SMD) [15] and various evolutions of the algorithm families [16]–[18]. All of these algorithms approximately diagonalise the parahermitian matrix in an iterative manner, stopping when some suitable threshold is reached. Both SBR2 and SMD are computationally costly to compute, and any cost savings that can be applied to these algorithms will provide an advantage for applications.

Efforts to reduce the algorithmic cost have mostly been focused on the trimming of polynomial matrix factors to curb growth in order [12], [19]–[21], which translates directly into a growth of computational complexity and memory storage requirements. Recent efforts [19], [21] have been dedicated to the paraunitary matrices whilst [12], [15] consider the parahermitian matrices after every iteration.

Here we exploit the natural symmetry of the parahermitian matrix structure and only store one half of its elements, and show how this can be reconciled with the required row- and column shift operations in PEVD algorithms. Trimming of the parahermitian matrix is integrated into every iteration, such that no matrix multiplications are executed on terms that will subsequently be discarded.

Below, Sec. II will provide a brief overview over the SMD method as a representative example of PEVD algorithms. The proposed approach for storing and operating on a reduced parahermitian matrix, including an integrated trimming strategy, 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

This section reviews aspects of the SMD algorithm [15] as a representive of iterative PEVD schemes in Sec. II-A, with an assessment of the main algorithmic cost and memory requirements in Sec. II-B.

A. Algorithm Overview

The SMD 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 step is used to move the column with the largest energy in its off-diagonal elements to the zero lag; then an EVD diagonalises the zero lag matrix, transferring the shifted off-diagonal energy onto the diagonal.

The SMD algorithm is initialised with a 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 $\mathbf{Q}^{(0)}$ - which is obtained from 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, $i = 1, 2, \ldots L$, the SMD algorithm calculates a transformation of the form

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

in which

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

The product in (2) consists of an elementary paraunitary 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

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

and

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

The selection of $\boldsymbol{\Lambda}^{(i)}(z)$ and $\mathbf{Q}^{(i)}$ in the ith iteration depends on the position of the dominant off-diagonal column in $\boldsymbol{S}^{(i-1)}(z) \bullet\!\!-\!\!\circ\ \mathbf{S}^{(i-1)}[\tau]$, as identified by the parameter set

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

where

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

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 its parahermitian form, the shifting process in (4) moves both the dominant off-diagonal row and the dominant off-diagonal column into the zero-lag coefficient matrix and so 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 modal matrix $\mathbf{Q}^{(i)}$ in (5) that diagonalises $\mathbf{S}^{(i)\prime}[0]$ by means of an ordered EVD.

The iterative process — which has been shown to converge [15] — continues for I steps, say, until $\boldsymbol{S}^{(I)}(z)$ is sufficiently diagonalised with the dominant off-diagonal column norm

$$\max_{k,\tau} \|\hat{\mathbf{s}}_k^{(I)}[\tau]\|_2 \leq \rho\quad , \tag{8}$$

where the value of ρ is chosen to be arbitrarily small. This completes the SMD algorithm and generates an approximate PEVD given by

$$\boldsymbol{S}^{(I)}(z) = \boldsymbol{H}^{(I)}(z)\boldsymbol{R}(z)\tilde{\boldsymbol{H}}^{(I)}(z)\ . \tag{9}$$

Concatenation of the elementary paraunitary matrices

$$\begin{aligned}\boldsymbol{H}^{(I)}(z) &= \boldsymbol{U}^{(I)}(z)\boldsymbol{U}^{(I-1)}(z)\cdots\boldsymbol{U}^{(1)}(z)\boldsymbol{U}^{(0)}(z)\\ &= \prod_{i=0}^{I-1}\boldsymbol{U}^{(I-i)}(z)\end{aligned} \tag{10}$$

extracts the PU matrix $\boldsymbol{H}^{(I)}(z)$ for (9).

B. Complexity and Memory Requirements

If at the ith iteration $\mathbf{S}^{(i-1)}[\tau] = \mathbf{0}\ \forall\ |\tau| > N^{(i-1)}$, the memory to store $\boldsymbol{S}^{(i-1)}(z)$ must hold $(2N^{(i-1)}+1)M^2$ coefficients. The maximum column search requires the calculation of $(M-1)M(2N^{(i-1)}+1)$ multiply accumulates (MACs) for the modified column norms according to (7).

During the ith iteration, the polynomial order growth leads to $N^{(i)} = N^{(i-1)} + |\tau^{(i)}|$, and the calculation of (4) is implemented as a combination of two block memory moves: one for the rows of $\mathbf{S}^{(i-1)}[\tau]$, and one for the columns. The number of coefficients of $\mathbf{S}^{(i-1)}[\tau] \in \mathbb{C}^{M\times M}$, $\mathbf{S}^{(i-1)}[\tau] = \mathbf{0}\ \forall\ |\tau| > N^{(i-1)}$, to be moved can therefore be approximated by $2(2N^{(i-1)}+1)M \approx 4N^{(i-1)}M$, assuming $N^{(i-1)}$ is large.

For (5), every matrix-valued coefficient in $\boldsymbol{S}^{(i)\prime}(z)$ must be left- and right-multiplied with a unitary matrix. Accounting for a multiplication of 2 $M \times M$ matrices by M^3 MACs, a total of $(2(2N^{(i)}+1)M^3) \approx 4N^{(i)}M^3$ MACs arise to generate $\boldsymbol{S}^{(i)}(z)$ from $\boldsymbol{S}^{(i)\prime}(z)$. It is therefore this latter cost that dominates the computational cost of the ith iteration step.

III. Reduced Parahermitian Matrix Representation

Based on the reduced matrix representation of a parahermitian matrix defined in Sec. III-A, Sec. III-B outlines modification to the parameter search, which is then applied for modified shifts in Sec. III-C. Rotation with an integrated truncation is proposed in Sec. III-D, and the resource requirements are compared to the previous standard approach in Sec. III-E.

A. Matrix Structure

By segmenting a parahermitian matrix $\boldsymbol{R}(z)$, it is possible to write

$$\boldsymbol{R}(z) = \boldsymbol{R}^{(-)}(z) + \mathbf{R}[0] + \boldsymbol{R}^{(+)}(z)\ , \tag{11}$$

where $\mathbf{R}[0]$ is the zero lag matrix, $\boldsymbol{R}^{(+)}(z)$ contains terms for positive lag elements only, and $\boldsymbol{R}^{(-)}(z) = \tilde{\boldsymbol{R}}^{(+)}(z)$. It is therefore sufficient to record half of $\boldsymbol{R}(z)$, which here without loss of generality is $\mathbf{R}[0] + \boldsymbol{R}^{(+)}(z)$. As an example, Fig. 1 demonstrates the reduction of a 5×5 matrix with maximum lag $N = 3$.

In the initialisation step of SMD, the matrix $\mathbf{R}[0]$ would be diagonalised, and in every subsequent step, $\boldsymbol{S}^{(i)}(z)$ is split analogously to (11) such that only its causal part is recorded.

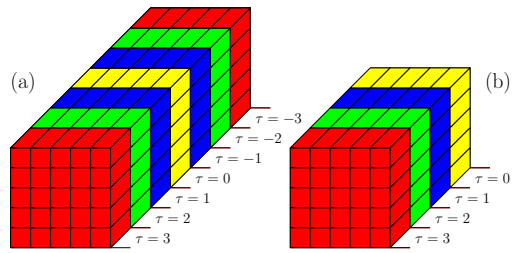

Fig. 1. (a) Full and (b) reduced representation of the parahermitian matrix $\boldsymbol{R}(z)$ for $N = 3$.

B. Modified Search Strategy

To find the correct shift parameters for the SMD algorithm, (6) can be used directly but with a restriction of the search space for column norms to $\tau \geq 0$, such that $\tau^{(i)} \geq 0$ is also imposed as a constraint. This requires only half the search space of the standard SMD approach, but neglects to search column norms for negative time lags, hence yielding a solution that is equivalent to the causally-constrained SMD algorithm [18].

If column norms for negative lags values $\tau < 0$ are to be included in the search, then due to its parahermitian structure, searching column norms of $\mathbf{S}^{(i-1)}[\tau]$ for $\tau < 0$ is equivalent to searching row norms for $\tau \geq 0$. If a modified row norm for the kth row is defined as

$$\|\hat{\mathbf{s}}^{(i-1)}_{(\mathrm{r}),k}[\tau]\|_2 = \sqrt{\sum_{m=1, m\neq k}^{M} |\mathbf{s}^{(i-1)}_{k,m}[\tau]|^2} \,, \tag{12}$$

then the modified parameter search is

$$\{k^{(i)}, \tau^{(i)}\} = \arg\max_{k,\tau} \left\{ \|\hat{\mathbf{s}}^{(i-1)}_{k}[\tau]\|_2 \,,\; \|\hat{\mathbf{s}}^{(i-1)}_{(\mathrm{r}),k}[-\tau]\|_2 \right\} \,, \tag{13}$$

whereby both $\|\hat{\mathbf{s}}^{(i-1)}_{k}[\nu]\|_2$ and $\|\hat{\mathbf{s}}^{(i-1)}_{(\mathrm{r}),k}[\nu]\|_2$ are only evaluated for arguments $\nu \geq 0$. If (13) returns $\tau^{(i)} > 0$, then as previously the $k^{(i)}$th column is to be shifted. If $\tau^{(i)} < 0$, it is the $k^{(i)}$th row that requires shifting by $-\tau^{(i)}$; alternatively due to the parahermitian property, the $k^{(i)}$th column can also be shifted by $\tau^{(i)}$, thus covering both cases of positive and negative shifts.

C. Shifting Approach

The delay step (4) in the SMD algorithm can be performed with the reduced parahermitian matrix representation in the ith iteration by shifting either the $k^{(i)}$th column or row — whichever has the greater modified norm according to (7) or (12) — by $|\tau^{(i)}|$ coefficients to the zero lag. Elements that are shifted beyond the zero lag, i.e. outside the recorded half-matrix, have to be stored as parahermitian (i.e. Hermitian transposed and time reversed) and appended onto the $k^{(i)}$th row or column of the shifted matrix at lag-zero. The concatenated row or column is then shifted by $|\tau^{(i)}|$ elements towards increasing τ.

An example of the shift operation is depicted in Fig. 2 for the case of $\boldsymbol{S}^{(i-1)}(z) \in \mathbb{C}^{5\times 5}$ with parameters $k^{(i)} = 2$ and

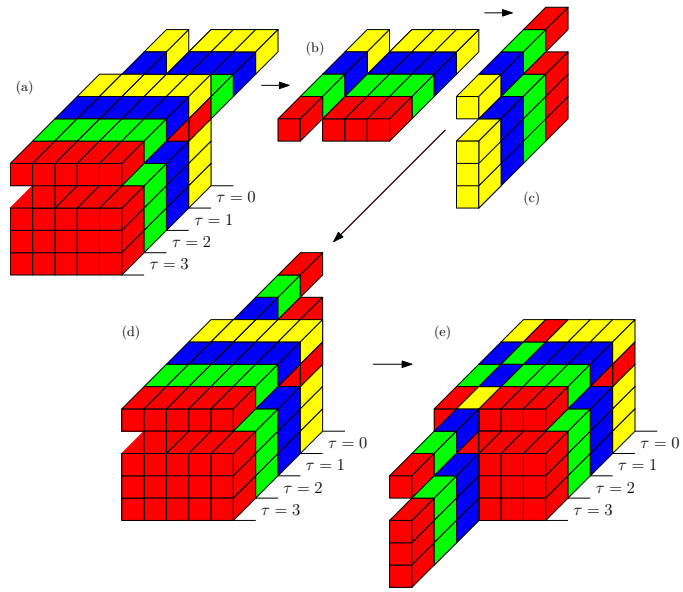

Fig. 2. Example for a matrix where in the ith iteration a modified row norm is maximum: (a) the row is shifted, with non-diagonal elements in the $k^{(i)}$th row past the zero lag (b) extracted and (c) parahermitian transposed; (d) these elements are appended to the $k^{(i)}$th column at zero-lag and (e) shifted in the opposite direction with all off-diagonal column elements.

$\tau^{(i)} = -3$. Because of the negative sign of $\tau^{(i)}$, it is here the 2nd row that has to be shifted first, followed by the 2nd column shifted in the opposite direction. Note that because the row and column shifts operate in opposite directions , the polynomial in the $k^{(i)}$th position along the diagonal remains unaltered. An efficient implementation will therefore exclude this polynomial from otherwise spurious shift operations, as shown in the example of Fig. 2.

D. Truncation and Rotation

Following this shifting procedure, for the 2nd step of the SMD iteration an ordered EVD of the lag-zero matrix $\mathbf{S}^{(i)\prime}[0]$ is perfomed, yielding the modal matrix $\mathbf{Q}^{(i)}$ that has to be applied to all lags $0 \leq \tau \leq N^{(i)}$.

To curb the growth in order of $\boldsymbol{S}^{(i)}(z)$ at every iteration, trimming of matrices $\mathbf{S}^{(i)}[\tau]$ for outer lag values has been advocated previously [12], [15], [20]. This trimming is based on a threshold ϵ applied to the Frobenius norm $\|\cdot\|_{\mathrm{F}}$, whereby $\boldsymbol{S}^{(i)}(z)$ is reduced to an order $\tilde{N}^{(i)}$ such that

$$\|\mathbf{S}^{(i)}[\tau]\|_{\mathrm{F}} < \epsilon \;\forall\; \tau > \tilde{N}^{(i)} \,. \tag{14}$$

Since $\mathbf{Q}^{(i)}$ is unitary, $\|\mathbf{S}^{(i)}[\tau]\|_{\mathrm{F}} = \|\mathbf{S}^{(i)\prime}[\tau]\|_{\mathrm{F}}$. Truncation can therefore be performed on $\boldsymbol{S}^{(i)\prime}(z)$ prior to the computationally expensive rotation to create $\boldsymbol{S}^{(i)}(z)$.

There is also an option to combine the calculation of the Frobenius norm at the ith iteration with the calculation of column and row norms, which will be required for the parameter search of $\{k^{(i+1)},\; \tau^{(i+1)}\}$ at the $(i+1)$th iteration. By keeping a record of these modified norms and tracking column and row shifts, every modified norm according to (7) and (12) only changes in a single coefficient. Tracking norms

TABLE I
APPPROXIMATE RESOURCE REQUIREMENTS.

Method	Complexity	Storage	Memory Moves
standard	$4N^{(i)}M^3$	$(2N^{(i)}+1)M^2$	$(4N^{(i-1)}+2)M$
proposed	$2N^{(i)}M^3$	$(N^{(i)}+1)M^2$	$(2N^{(i-1)}+2)(M-1)$

therefore is particular advantageous for larger dimensions M when factorising $\boldsymbol{R}(z) \in \mathbb{C}^{M\times M}$.

E. Reduction in Memory and Computational Cost

The memory required to store the causal part of $\boldsymbol{S}^{(i)}(z)$ at the ith iteration is equal to $(N^{(i)}+1)M^2$ coefficients, and therefore approximately half of what a full storage needs. During the ith iteration, the first delay step involves $2(N^{(i-1)}+1)(M-1)$ coefficients to be shifted in memory, as opposed to $2(2N^{(i-1)}+1)M$ for a full matrix representation. Therefore, the number of coefficient moves during the shift step is also halved using the proposed approach.

In terms of multiply-accumulates, the rotation operation with $\mathbf{Q}^{(i)}$ during the 2nd step of the ith iteration requires $2M^3(\tilde{N}^{(i)}+1)$ MACs, saving more than half of the operations executed during the standard approach outlined in Sec. II-B. The various aspects of resource requirements are summarised in Tab. I.

For $i \gg 1$, simulations in [15] indicate that the order of the parahermitian matrix $\boldsymbol{S}^{(i)}(z)$ does no longer increase when trimmed *after* every iteration. Therefore,

$$\tilde{N}^{(i)} \approx N^{(i)} - |\tau^{(i)}| \tag{15}$$

and the additional cost reduction due to trimming $\boldsymbol{S}^{(i)\prime}(z)$ *prior* to the 2nd iteration step saves an additional $2M^3|\tau^{(i)}|$ MACs.

IV. RESULTS

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

A. Performance Metric

Since SMD iteratively minimises off-diagonal energy, a suitable normalised metric defined in [15] is

$$E^{(i)}_{\text{norm}} = \frac{\sum_\tau \sum_{k=1}^{M} \|\hat{\mathbf{s}}^{(i)}_k[\tau]\|_2^2}{\sum_\tau \|\mathbf{R}[\tau]\|_{\text{F}}^2}\,, \tag{16}$$

which divides the off-diagonal energy at the ith iteration by the total energy. Since the total energy remains unaltered under paraunitary operations, the normalisation is performed by $\boldsymbol{R}(z)$ which can be calculated once, rather than by evaluating $\boldsymbol{S}^{(i)}(z)$ at every iteration. The definition of $\hat{\mathbf{s}}^{(i)}_k[\tau]$ is given in (7). For a logarithmic metric, the notation $5\log_{10} E^{(i)}_{\text{norm}}$ reflects that quadratic covariance terms are squared once more for the norm calculations in (16).

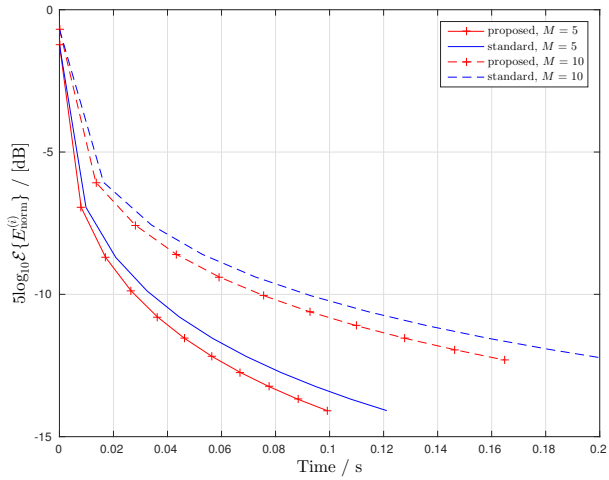

Fig. 3. Diagonalisation metric vs. algorithm execution time for the proposed reduced and standard SMD implementations for $M \in \{5;\ 10\}$.

B. Simulation Scenario

The simulations below have been performed over an ensemble of 10^3 instantiations of $\boldsymbol{R}(z) \in \mathbb{C}^{M\times M}$, $M \in \{5;\ 10\}$, based on the randomised source model in [15]. 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.

During iterations, a truncation parameter of $\mu = 10^{-6}$ and a stopping threshold of $\rho = 10^{-6}$ were used. The standard and proposed SMD 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 (16) was calculated for both the standard and reduced SMD implementation. While both algorithms are functionally identical and exhibit the same diagonalisation performance over algorithm iterations, the cost per iteration step for both methods is shown in Fig. 3. The curves demonstrate that for $M \in \{5;\ 10\}$, the lower complexity associated with the reduced SMD implementation translates to a faster diagonalisation than observed for the standard SMD realisation. Using a matrix with a larger spatial dimension of $M = 10$ versus $M = 5$ results in poorer diagonalisation for both algorithms, but the same relative performance increase is still seen for the proposed reduced approach.

Simulated in Matlab, the results in Fig. 3 are not as impressive as the computational savings suggested by Tab. I. This is partially due to the fact that both the reduced and standard implementation still have to update and maintain a record of

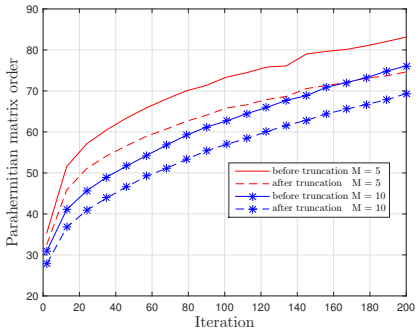

Fig. 4. Parahermitian matrix order before and after truncation vs. algorithm iteration for the proposed reduced implementation for $M \in \{5;\ 10\}$.

the paraunitary matrix as iterations proceed, requiring costly convolutions of paraunitary polynomial matrices. If these paraunitary matrices do not need to be extracted, then eliminating the cost for the paraunitary matrix update will widen the performance gap between the methods in Tab. I. Using the Matlab profiler further shows that the execution time for matrix multiplications (i.e., the number of matrix multiplications) has been substantially reduced by the proposed method; while Matlab is optimised for such matrix multiplication, its shifting of memory is as not efficient and dominates the execution time. Despite this, Tab. I indicates a 20% reduction in cost when using the proposed over the standard SMD implementation.

D. Truncation of Parahermitian Matrix

When using the proposed reduced representation, the impact of parahermitian matrix truncation according to (14) on the matrix order can be seen in Fig. 4. Note, the first two iterations have been omitted for clarity. By moving the truncation step to before the rotation stage in the proposed approach, it is clear that a significant number of redundant MAC operations have been avoided.

V. Conclusion

The symmetry in the parahermitian matrix has been exploited when calculating a polynomial EVD, which has been exemplified here by a focus on the SMD algorithm. We have proposed a reduced matrix representation which only records its causal part; this approach can produce the same accuracy of decomposition as a standard matrix representation in the SMD algorithm, but with increased efficiency with respect to memory use and computational complexity. Simulation results underline that the same diagonalisation performance can be achieved by both methods, but within a shorter execution time for the approach based on a reduced representation.

When designing PEVD implementations for real applications, the potential for the proposed techniques to reduce complexity and memory requirements therefore offers benefits without deficits w.r.t. important performance metrics such as the diagonalisation of the SMD algorithm. The reduced representation of parahermitian matrices proposed here can be extended to any PEVD algorithm by adapting the shift 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] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, Englewood Cliffs, 1993.

[2] C. Liu, C. H. Ta, and S. Weiss. Joint precoding and equalisation design using oversampled filter banks for dispersive channels with correlated noise. In *IEEE Workshop on Signal Processing Systems*, pp. 232–236, Shanghai, China, Oct. 2007.

[3] S. Weiss, S. Redif, T. Cooper, C. Liu, P. D. Baxter, and J. G. McWhirter. Paraunitary oversampled filter bank design for channel coding. *EURASIP Journal of Applied Signal Processing*, 2006.

[4] C. H. Ta and S. Weiss. A design of precoding and equalisation for broadband MIMO systems. In *41st Asilomar Conf. Signals, Systems and Computers*, pp. 1616–1620, Pacific Grove, CA, USA, Nov. 2007.

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

[6] M. Alrmah, S. Weiss, and S. Lambotharan. An extension of the music algorithm to broadband scenarios using polynomial eigenvalue decomposition. In *19th European Signal Processing Conference*, pp. 629–633, Barcelona, Spain, Aug. 2011.

[7] S. Weiss, M. Alrmah, S. Lambotharan, J. McWhirter, and M. Kaveh. Broadband angle of arrival estimation methods in a polynomial matrix decomposition framework. In *IEEE 5th Int. Workshop Comp. Advances in Multi-Sensor Adaptive Proc.*, St. Martin, pp. 109–112, Dec. 2013.

[8] A. Alzin, F. Coutts, J. Corr, S. Weiss, I. K. Proudler, and J. A. Chambers. Adaptive broadband beamforming with arbitrary array geometry. In *IET/EURASIP Intelligent Signal Processing*, London, UK, Dec. 2015.

[9] S. Weiss, S. Bendoukha, A. Alzin, F. Coutts, I. Proudler, and J. Chambers. MVDR broadband beamforming using polynomial matrix techniques. In *23rd European Signal Processing Conference*, pp. 839–843, Nice, France, Sep. 2015.

[10] Z. Wang, J. G. McWhirter, and S. Weiss. Multichannel spectral factorization algorithm using polynomial matrix eigenvalue decomposition. In *49th Asilomar Conf. Signals, Systems and Computers*, Pacific Grove, CA, Nov. 2015.

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

[12] 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(5):2158–2169, May 2007.

[13] P. Vaidyanathan. Theory of optimal orthonormal subband coders. *IEEE Transactions on Signal Processing*, 46(6):1528–1543, June 1998.

[14] S. Icart, P. Comon. Some properties of Laurent polynomial matrices. In *Conf. Math. Signal Proc.*, Birmingham, UK, Dec. 2012.

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

[16] 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 Workshop on Statistical Signal Processing*, pp. 312–315, Gold Coast, Australia, June 2014.

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

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

[19] J. Corr, K. Thompson, S. Weiss, I. Proudler, and J. McWhirter. Row-shift corrected truncation of paraunitary matrices for PEVD algorithms. In *Europ. Signal Proc. Conf.*, pp. 849–853, Nice, France, Sep. 2015.

[20] J. Foster, J. G. McWhirter, and J. Chambers. Limiting the order of polynomial matrices within the SBR2 algorithm. In *IMA Int. Conf. Mathematics in Signal Processing*, Cirencester, UK, Dec. 2006.

[21] C. H. Ta and S. Weiss. Shortening the order of paraunitary matrices in SBR2 algorithm. In *6th Int. Conf. Information, Communications & Signal Processing*, pp. 1–5, Singapore, Dec. 2007.