Divide-and-Conquer Sequential Matrix Diagonalisation for Parahermitian Matrices

Fraser K. Coutts*, Jamie Corr*, Keith Thompson*, Ian K. Proudler*,†, Stephan Weiss*
* Department of Electronic & Electrical Engineering, University of Strathclyde, Glasgow, Scotland
† School of Electrical, Electronics & Systems Engineering, Loughborough Univ., Loughborough, UK
{fraser.coutts,jamie.corr,keith.thompson,ian.proudler,stephan.weiss}@strath.ac.uk

***Abstract*—A number of algorithms capable of iteratively calculating 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. Inspired by the existence of low complexity divide-and-conquer solutions to eigenproblems, this paper addresses a divide-and-conquer approach to the PEVD utilising the sequential matrix diagonalisation (SMD) algorithm. We demonstrate that with the proposed techniques, encapsulated in a novel algorithm titled divide-and-conquer sequential matrix diagonalisation (DC-SMD), algorithm complexity can be significantly reduced. This reduction impacts on a number of broadband multichannel problems, including those involving large arrays.**

I. Introduction

Polynomial matrix formulations can be used to express broadband multichannel problems. Examples include broadband MIMO precoding and equalisation [1], polyphase analysis and synthesis matrices for filter banks [2], and broadband beamforming [3], [4]. Typically, these problems involve parahermitian polynomial matrices, which are identical to their parahermitian conjugate, i.e., $\boldsymbol{R}(z) = \tilde{\boldsymbol{R}}(z) = \boldsymbol{R}^{\mathrm{H}}(1/z^*)$ [2]. Matrix $\boldsymbol{R}(z)$ can arise as the z-transform of a space-time covariance matrix $\mathbf{R}[\tau]$.

A polynomial matrix eigenvalue decomposition (PEVD) has been defined as an extension of the eigenvalue decomposition (EVD) to parahermitian polynomial matrices in [5], [6]. The PEVD uses finite impulse response (FIR) paraunitary matrices [7] to approximately diagonalise and spectrally majorise [8] a space-time covariance matrix.

Existing PEVD algorithms include the second-order sequential best rotation (SBR2) algorithm [6], sequential matrix diagonalisation (SMD) [9], and various evolutions of the algorithm families [10]–[12]. Each of these algorithms use an iterative approach to approximately diagonalise a parahermitian matrix. For matrices of high dimensionality, these algorithms can be computationally costly to implement; therefore, any cost savings will be advantageous for applications.

Efforts to reduce the cost of PEVD algorithms include techniques for the trimming of polynomial matrices to curb growth in order [6], [13]–[15], which translates directly into a growth of computational complexity and memory storage requirements. Recently, techniques in [16], [17] have successfully reduced the complexity of existing PEVD algorithms through the removal of algorithmic redundancy.

Research in [18]–[20] has demonstrated that complexity reduction can be obtained by using a divide-and-conquer approach to eigenproblems. Inspired by this work, here we describe a divide-and-conquer approach for the PEVD, which can be utilised to reduce algorithm complexity with minimal loss in accuracy. The framework of the developed algorithm — titled divide-and-conquer sequential matrix diagonalisation (DC-SMD) — is based on the SMD algorithm.

Below, Sec. II will provide a brief overview over the SMD algorithm. The proposed divide-and-conquer approach is 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 [9] 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 $\boldsymbol{R}(z) \in \mathbb{C}^{M\times M}$ and its associated coefficient matrix, $\mathbf{R}[\tau]$.

Upon initialisation, the algorithm diagonalises the lag-zero coefficient matrix $\mathbf{R}[0]$ by means of its modal matrix $\mathbf{Q}^{(0)}$; i.e., $\boldsymbol{S}^{(0)}(z) = \mathbf{Q}^{(0)}\boldsymbol{R}(z)\mathbf{Q}^{(0)\mathrm{H}}$. The unitary $\mathbf{Q}^{(0)}$ — obtained from the EVD of the lag-zero slice $\mathbf{R}[0]$ — is applied to all coefficient matrices $\mathbf{R}[\tau]\ \forall\ \tau$, and initialises $\boldsymbol{H}^{(0)}(z) = \mathbf{Q}^{(0)}$.

In the ith step, $i = 1, 2, \ldots I$, the SMD algorithm computes

$$\begin{aligned}\boldsymbol{S}^{(i)}(z) &= \boldsymbol{U}^{(i)}(z)\boldsymbol{S}^{(i-1)}(z)\tilde{\boldsymbol{U}}^{(i)}(z)\\ \boldsymbol{H}^{(i)}(z) &= \boldsymbol{U}^{(i)}(z)\boldsymbol{H}^{(i-1)}(z)\quad ,\end{aligned} \tag{1}$$

in which

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

The product in (2) consists of a 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. For subsequent discussion, it is convenient to define intermediate variables $\boldsymbol{S}^{(i)\prime}(z)$ and $\boldsymbol{H}^{(i)\prime}(z)$ where

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

and

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

Matrices $\mathbf{\Lambda}^{(i)}(z)$ and $\mathbf{Q}^{(i)}$ are selected based 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

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

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, $\mathbf{S}^{(i-1)}[\tau]$.

The shifting process in (4) moves the dominant off-diagonal row and column into the zero lag coefficient matrix $\mathbf{S}^{(i)\prime}[0]$. The off-diagonal energy in the shifted row and column is then transferred onto the diagonal by the unitary matrix $\mathbf{Q}^{(i)}$ in (5), which diagonalises $\mathbf{S}^{(i)\prime}[0]$ by means of an ordered EVD.

Iterations continue for I steps until $\boldsymbol{S}^{(I)}(z)$ is sufficiently diagonalised with dominant off-diagonal column norm

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

where the value of ϵ is chosen to be arbitrarily small. On completion, SMD generates an approximate PEVD given by

$$\boldsymbol{D}(z) = \boldsymbol{S}^{(I)}(z) = \boldsymbol{F}(z)\boldsymbol{R}(z)\tilde{\boldsymbol{F}}(z) \,, \tag{9}$$

where $\boldsymbol{F}(z)$ is a concatenation of the paraunitary matrices:

$$\boldsymbol{F}(z) = \boldsymbol{H}^{(I)}(z) = \boldsymbol{U}^{(I)}(z)\cdots\boldsymbol{U}^{(0)}(z) = \prod_{i=0}^{I} \boldsymbol{U}^{(I-i)}(z) \,. \tag{10}$$

Truncation of outer coefficients of $\boldsymbol{H}^{(i)}(z)$ with small Frobenius norm $\|\cdot\|_{\mathrm{F}}$ is used to limit growth in order, whereby the maximum and minimum lags of $\boldsymbol{H}^{(i)}(z)$ at iteration i are reduced from τ_1 and τ_2 to $\tilde{\tau}_1$ and $\tilde{\tau}_2$, respectively, such that

$$\textstyle\sum_{\tau=\tilde{\tau}_1+1}^{\tau_1} \|\mathbf{H}^{(i)}[\tau]\|_{\mathrm{F}}^2 < \frac{\mu \sum_\tau \|\mathbf{H}^{(i)}[\tau]\|_{\mathrm{F}}^2}{2} > \textstyle\sum_{\tau=\tau_2}^{\tilde{\tau}_2-1} \|\mathbf{H}^{(i)}[\tau]\|_{\mathrm{F}}^2 \,. \tag{11}$$

Truncation of $\boldsymbol{S}^{(i)}(z)$ is similar, with its maximum and minimum lags reduced from τ_3 and $-\tau_3$ to $\tilde{\tau}_3$ and $-\tilde{\tau}_3$, such that

$$\textstyle\sum_{\tau=\tilde{\tau}_3+1}^{\tau_3} \|\mathbf{S}^{(i)}[\tau]\|_{\mathrm{F}}^2 < \frac{\mu \sum_\tau \|\mathbf{S}^{(i)}[\tau]\|_{\mathrm{F}}^2}{2} \,. \tag{12}$$

B. Algorithm Complexity

At the ith iteration, the length of $\boldsymbol{S}^{(i)\prime}(z)$ is equal to $L\{\boldsymbol{S}^{(i)\prime}\}$, where $L\{\cdot\}$ computes the length of a polynomial matrix. 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 $2L\{\boldsymbol{S}^{(i)\prime}\}M^3$ MACs arise to generate $\boldsymbol{S}^{(i)}(z)$. Every matrix-valued coefficient in $\boldsymbol{H}^{(i)\prime}(z)$ must also be left-multiplied with a unitary matrix; thus, a total of $L\{\boldsymbol{H}^{(i)\prime}\}M^3$ MACs arise to generate $\boldsymbol{H}^{(i)}(z)$. The cumulative complexity of the SMD algorithm over I iterations can therefore be approximated as $M^3 \sum_{i=0}^{I} (2L\{\boldsymbol{S}^{(i)\prime}\} + L\{\boldsymbol{H}^{(i)\prime}\})$.

III. Divide-and-Conquer Approach

Inspired by the development of divide-and-conquer solutions to eigenproblems in [18]–[20], this section outlines the components of a novel divide-and-conquer sequential matrix diagonalisation PEVD algorithm — which is summarised in Sec. III-A. Sec. III-B and Sec. III-C explain the key stages of this algorithm by detailing the divide and conquer steps, respectively. The complexity requirements of this algorithm are derived in Sec. III-D.

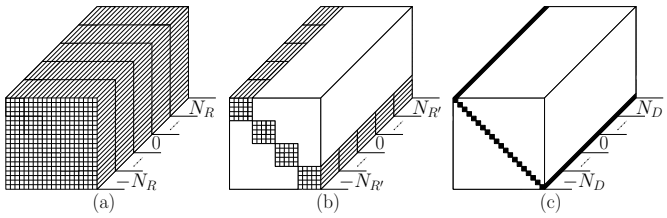

Fig. 1. (a) Original matrix $\mathbf{R}[\tau] \in \mathbb{C}^{20\times 20}$, (b) segmented result $\mathbf{R}'[\tau]$, and (c) diagonalised output $\mathbf{D}[\tau]$. N_R, $N_{R'}$, and N_D are the maximum lags for matrices $\mathbf{R}[\tau]$, $\mathbf{R}'[\tau]$, and $\mathbf{D}[\tau]$, respectively.

A. Divide-and-Conquer Sequential Matrix Diagonalisation

The DC-SMD algorithm diagonalises a parahermitian matrix $\boldsymbol{R}(z) \in \mathbb{C}^{M\times M}$ via a number of paraunitary operations. An output diagonal matrix $\boldsymbol{D}(z)$ contains the eigenvalues, and $\boldsymbol{F}(z)$ contains the corresponding eigenvectors.

While the SMD algorithm attempts to diagonalise an entire $M \times M$ parahermitian matrix at once, the DC-SMD algorithm first divides the matrix into a number of smaller, independent parahermitian matrices, before diagonalising — or conquering — each matrix separately. For example, a matrix $\boldsymbol{R}(z) \in \mathbb{C}^{20\times 20}$ might be divided into four 5×5 parahermitian matrices, each of which can be diagonalised independently. Fig. 1 shows the state of the parahermitian matrix at each stage of the process for this example.

If matrix $\boldsymbol{R}(z)$ is of large spatial dimension, an algorithm named sequential matrix segmentation (SMS) is used to recursively divide the matrix into multiple independent parahermitian matrices. Each of these is stored on the diagonal of matrix $\boldsymbol{R}'(z)$; thus, $\boldsymbol{R}'(z)$ is block diagonal by construction. The matrices $\boldsymbol{T}(z)$ — which SMS generates to divide each $\hat{\boldsymbol{R}}(z)$ — are concatenated to form an overall dividing matrix $\boldsymbol{G}(z)$. It is therefore possible to approximately reconstruct $\boldsymbol{R}(z)$ from the product $\tilde{\boldsymbol{G}}(z)\boldsymbol{R}'(z)\boldsymbol{G}(z)$.

Each block on the diagonal of matrix $\boldsymbol{R}'(z)$ is then diagonalised in sequence through the use of the SMD algorithm. The diagonalised outputs, $\hat{\boldsymbol{D}}(z)$, are placed on the diagonal of matrix $\boldsymbol{D}(z)$, and the corresponding paraunitary matrices, $\hat{\boldsymbol{H}}(z)$, are stored on the diagonal of matrix $\boldsymbol{H}(z)$. Matrix $\boldsymbol{R}'(z)$ can be approximately reconstructed from $\tilde{\boldsymbol{H}}(z)\boldsymbol{D}(z)\boldsymbol{H}(z)$; by extension, it is possible to approximately reconstruct $\boldsymbol{R}(z)$ from the product $\tilde{\boldsymbol{G}}(z)\tilde{\boldsymbol{H}}(z)\boldsymbol{D}(z)\boldsymbol{H}(z)\boldsymbol{G}(z) = \tilde{\boldsymbol{F}}(z)\boldsymbol{D}(z)\boldsymbol{F}(z)$.

Algorithm 1 summarises the above steps of DC-SMD in more detail. Of the parameters input to DC-SMD, μ is a truncation parameter, and ϵ is the previously mentioned stopping threshold for SMD. Matrices of spatial dimension greater than $\hat{M} \times \hat{M}$ will be subject to DC-SMD. Parameters P, δ, I_D, and I_C will be discussed in subsequent sections. Matrices $\mathbf{I}_{M\times M}$ and $\mathbf{0}_{M\times M}$ are identity and zero matrices of spatial dimensions $M \times M$, respectively.

B. Recursive Polynomial Matrix Segmentation

When $\boldsymbol{R}(z)$ is measured to have spatial dimension $M > \hat{M}$, the divide stage of DC-SMD comes into effect. This stage

Input: $\boldsymbol{R}(z)$, P, δ, $\hat{M}$, μ, ϵ, I_D, I_C
Output: $\boldsymbol{D}(z)$, $\boldsymbol{F}(z)$
Determine if input matrix is large:
if $M > \hat{M}$ **then**
  Large matrix — divide and conquer:
  $M' = M$, $\hat{\boldsymbol{R}}(z) = \boldsymbol{R}(z)$, $\boldsymbol{G}(z) = \mathbf{I}_{M\times M}$, $\boldsymbol{R}'(z), \boldsymbol{H}(z), \boldsymbol{D}(z) = \mathbf{0}_{M\times M}$, $\alpha = 0$
  Divide matrix:
  while $M' > \hat{M}$ **do**
    $\alpha = \alpha + 1$
    $[\hat{\boldsymbol{R}}_{11}(z), \hat{\boldsymbol{R}}_{22}(z), \boldsymbol{T}(z)]$ = *SMS*($\hat{\boldsymbol{R}}(z), I_D, P, \mu, \delta$)
    $(M - M')$ *ones appended to lag-zero diagonal of* $\boldsymbol{T}(z)$ *to form* $\hat{\boldsymbol{T}}(z)$
    Store $\hat{\boldsymbol{R}}_{22}(z)$ *on diagonal of* $\boldsymbol{R}'(z)$ *in* α*th* $P \times P$ *block from bottom-right*
    $\boldsymbol{G}(z) = \hat{\boldsymbol{T}}(z)\boldsymbol{G}(z)$, $\hat{\boldsymbol{R}}(z) = \hat{\boldsymbol{R}}_{11}(z)$, $M' = M' - P$
  end
  Store $\hat{\boldsymbol{R}}(z)$ *on diagonal of* $\boldsymbol{R}'(z)$ *in top-left* $M' \times M'$ *block*
  Conquer independent matrices:
  for $\gamma = 1$ **to** $(\alpha + 1)$ **do**
    $\boldsymbol{A}(z)$ *is* γ*th block of* $\boldsymbol{R}'(z)$ *from bottom-right*
    $[\hat{\boldsymbol{H}}(z), \hat{\boldsymbol{D}}(z)]$ = *SMD*($\boldsymbol{A}(z), I_C, \epsilon, \mu$)
    Store $(\hat{\boldsymbol{D}}(z), \hat{\boldsymbol{H}}(z))$ *in* γ*th block of* $(\boldsymbol{D}(z), \boldsymbol{H}(z))$ *from bottom-right*
  end
  $\boldsymbol{F}(z) = \boldsymbol{H}(z)\boldsymbol{G}(z)$
else
  Small matrix — perform SMD only:
  $[\boldsymbol{F}(z), \boldsymbol{D}(z)]$ = *SMD*($\boldsymbol{R}(z), I_D, \epsilon, \mu$)
end

Algorithm 1: DC-SMD Algorithm

recursively applies sequential matrix segmentation (SMS) to divide $\boldsymbol{R}(z)$ into multiple independent parahermitian matrices. SMS is a novel variant of SMD designed to segment a matrix $\hat{\boldsymbol{R}}(z) \in \mathbb{C}^{M'\times M'}$ into two independent parahermitian matrices $\hat{\boldsymbol{R}}_{11}(z) \in \mathbb{C}^{(M'-P)\times(M'-P)}$ and $\hat{\boldsymbol{R}}_{22}(z) \in \mathbb{C}^{P\times P}$, and two matrices $\hat{\boldsymbol{R}}_{12}(z) \in \mathbb{C}^{(M'-P)\times P}$ and $\hat{\boldsymbol{R}}_{21}(z) \in \mathbb{C}^{P\times(M'-P)}$, where $\hat{\boldsymbol{R}}_{12}(z) = \tilde{\hat{\boldsymbol{R}}}_{21}(z)$ are approximately zero. The dimensions of the smaller matrix produced during division, P, is forced to satisfy $P \leq \hat{M}$.

The SMS algorithm is initialised and operates in a similar manner to the SMD algorithm in Sec. II-A, but with a few key differences. Instead of iteratively shifting single row-column pairs in an effort to diagonalise a parahermitian matrix $\boldsymbol{S}^{(i)}(z)$, SMS iteratively minimises the energy in select regions of $\boldsymbol{S}^{(i)}(z)$ in an attempt to segment the matrix. Fig. 2 illustrates the segmentation process for $M' = 5$ and $P = 2$.

To achieve this segmentation, the delay matrix (3) from SMD is replaced with paraunitary delay matrix

$$\boldsymbol{\Lambda}^{(i)}(z) = \mathrm{diag}\{\underbrace{1 \;\ldots\; 1}_{M'-P}\; \underbrace{z^{-\tau^{(i)}} \;\ldots\; z^{-\tau^{(i)}}}_{P}\} \tag{13}$$

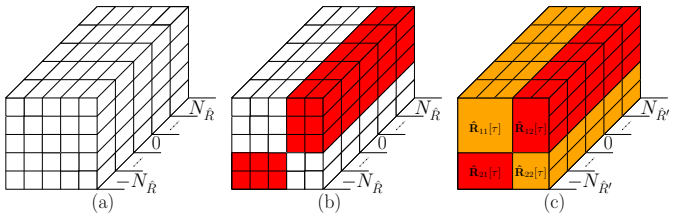

Fig. 2. (a) Original matrix $\hat{\mathbf{R}}[\tau] \in \mathbb{C}^{5\times 5}$, (b) regions (red) to be iteratively driven to zero in SMS for $P = 2$, and (c) segmented result. $N_{\hat{R}}$ and $N_{\hat{R}'}$ are the maximum lags for the original and segmented matrices, respectively.

Input: $\hat{\boldsymbol{R}}(z)$, I_D, P, μ, δ
Output: $\hat{\boldsymbol{R}}_{11}(z)$, $\hat{\boldsymbol{R}}_{22}(z)$, $\boldsymbol{T}(z)$
Find eigenvectors $\mathbf{Q}^{(0)}$ that diagonalise $\hat{\mathbf{R}}[0] \in \mathbb{C}^{M'\times M'}$
$\boldsymbol{S}^{(0)}(z) = \mathbf{Q}^{(0)}\hat{\boldsymbol{R}}(z)\mathbf{Q}^{(0)\mathrm{H}}$, $\boldsymbol{H}^{(0)}(z) = \mathbf{Q}^{(0)}$, $i = 0$, stop = 0
do
  $i = i + 1$
  Find $\tau^{(i)}$ from (14); generate $\boldsymbol{\Lambda}^{(i)}(z)$ from (13)
  $\boldsymbol{S}^{(i)\prime}(z) = \boldsymbol{\Lambda}^{(i)}(z)\boldsymbol{S}^{(i-1)}(z)\tilde{\boldsymbol{\Lambda}}^{(i)}(z)$
  Find eigenvectors $\mathbf{Q}^{(i)}$ that diagonalise $\mathbf{S}^{(i)\prime}[0]$
  $\boldsymbol{S}^{(i)}(z) = \mathbf{Q}^{(i)}\boldsymbol{S}^{(i)\prime}(z)\mathbf{Q}^{(i)\mathrm{H}}$
  $\boldsymbol{H}^{(i)}(z) = \mathbf{Q}^{(i)}\boldsymbol{H}^{(i)\prime}(z) = \mathbf{Q}^{(i)}\boldsymbol{\Lambda}^{(i)}(z)\boldsymbol{H}^{(i-1)}(z)$
  Truncate $\boldsymbol{H}^{(i)}(z)$ according to (11)
  Truncate $\boldsymbol{S}^{(i)}(z)$ according to (12)
  if $i > I_D$ or (16) satisfied **then**
    stop = 1;
  end
while stop = 0
$\boldsymbol{T}(z) = \boldsymbol{H}^{(i)}(z)$
$\hat{\boldsymbol{R}}_{11}(z)$ is top-left $(M' - P) \times (M' - P)$ block of $\boldsymbol{S}^{(i)}(z)$
$\hat{\boldsymbol{R}}_{22}(z)$ is bottom-right $P \times P$ block of $\boldsymbol{S}^{(i)}(z)$

Algorithm 2: SMS algorithm

at the ith iteration of SMS, where

$$\tau^{(i)} = \arg\max_{\tau} \|\mathbf{S}_{21}^{(i-1)}[\tau]\|_{\mathrm{F}} \tag{14}$$

and

$$\|\mathbf{S}_{21}^{(i-1)}[\tau]\|_{\mathrm{F}} = \sqrt{\textstyle\sum_{m=M'-P+1}^{M'} \sum_{k=1}^{M'-P} |\mathbf{S}_{m,k}^{(i-1)}[\tau]|^2} \quad . \tag{15}$$

Where $\mathbf{S}_{m,k}^{(i-1)}[\tau]$ represents the element in the mth row and kth column of the coefficient matrix $\mathbf{S}^{(i-1)}[\tau]$ at lag τ. Equations (4) and (5) are similarly implemented in SMS, where unitary matrix $\mathbf{Q}^{(i)}$ again diagonalises $\mathbf{S}^{(i)\prime}[0]$.

After I_D iterations, or when matrix $\boldsymbol{S}_{21}^{(I)}(z)$ contains energy below $\delta \sum_\tau \|\mathbf{S}^{(I)}[\tau]\|_{\mathrm{F}}^2$ at some iteration I; i.e.,

$$\textstyle\sum_\tau \|\mathbf{S}_{21}^{(I)}[\tau]\|_{\mathrm{F}} < \delta \sum_\tau \|\mathbf{S}^{(I)}[\tau]\|_{\mathrm{F}}^2 \quad , \tag{16}$$

the SMS algorithm returns matrices $\hat{\boldsymbol{R}}_{11}(z)$, $\hat{\boldsymbol{R}}_{22}(z)$, and $\boldsymbol{T}(z)$. The latter is constructed from the concatenation of the elementary paraunitary matrices as in (10). A parameter μ is used to truncate the paraunitary and parahermitian matrices at each iteration as described in (11), (12).

The above steps of SMS are summarised in Algorithm 2.

C. Independent Conquering of Divided Polynomial Matrices

At this stage of DC-SMD, $\boldsymbol{R}(z) \in \mathbb{C}^{M\times M}$ has been segmented into multiple independent parahermitian matrices,

which are stored as blocks on the diagonal of $\boldsymbol{R}'(z)$. Each matrix can now be diagonalised individually through the use of a PEVD algorithm; here, the SMD algorithm is chosen. Each instance of SMD is provided with a parameter I_C — which defines the maximum possible number of algorithm iterations — a stopping threshold ϵ, and a truncation parameter μ. Upon completion, the SMD algorithm returns matrices $\hat{\boldsymbol{H}}(z)$ and $\hat{\boldsymbol{D}}(z)$, which contain the polynomial eigenvectors and eigenvalues for input matrix $\boldsymbol{A}(z)$, respectively. At iteration γ of this stage, $\boldsymbol{A}(z)$ contains the γth block of $\boldsymbol{R}'(z)$ from the bottom-right.

D. Algorithm Complexity

The instantaneous complexity of DC-SMD varies as the algorithm progresses, due to the changing spatial dimensions of the matrices being processed. The main cost of the SMS and SMD algorithms internal to DC-SMD is a matrix multiplication step; therefore, the calculation of the cumulative complexities of DC-SMD is similar to Sec. II-B.

In DC-SMD, one instance of the SMS algorithm has a maximum cumulative complexity of $M_\alpha^3 \sum_{i=0}^{I_D} (2L\{\boldsymbol{S}^{(i)\prime}\} + L\{\boldsymbol{H}^{(i)\prime}\})$, and SMD has a similar maximum of $M_\gamma^3 \sum_{i=0}^{I_C} (2L\{\boldsymbol{S}^{(i)\prime}\} + L\{\boldsymbol{H}^{(i)\prime}\})$, where M_α and M_γ are the dimensions of the matrices input to each algorithm, respectively. Function $L\{\cdot\}$ computes the length of the parahermitian and paraunitary matrix in each algorithm at iteration i. The total cumulative complexity of DC-SMD can be approximated by summing the cumulative complexities of each instance of the SMS and SMD algorithms.

From the description of DC-SMD in Algorithm 1, it can be seen that an $M \times M$ matrix is only ever processed in the first recursion of the division step; at all other points in the algorithm, the processed matrices are of lower spatial dimension. Given that the complexity is proportional to the cube of the spatial dimension, significantly lower complexity will be observed beyond the first recursion of DC-SMD.

IV. Results

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

A. Performance Metrics

Since SMD and DC-SMD iteratively minimise off-diagonal energy, a suitable metric $E_{\text{norm}}^{(i)}$, defined in [9], is used; this metric divides the off-diagonal energy in the parahermitian matrix at the ith iteration by the total energy. Computation of $E_{\text{norm}}^{(i)}$ generates squared covariance terms; therefore a logarithmic notation of $5 \log_{10} E_{\text{norm}}^{(i)}$ is employed.

When truncation is employed, the eigenvectors and eigenvalues output from SMD are only able to approximately reconstruct the input matrix. DC-SMD experiences similar error from truncation, and also introduces further error in its divide step, due to imperfect segmentation in SMS. A metric capable of measuring the difference between the original and reconstructed matrices is the mean squared error:

$$\text{MSE} = \tfrac{1}{M^2 L\{\boldsymbol{E}_R\}} \textstyle\sum_\tau \|\mathbf{E}_R[\tau]\|_{\text{F}}^2 \,, \tag{17}$$

where $\mathbf{E}_R[\tau] = \bar{\mathbf{R}}[\tau] - \mathbf{R}[\tau] \;\forall\; \tau$, $\bar{\boldsymbol{R}}(z) = \tilde{\boldsymbol{F}}(z)\boldsymbol{D}(z)\boldsymbol{F}(z)$, and $\boldsymbol{F}(z)$ and $\boldsymbol{D}(z)$ are obtained from SMD or DC-SMD.

The contents of Sec. II-B and Sec. III-D allow approximate measurements of cumulative complexity to be made at each iteration of both algorithms.

The output paraunitary matrix $\boldsymbol{F}(z)$ can be used in signal processing applications. A useful metric for gauging the implementation cost of this matrix is its length.

B. Simulation Scenario

The simulations below have been performed over an ensemble of 10^3 instantiations of $\boldsymbol{R}(z) \in \mathbb{C}^{M\times M}$, $M \in \{20;\ 40\}$, based on the randomised source model in [9]. This source model generates $\boldsymbol{R}(z) = \tilde{\boldsymbol{U}}(z)\boldsymbol{W}(z)\boldsymbol{U}(z)$, whereby the diagonal $\boldsymbol{W}(z) \in \mathbb{C}^{M\times M}$ contains the power spectral densities (PSDs) of 10 independent sources. These sources are spectrally shaped by innovation filters such that $\boldsymbol{W}(z)$ has an order of 120, and limits 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 an order of 240.

During iterations, a truncation parameter of $\mu = 10^{-6}$ and stopping thresholds of $\epsilon = 10^{-6}$ and $\delta = 10^{-3}$ were used. The standard SMD implementation was run over $I = 800$ iterations for $M = 20$, and $I = 400$ iterations for $M = 40$. DC-SMD was executed with input parameters $I_D = 100$, $I_C = 200$, $P = 8$, and $\hat{M} = 8$. At every iteration step of both algorithms, the diagonalisation and cumulative complexity metrics defined in Sec. IV-A were recorded together with the elapsed execution time. The MSE metric defined in (17) and the length of $\boldsymbol{F}(z)$ were recorded upon each algorithm's completion.

C. Diagonalisation

The ensemble-averaged diagonalisation was calculated for both the standard and proposed implementations. The diagonalisation performance versus time and cumulative complexity for both methods are shown in Figs. 3 and 4, respectively. The curves of Fig. 3 demonstrate that for $M \in \{20;\ 40\}$, the proposed implementation operates with a lower cumulative complexity than the standard SMD realisation, and is able to achieve a similar degree of diagonalisation. In addition, Fig. 4 shows that the lower complexity associated with DC-SMD translates to a faster diagonalisation than observed for SMD. Using a matrix with a larger spatial dimension of $M = 40$ demonstrates a larger increase in diagonalisation performance with respect to execution time. In both plots, $\mathcal{E}\{\cdot\}$ is the expectation operator.

The 'stepped' characteristics of the curves for DC-SMD are a result of the algorithm's recursive two-stage implementation. The divide step of the algorithm exhibits low diagonalisation for a large increase in cumulative complexity and execution time. In the conquer step, high diagonalisation is seen for a small increase in cumulative complexity and execution time.

D. Reconstruction Error

The ensemble-averaged mean squared reconstruction error was calculated for both algorithms, according to (17). Tab. I shows the results for $M \in \{20;\ 40\}$; from this, it is clear

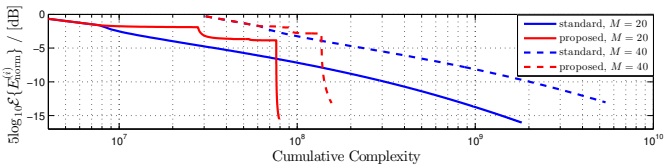

Fig. 3. Diagonalisation metric vs. cumulative algorithm complexity for the proposed and standard implementations for $M \in \{20;\ 40\}$.

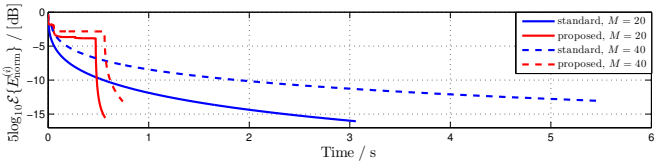

Fig. 4. Diagonalisation metric vs. algorithm execution time for the proposed and standard implementations for $M \in \{20;\ 40\}$.

TABLE I. AVERAGE MSE AND PU FILTER LENGTH COMPARISON.

Method	MSE		PU Filter Length	
	$M = 20$	$M = 40$	$M = 20$	$M = 40$
standard	1.991×10^{-6}	5.643×10^{-7}	116.8	79.13
proposed	7.991×10^{-6}	3.477×10^{-6}	154.3	121.8

that the increased diagonalisation speed and lower cumulative complexity of DC-SMD comes with the cost of higher reconstruction error. To reduce this error, parameter δ can be decreased; however, this will reduce the speed and increase the complexity of the algorithm, as more effort will be contributed to the divide step. Note that the relative difference in average MSE is larger for the case where $M = 40$, which suggests that the algorithm's much improved diagonalisation performance for $M = 40$ is not without cost.

E. Paraunitary Filter Length

The ensemble-averaged paraunitary (PU) filter lengths were calculated for both algorithms. Tab. I shows the results for $M \in \{20;\ 40\}$. It can be seen from this table that the average paraunitary filter length is larger for DC-SMD than SMD; this is disadvantageous for application purposes. The relative difference in average paraunitary filter length is larger for the case where $M = 40$, which validates the previous observation that the algorithm's increased diagonalisation performance for $M = 40$ brings more substantial disadvantages.

V. CONCLUSION

We have proposed an alternative technique to compute the polynomial EVD of a parahermitian matrix; this algorithm — named DC-SMD — makes use of a divide-and-conquer approach to the PEVD, and has been shown to operate with lower computational complexity than the traditional SMD algorithm. Simulation results have demonstrated that this complexity reduction, and the associated execution time decrease, come with the disadvantage of increasing the mean squared reconstruction error and the paraunitary filter order.

When designing PEVD implementations for real applications, the potential for the proposed techniques to increase diagonalisation performance while reducing complexity requirements offers benefits. A further advantage of the DC-SMD algorithm is its ability to produce multiple independent parahermitian matrices, which may be processed in parallel. Simulation results demonstrate that DC-SMD outperforms SMD more significantly for larger values of M; therefore, DC-SMD is suitable for broadband multichannel applications with a large number of sensors.

ACKNOWLEDGEMENT

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

This work was supported in parts by the Engineering and Physical Sciences Research Council (EPSRC) Grant number EP/K014307/1 and the MOD University Defence Research Collaboration in Signal Processing.

REFERENCES

[1] 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, USA, Nov. 2007.

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

[3] 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 ISP*, London, UK, Dec. 2015.

[4] 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.

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

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

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

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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, Sep. 2015.

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

[15] C. H. Ta and S. Weiss. Shortening the order of paraunitary matrices in SBR2 algorithm. In *ICICSP*, pp. 1–5, Singapore, Dec. 2007.

[16] F. Coutts, J. Corr, J. Thompson, S. Weiss, J. Proudler, and J. McWhirter. Memory and Complexity Reduction in Parahermitian Matrix Manipulations of PEVD Algorithms. In *EUSIPCO*, pp. 1633-1637, Budapest, Hungary, August 2016.

[17] F. Coutts, J. Corr, J. Thompson, S. Weiss, I. Proudler, and J. McWhirter. Complexity and Search Space Reduction in Cyclic-by-Row PEVD Algorithms. In *Asilomar SSC*, Pacific Grove, CA, Nov. 2016.

[18] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Num. Mathematik*, 36(2):177–195, June 1980.

[19] J. J. Dongarra and D. C. Sorensen. A fully parallel algorithm for the symmetric eigenvalue problem. *SIAM JSSC*, 8(2):139–154, March 1987.

[20] D. Gill and E. Tadmor. An $O(N^2)$ method for computing the eigensystem of $N \times N$ symmetric tridiagonal matrices by the divide and conquer approach. *SIAM JSSC*, 11(1):161–173, Jan. 1990.