Signal Processing 240 (2026) 110340

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Polynomial matrix SVD via generalized sequential matrix diagonalization

Faizan A. Khattak [a,*], Soydan Redif [b], Mohammed Bakhit [c]

[a] Department of Computer Science, University of Leeds, Leeds LS2 9JT, England, United Kingdom
[b] College of Engineering and Technology, American University of the Middle East, Kuwait
[c] Department of Electronic & Electrical Engineering, University of Strathclyde, Glasgow G1 1XW, Scotland, United Kingdom

ARTICLE INFO

Keywords:
Polynomial matrix
Polynomial singular value decomposition
Broadband MIMO
Sequential matrix diagonalization
MIMO channel equalization

ABSTRACT

The singular value decomposition (SVD) of polynomial matrices serves as a cornerstone in the analysis and optimization of broadband multi-input multi-output (MIMO) systems. This paper introduces novel algorithms for performing the SVD of polynomial matrices, leveraging a sequential matrix diagonalization (SMD) framework. The proposed methodology begins by identifying the column or row with the highest off-diagonal energy using a maximum search procedure. Subsequently, this energy is transferred to the zero-lag coefficient matrix through a delay operation, which is then diagonalized using a conventional SVD. This iterative process continues until the maximum off-diagonal element falls below a predefined threshold. The proposed framework encompasses multiple algorithmic variants, each designed to offer distinct convergence speeds, thereby addressing diverse computational and accuracy requirements. Rigorous proofs of convergence are provided, alongside a thorough comparative analysis of the computational efficiency and diagonalization accuracy of the algorithms. Extensive simulations, conducted on ensembles of randomly generated polynomial matrices, demonstrate that the proposed algorithms consistently outperform state-of-the-art polynomial SVD (PSVD) methods across all evaluated performance metrics. Furthermore, the application of the proposed algorithm to decouple broadband or convolutive MIMO channels validates its accuracy and effectiveness in practical scenarios.

1. Introduction

Extending matrix algebra to polynomial matrices is highly motivated by applications across control systems, digital signal processing, and broadband communications, notably in MIMO system design and broadband source separation [1–7]. Recent advancements have enabled extensions of fundamental decompositions such as eigenvalue decomposition (EVD) [8–12], QR decomposition [13–15], and singular value decomposition (SVD) [13,16–18], along with methods like the power method [19], into the polynomial domain. These polynomial matrix factorizations – including polynomial EVD (PEVD), polynomial QR decomposition (PQRD), and polynomial SVD (PSVD) – have enhanced both the accuracy and efficiency of broadband signal processing applications [2,8–19].

Unlike the polynomial EVD (pEVD), which has seen broad application and algorithmic advancement, the polynomial SVD (PSVD) has received comparatively less attention, primarily because it can be computed using two pEVDs. The first dedicated PSVD algorithm utilized the Kogbetliantz transformation [20], extending the pEVD approach known as second-order sequential best rotation (SBR2) [8] to ordinary polynomial matrices [16]. Additionally, the extension of QR decomposition to polynomial matrices in [13] offered an iterative approach for computing the PSVD, similar to the QR algorithm for standard matrices [21]. However, this method requires repeated application of the polynomial QR decomposition (pQRD), and with the truncation steps necessary to limit polynomial order growth, it incurs error accumulation, reduced decomposition accuracy, and significant computational cost. Due to the high polynomial order, hardware-implementation costs increase in broadband MIMO applications and, presumably, in other applications. To address this issue, [18] have utilized a phase smoothing procedure originally proposed for extracting analytic eigenvectors of a para-Hermitian matrix [11]. This method operates in the discrete Fourier transform (DFT) domain. While it has been shown to converge, it is only suitable for polynomial matrices with small spatial and temporal dimensions.

Recently, it was proven that an analytic SVD exists for polynomial matrices, allowing singular values to be real-valued on the unit circle, but they can also take negative values, unlike in ordinary matrices [22]. A novel singular value extraction algorithm was proposed in [23], offering high scalability in spatial and temporal dimensions. However, this approach has limitations. It lacks a balance between execution

* Corresponding author.
E-mail addresses: f.a.khattak@leeds.ac.uk (F.A. Khattak), soydan.redif@aum.edu.kw (S. Redif), mohammed.bakhit@strath.ac.uk (M. Bakhit).

https://doi.org/10.1016/j.sigpro.2025.110340
Received 9 January 2025; Received in revised form 10 September 2025; Accepted 13 October 2025
Available online 16 October 2025

time and diagonalization precision, and struggles with estimated polynomial matrices, particularly when polynomial singular values are of high order, leading to longer runtimes and difficulty in extracting approximate values. Additionally, it focuses solely on singular value extraction without providing a scalable method for singular vectors. Since estimated polynomial matrices exhibit spectrally majorized singular values that are strictly non-negative on the unit circle [24,25], therefore, the method in [23], which permutes singular values in each DFT bin to establish smooth association, is of little avail in cases where a polynomial matrix is estimated from a finite set of samples.

In this paper, we address the challenges associated with extending matrix decompositions to polynomial matrices. Specifically, we introduce novel PSVD algorithms based on the sequential matrix diagonalization (SMD) approach in [9], enhancing both computational efficiency and accuracy. Building on our previous work in [26], we propose generalized SMD (GSMD) algorithms for computing the PSVD of polynomial matrices. The proposed approach transfers the maximum off-diagonal elements from the non-zero slice of a polynomial matrix to the zero-lag coefficient matrix using paraunitary delay operations [27], followed by performing the SVD on the zero-lag matrix to diagonalize it. This process is repeated iteratively until the off-diagonal elements fall below a predefined threshold. Multiple algorithmic variants are introduced, each with its own advantages. We also provide a proof of convergence, demonstrating that the proposed algorithms consistently achieve the desired results.

The remainder of this paper is organized as follows: Sections II and III provide an overview of the existence of analytic SVD and the corresponding algorithmic solutions. Section 4 details the framework of the proposed algorithm, while Section 5 enumerates the potential variants within this framework. Section 6 presents simulations and discusses the results, followed by Section 7, which illustrates an application scenario of the proposed algorithm. Finally, the conclusions are summarized in Section VIII.

2. Notations and definitions

A polynomial matrix in the complex variable z with spatial dimension $M \times N$ can be represented as

$$\boldsymbol{A}(z) = \begin{bmatrix} a_{1,1}(z) & \dots & a_{1,N}(z) \\ \vdots & \ddots & \vdots \\ a_{M,1}(z) & \dots & a_{M,N}(z) \end{bmatrix} = \sum_{\tau=-\infty}^{\infty} \mathbf{A}[\tau] z^{-\tau},$$

where each element is a Laurent polynomial in the variable z, allowing both positive and negative powers [28]. The polynomial matrix and vector are denoted by bold italic upper and lower case letters with the continuous variable written in parentheses, e.g., $\boldsymbol{A}(z)$ and $\boldsymbol{a}(z)$. A polynomial element in the mth row and nth column is denoted by $a_{m,n}(z)$. The ith row of the matrix $\boldsymbol{A}(z)$ is denoted by $\boldsymbol{a}_{i,:}(z)$ and the ith column by $\boldsymbol{a}_{:,i}(z)$. The discrete time-domain quantity of a polynomial matrix $\boldsymbol{A}(z)$ is denoted by $\mathbf{A}[\tau] \bullet\!\!-\!\!\circ\ \boldsymbol{A}(z)$, where $\tau \in \mathbb{Z}$, with the transform pair denoted by $\bullet\!\!-\!\!\circ$. Note that matrices or vectors that depend on the discrete variable are denoted with upright bold upper and lower case letters with the discrete variable inside square brackets.

Akin to the Hermitian operator, denoted by $\{\cdot\}^{\mathrm{H}}$ in ordinary matrices, $\boldsymbol{A}^{\mathrm{P}}(z)$ is the para Hermitian [28] of $\boldsymbol{A}(z)$, obtained by taking the Hermitian transpose and then reversing time, that is by replacing z with z^{-1}, hence $\boldsymbol{A}^{\mathrm{P}}(z) = \boldsymbol{A}^{\mathrm{H}}(z^{-1}) \bullet\!\!-\!\!\circ\ \mathbf{A}^{\mathrm{H}}[-\tau]$. Similar to a unitary matrix, a polynomial matrix $\boldsymbol{U}(z) \in \mathbb{C}^{M\times M}$ is paraunitary if $\boldsymbol{U}^{\mathrm{P}}(z)\boldsymbol{U}(z) = \boldsymbol{U}(z)\boldsymbol{U}^{\mathrm{P}}(z) = \mathbf{I}_M$, where $\mathbf{I}_M$ is an $M \times M$ identity matrix. In the context of this paper, we define $\mathbf{A}[0]$ as the zero-lag coefficient matrix of the polynomial matrix $\boldsymbol{A}(z)$. In addition, the coefficient matrix at $\tau = \tau_0$, i.e., $\mathbf{A}[\tau]\big|_{\tau=\tau_0}$, is referred to as the τ_0th lag coefficient matrix. A quantity with an exponent is denoted by $\boldsymbol{A}^i(z)$, while its ith update in an algorithm is denoted by parentheses in the superscript, e.g., $\boldsymbol{A}^{(i)}(z)$.

3. Background

3.1. Analytic SVD existence

Matrices that are functions of a real and continuous variable t are commonly encountered in control engineering. To address problems such as stability analysis, robust control design, and H_∞ control [29, 30], an analytic SVD for an analytic $\boldsymbol{A}(t)$, i.e.,

$$\boldsymbol{A}(t) = \boldsymbol{U}(t)\boldsymbol{\Sigma}(t)\boldsymbol{V}^{\mathrm{H}}(t),$$

has been shown to exist over a real interval $t_1 < t < t_2$, where both the left- and right-singular vectors admit analytic properties [31]. However, for the singular values – the diagonal elements of $\boldsymbol{\Sigma}(t)$ – to admit analyticity, they are allowed to take non-negative values. For such applications, various algorithms have been proposed to compute the analytic SVD for these matrices [32,33].

However, often in the signal processing domain, matrices that are functions of discrete variables, such as $\mathbf{A}[\tau]$ with $\tau \in \mathbb{Z}$, frequently arise in filter design [28] and other applications. Its z-transform $\boldsymbol{A}(z)$, being analytic in the continuous complex variable z, has therefore also received significant attention. In this work, we focus on such complex-variable analytic polynomial matrices. For matrices $\boldsymbol{A}(z) \in \mathbb{C}^{M\times N}$ with $M \geq N$, the existence of an analytic SVD – where the singular values are real-valued on the unit circle – depends on the number of spectral nulls of the singular values on the unit circle with odd multiplicity [22]. Mathematically, it can be represented as

$$\boldsymbol{A}(z^q) = \boldsymbol{U}(z)\boldsymbol{\Sigma}(z)\boldsymbol{V}^{\mathrm{P}}(z) \tag{1}$$

where $\{\cdot\}^{\mathrm{P}}$ denotes a para-Hermitian operator, which is equivalent to taking the Hermitian conjugate followed by time-reversal, i.e., $\boldsymbol{V}^{\mathrm{P}}(z) = \boldsymbol{V}^{\mathrm{H}}(1/z^*)$. Here, $\boldsymbol{U}(z) \in \mathbb{C}^{M\times M}$ and $\boldsymbol{V}(z) \in \mathbb{C}^{N\times N}$ contain left and right singular vectors, respectively, and $\boldsymbol{\Sigma}(z) \in \mathbb{C}^{M\times N}$ contains the analytic singular values on its diagonal. These singular values are real-valued on the unit circle but may also take on negative values. If the number of spectral nulls is odd, q is 2; otherwise, $q = 1$. However, if the singular values are not constrained to be real-valued on the unit circle, the SVD can still be computed without upsampling by a factor of 2 (i.e., $q = 2$) [22], even when the singular values have an odd number of spectral nulls on the unit circle.

Generally, the factors in (1) are of transcendental functions, but due to the property of analyticity, they can be approximated sufficiently well by Laurent polynomials [22,34]. The left and right singular vectors are ambiguous up to an all-pass factor; that is, given an all-pass function $\phi_m(z)$, $\boldsymbol{u}_m(z)\phi_m(z)$ remains a valid mth left singular vector, and similarly for $\boldsymbol{v}(z)$.

A polynomial matrix $\boldsymbol{A}(z) \in \mathbb{C}^{M\times N}$, estimated from finite number of sensor measurements, will manifest spectrally majorised singular values, if evaluated on the unit circle [24,25], i.e.,

$$\sigma_1(\mathrm{e}^{\mathrm{j}\Omega}) > \sigma_2(\mathrm{e}^{\mathrm{j}\Omega}) > \cdots > \sigma_M(\mathrm{e}^{\mathrm{j}\Omega})\ \forall\Omega \tag{2}$$

even if the ground truth singular values are overlapping, i.e., not spectrally majorised, on the unit circle. This effect due to estimation error is beneficial for time-domain algorithms which tend to converge to spectral majorised singular values.

3.2. PSVD computing algorithms

In this section, we provide a brief overview of existing approaches of computing a PSVD of a polynomial matrix.

3.2.1. Computation via two PEVDs

A PSVD of a polynomial matrix can be computed by employing two PEVD operations

$$\boldsymbol{R}_1(z) = \boldsymbol{A}(z)\boldsymbol{A}^{\mathrm{P}}(z) = \boldsymbol{H}_1(z)\boldsymbol{\Lambda}_1(z)\boldsymbol{H}_1^{\mathrm{P}}(z),$$
$$\boldsymbol{R}_2(z) = \boldsymbol{A}^{\mathrm{P}}(z)\boldsymbol{A}(z) = \boldsymbol{H}_2(z)\boldsymbol{\Lambda}_2(z)\boldsymbol{H}_2^{\mathrm{P}}(z),$$

where $\boldsymbol{H}_1(z)$ and $\boldsymbol{H}_2(z)$ are paraunitary matrices [34] that contain the eigenvectors of $\boldsymbol{R}_1(z)$ and $\boldsymbol{R}_2(z)$, respectively. These eigenvector matrices are related to $\boldsymbol{U}(z)$ and $\boldsymbol{V}(z)$ via diagonal matrices of allpass filters, that is, $\boldsymbol{H}_1(z) = \boldsymbol{U}(z)\boldsymbol{\Phi}_1(z)$ and $\boldsymbol{H}_2(z) = \boldsymbol{V}(z)\boldsymbol{\Phi}_2(z)$, where $\boldsymbol{\Phi}_1(z) = \mathrm{diag}\{\phi_{11}(z), \ldots, \phi_{1M}(z)\}$ and $\boldsymbol{\Phi}_2(z) = \mathrm{diag}\{\phi_{21}(z), \ldots, \phi_{2N}(z)\}$, and where $\phi_{1m}(z)$ for $m = 1, \ldots, M$ and $\phi_{2n}(z)$ for $n = 1, \ldots, N$ are allpass filters [22]. With the help of these singular vectors, one can compute the singular values via (1); however, they may not be real valued on the unit circle due to the aforementioned allpass ambiguity unless $\boldsymbol{\Phi}_1(z) = \boldsymbol{\Phi}_2(z)$. The PEVD is often computed using SBR2 or SMD algorithms [8,17]. This method of PSVD computation is computationally expensive as it computes the PEVD for two parahermitian matrices, i.e., $\boldsymbol{R}_1(z), \boldsymbol{R}_2(z)$, each with a polynomial order twice that of $\boldsymbol{A}(z)$.

3.2.2. Computation via repeated polynomial QR decompositions

This method for computing the PSVD of a polynomial matrix involves multiple PQRDs of polynomial matrices. It begins by computing the PQRD of $\boldsymbol{A}(z)$, where $\boldsymbol{A}(z)$ is factorized into $\boldsymbol{Q}_1(z)\boldsymbol{R}_1(z)$, with $\boldsymbol{R}_1(z)$ being an upper triangular matrix. Next, the PQRD of $\boldsymbol{R}_1^{\mathrm{P}}(z)$ is computed as $\boldsymbol{Q}_2(z)\boldsymbol{R}_2(z)$. With these two PQRDs, one iteration of the process is completed. The procedure continues by replacing $\boldsymbol{A}(z)$ with $\boldsymbol{A}_1(z)$, defined by $\boldsymbol{Q}_1^{\mathrm{P}}(z)\boldsymbol{A}(z)\boldsymbol{Q}_2(z) = \boldsymbol{R}_2^{\mathrm{P}}(z)$. This process is repeated until $\boldsymbol{R}_2(z)$ is a diagonal matrix, which means that its off-diagonal elements fall below a predefined small threshold [13]. The PQRDs are computed using various algorithmic variants [15,35]. Although this approach is computationally very expensive and also produces singular vectors with high polynomial order, it has also been reported to offer improved stability and superior performance compared to PEVD-based approaches [3].

3.2.3. Computation via kogbetliantz transformation based generalized SBR2 (GSBR2) algorithm

This PSVD algorithm [16] is a generalization of the SBR2 algorithm, extending its application from parahermitian to general matrices, and is thus termed the generalized SBR2 (GSBR2) in this paper. It employs either Givens rotations or the complex Kogbetliantz transformation [20], an extension of the Jacobi transformation to nonsymmetric matrices. Both transformations are used to transfer the off-diagonal energy onto the diagonal depending on the location of the maximum off-diagonal element. When the maximum off-diagonal element lies outside the upper $N \times N$ submatrix, a Givens rotation is applied from the left to eliminate it. If the maximum off-diagonal element is within the upper $N \times N$ submatrix, a Kogbetliantz transformation is then applied which is a combination of Givens rotation, symmetrization, and the Jacobi transformation. Although this is a direct method for diagonalization, it has the drawbacks of slow convergence as it transfers a single element energy onto the diagonal in each iteration, and incurs higher computational cost compared to the SBR2 algorithm, primarily due to the complexity of the Kogbetliantz transformation [20]. Furthermore, the multiple shift strategy, applicable in the case of SBR2 [36], seems incompatible with the Kogbetliantz transformation due to its complex nature of combining Givens rotation, symmetrization, and Jacobi transformation [21].

3.2.4. DFT-domain compact order PSVD

This approach computes a DFT of the polynomial matrix and then computes the ordinary matrix SVDs at these sample points. The singular vectors in these sample points are not phase-coherent with those of adjacent samples. To address this, a phase smoothing procedure [11] is applied across all samples to establish phase coherence and thereby obtain compact-order polynomial singular vectors [12,18]. If the phase-smoothing procedure does not converge, the DFT size is increased, and the process is repeated until convergence is achieved. Although effective, this approach is computationally very expensive, because the phase smoothing, which is proven to be an NP-hard problem [11], is repeated at increasingly larger DFT sizes. As a result, it is only practical for polynomial matrices with small spatial dimensions and relatively low polynomial orders.

4. The GSMD algorithm

This section outlines the initialization and iterative steps of the proposed algorithm, assuming without loss of generality that $M \geq N$. If $M < N$, diagonalization can be performed on $\boldsymbol{A}^{\mathrm{P}}(z)$.

4.1. Initial iteration

The proposed approach begins by computing the conventional SVD of the zero-lag coefficient matrix of the given polynomial matrix $\boldsymbol{A}(z)$, i.e., $\mathbf{A}[0]$, as

$$\mathbf{A}[0] = \mathbf{U}^{(0)}\mathbf{S}^{(0)}[0]\mathbf{V}^{(0),\mathrm{H}}, \tag{3}$$

where $\mathbf{U}^{(0)} \in \mathbb{C}^{M\times M}$, $\mathbf{V}^{(0)} \in \mathbb{C}^{N\times N}$ contains the left and right singular vectors, respectively, and $\mathbf{S}^{(0)} \in \mathbb{C}^{M\times N}$ is diagonal matrix containing singular values. We multiply $\boldsymbol{A}(z)$ from the left with $\mathbf{U}^{(0),\mathrm{H}}$ and from the right with $\mathbf{V}^{(0)}$ as

$$\boldsymbol{S}^{(0)}(z) = \mathbf{U}^{(0),\mathrm{H}}\boldsymbol{A}(z)\mathbf{V}^{(0)}. \tag{4}$$

This ensures that the zero-lag coefficient matrix $\mathbf{S}^{(0)}[0]$ of $\boldsymbol{S}^{(0)}(z) \bullet\!\!-\!\!\circ\, \mathbf{S}^{(0)}[\tau]$ is a real-valued diagonal matrix. This concludes the first iteration, and the intermediate left and right polynomial singular vectors matrices are updated at the end of first iteration as $\boldsymbol{U}^{(0)}(z) = \mathbf{U}^{(0)}$ and $\boldsymbol{V}^{(0)}(z) = \mathbf{V}^{(0)}$.

4.2. Iterative procedure

In subsequent iterations $i = 1, 2, \ldots$, the algorithm first performs a maximum search over all time lags of $\mathbf{S}^{(i-1)}[\tau]$ to find a column of $\mathbf{S}^{(i-1)}[\tau]$ carrying the maximum off-diagonal energy (or equivalently the off-diagonal ℓ_2-norm). The index of the maximum column and the time lag where it resides is determined as

$$\{\tau_i, n_i\} = \underset{\tau,n}{\mathrm{argmax}} \left\{ \sum_{m=1,m\neq n}^{M} |s_{m,n}^{(i-1)}[\tau]|^2 \right\}^{\frac{1}{2}}, \tag{5}$$

where

$$\gamma_{c(n_i,\tau_i)} = \left\{ \sum_{m=1,m\neq n_i}^{M} |s_{m,n_i}^{(i-1)}[\tau_i]|^2 \right\}^{\frac{1}{2}}, \tag{6}$$

shows the off-diagonal energy (or the off-diagonal ℓ_2-norm) of the n_ith column at $\tau = \tau_i$ with $m \neq n$ reflecting that diagonal elements are not considered. This n_ith column present at $\tau = \tau_i$ is to be time-shifted to the zero-lag, i.e., $\tau = 0$, such that the diagonal elements of the zero-lag coefficient matrix of $\mathbf{S}^{(i-1)}[\tau]$ remains unchanged. This is accomplished by multiplying $\boldsymbol{S}^{(i-1)}(z)$ from the right by

$$\boldsymbol{B}_{\mathrm{r}}^{(i)}(z) = \mathrm{diag}\left\{ \mathbf{1}_{n_i-1}, z^{\tau_i}, \mathbf{1}_{N-n_i-1} \right\}, \tag{7}$$

where $\mathbf{1}_N$ denotes a vector of N ones, resulting in

$$\boldsymbol{S}'^{(i-1)}(z) = \boldsymbol{S}^{(i-1)}(z)\boldsymbol{B}_{\mathrm{r}}^{(i)}(z). \tag{8}$$

It applies a delay of τ_i to the n_ith column and time-shifts it from $\tau = \tau_i$ to $\tau = 0$. Due to $M \geq N$, the n_ith column shift has shifted the (n_i, n_i) diagonal element of the zero-lag coefficient matrix to $\tau = -\tau_i$. To bring it back to the $\tau = 0$, the n_ith row is time-shifted in opposite direction,

i.e., from $\tau = -\tau_i$ to $\tau = 0$. This is accomplished by multiplying $\boldsymbol{S}'^{(i-1)}(z)$ from the left by

$$\boldsymbol{B}_{\mathrm{l}}^{(i)}(z) = \mathrm{diag}\left\{\mathbf{1}_{n_i-1}, z^{-\tau_i}, \mathbf{1}_{M-n_i-1}\right\}, \tag{9}$$

which results in

$$\boldsymbol{S}^{(i-\frac{1}{2})}(z) = \boldsymbol{B}_{\mathrm{l}}^{(i)}(z)\boldsymbol{S}'^{(i-1)}(z). \tag{10}$$

This completes the time-shift of a column carrying maximum off-diagonal energy to the zero-lag coefficient matrix without disturbing the diagonal elements. Now similar to the first iteration, the off-diagonal elements of the zero-lag coefficient matrix of $\mathbf{S}^{(i-\frac{1}{2})}[\tau]$ are eliminated or in otherwords, the off-diagonal energy is transferred onto the diagonal by multiplying $\boldsymbol{S}^{(i-\frac{1}{2})}(z)$ on the left and right by $\mathbf{U}^{(i),\mathrm{H}}$ and $\mathbf{V}^{(i)}$, respectively, producing

$$\boldsymbol{S}^{(i)}(z) = \mathbf{U}^{(i),\mathrm{H}}\mathbf{S}^{(i-\frac{1}{2})}(z)\mathbf{V}^{(i)}, \tag{11}$$

where $\mathbf{U}^{(i)}$ and $\mathbf{V}^{(i)}$ are left and right-singular vector matrices of $\mathbf{S}^{(i-\frac{1}{2})}[0]$, respectively, and they are obtained via a conventional SVD, i.e., $\mathbf{S}^{(i-\frac{1}{2})}[0] = \mathbf{U}^{(i)}\mathbf{S}^{(i)}[0]\mathbf{V}^{(i),\mathrm{H}}$. It is important to remind that $\mathbf{U}^{(i)}$ and $\mathbf{V}^{(i)}$ being both unitary, only transfer the off-diagonal energy of the zero-lag coefficient matrix onto its diagonal without changing the total energy of $\boldsymbol{S}^{(i-\frac{1}{2})}(z)$. This completes the ith iteration of the proposed algorithm with the zero-lag coefficient matrix of $\boldsymbol{S}^{(i)}(z)$ fully diagonal and with more diagonal energy than it had in the previous iteration. The left and right polynomial matrices are updated at the end of ith iteration as

$$\boldsymbol{U}^{(i)}(z) = \boldsymbol{U}^{(i-1)}(z)\boldsymbol{B}_{\mathrm{l}}^{(i),\mathrm{P}}(z)\mathbf{U}^{(i)}, \tag{12}$$

$$\boldsymbol{V}^{(i)}(z) = \boldsymbol{V}^{(i-1)}(z)\boldsymbol{B}_{\mathrm{r}}^{(i)}(z)\mathbf{V}^{(i)}. \tag{13}$$

As evident from the iterative updates of all three polynomial matrices, the polynomial order of $\boldsymbol{S}^{(i)}(z)$ increases by $2|\tau_i|$ in each iteration whereas that of the left and right paraunitary singular vectors increases by $|\tau_i|$. This process is repeated in each iteration until $\gamma_{c(n_i,\tau_i)}$, defined in (6), falls below a small preset threshold ϵ, or until a maximum number of allowed iterations is reached. After I iterations, the approximate polynomial SVD factors of (1) are extracted as

$$\hat{\boldsymbol{U}}(z) = \boldsymbol{U}^{(I)}(z),\ \hat{\boldsymbol{\Sigma}}(z) = \boldsymbol{S}^{(I)}(z),\ \hat{\boldsymbol{V}}(z) = \boldsymbol{V}^{(I)}(z). \tag{14}$$

Akin to SMD [9] and SBR2 [8], the singular values extracted by the proposed algorithm are always spectrally majorised [37]. In other words, when singular values are evaluated on the unit circle, they are strictly ordered at each frequency, regardless of whether the ground-truth singular values are unmajorised (see (2)). The proof that the proposed algorithm always converges to a spectrally majorised solution follows identically from that of the SBR2 algorithm [38]. However, since it has been shown in [24,25] that any polynomial matrix – whether randomly generated or estimated from a finite set of sample points – exhibits spectrally majorised singular values. Consequently, for all such matrices, the proposed algorithm will converge to the ground-truth singular values which are also spectrally majorised. A complete iteration of the proposed algorithm is pictorially depicted in Fig. 1.

This version of the algorithm, which utilizes the maximum ℓ_2−norm of a column, is referred to as column-based generalized sequential matrix diagonalization (CGSMD). A complete outline of the CGSMD is provided in Algorithm 1. Conversely, if an ℓ_∞-norm is employed in place of the ℓ_2-norm in (5), i.e.,

$$\{\tau_i, n_i\} = \underset{\tau,n}{\mathrm{argmax}} \left\{ \sum_{m=1,m\neq n}^{M} |s_{m,n}^{(i-1)}[\tau]|^p \right\}^{\frac{1}{p}}, \tag{15}$$

where $p \to \infty$, the resulting algorithm is designated as the maximum element CGSMD (ME-CGSMD). This variant searches for the column containing the largest off-diagonal element instead of the column with the largest off-diagonal energy. Consequently, the ℓ_∞−norm based variant is expected to transfer less energy to the diagonal of the zero-lag coefficient in every iteration, and therefore may converge slow. This issue is further explored in the simulation and results section.

Algorithm 1: CGSMD Algorithm

Input: $\boldsymbol{A}(z), \epsilon$
Output: $\hat{\boldsymbol{U}}(z),\ \hat{\boldsymbol{V}}(z),\ \hat{\boldsymbol{\Sigma}}(z)$
$\mathbf{A}[0] = \mathbf{U}^{(0)}\mathbf{S}^{(0)}[0]\mathbf{V}^{(0),\mathrm{H}}$;
$\boldsymbol{S}^{(0)}(z) = \mathbf{U}^{(0),\mathrm{H}}\boldsymbol{A}(z)\mathbf{V}^{(0)}$;
$\boldsymbol{U}^{(0)}(z) = \mathbf{U}^{(0)},\ \boldsymbol{V}^{(0)}(z) = \mathbf{V}^{(0)}$;
$\gamma^2_{c(n_0,\tau_0)} = 1 + \epsilon; i = 1;$
while $i \leq I$ and $\gamma^2_{c(n_{i-1},\tau_{i-1})} > \epsilon$ **do**
  Perform maximum search for column;
  $\{\tau_i, n_i\} = \underset{\tau,n}{\mathrm{argmax}} \left\{ \sum_{m=1,m\neq n}^{M} |s_{m,n}^{(i-1)}[\tau]|^2 \right\}^{\frac{1}{2}}$, $\gamma_{c(n_i,\tau_i)} = \left\{ \sum_{\substack{m=1 \\ m\neq n_i}}^{M} |s_{m,n_i}[\tau_i]|^2 \right\}^{\frac{1}{2}}$;
  Construct right and left delay matrices;
  $\boldsymbol{B}_{\mathrm{r}}^{(i)}(z) = \mathrm{diag}\left\{\mathbf{1}_{n_i-1}, z^{\tau_i}, \mathbf{1}_{N-n_i-1}\right\}$, $\boldsymbol{B}_{\mathrm{l}}^{(i)}(z) = \mathrm{diag}\left\{\mathbf{1}_{n_i-1}, z^{-\tau_i}, \mathbf{1}_{M-n_i-1}\right\}$;
  Shift the maximum column onto the zero-lag;
  $\boldsymbol{S}^{(i-\frac{1}{2})}(z) = \boldsymbol{B}_{\mathrm{l}}^{(i)}(z)\boldsymbol{S}^{(i-1)}(z)\boldsymbol{B}_{\mathrm{r}}^{(i)}(z)$;
  Diagonalize the zero-lag coefficient matrix;
  $\mathbf{S}^{(i-\frac{1}{2})}[0] = \mathbf{U}^{(i)}\mathbf{D}^{(i)}\mathbf{V}^{(i),\mathrm{H}}$;
  Via unitary rotation, transfer the off-diagonal energy onto the diagonal of zero-lag coefficient matrix of $\boldsymbol{S}^{(i-\frac{1}{2})}(z)$;
  $\boldsymbol{S}^{(i)}(z) = \mathbf{U}^{(i),\mathrm{H}}\mathbf{S}^{(i-\frac{1}{2})}(z)\mathbf{V}^{(i)}$;
  Update paraunitary matrices;
  $\boldsymbol{U}^{(i)}(z) = \boldsymbol{U}^{(i-1)}(z)\boldsymbol{B}_{\mathrm{l}}^{(i),\mathrm{P}}(z)\mathbf{U}^{(i)}$;
  $\boldsymbol{V}^{(i)}(z) = \boldsymbol{V}^{(i-1)}(z)\boldsymbol{B}_{\mathrm{r}}^{(i)}(z)\mathbf{V}^{(i)}$;
  Limit the polynomial order growth through truncation as described in Sec. 4.4;
  $i \leftarrow i + 1$;
end
$\hat{\boldsymbol{U}}(z) = \boldsymbol{U}^{(i-1)}(z),\ \hat{\boldsymbol{V}}(z) = \boldsymbol{V}^{(i-1)}(z),\ \hat{\boldsymbol{\Sigma}}(z) = \boldsymbol{\Sigma}^{(i-1)}(z)$

4.3. Proof of convergence

To provide a proof of convergence for the proposed algorithm, the following quantities are defined:

$$\alpha_1\{\boldsymbol{S}^{(i)}(z)\} \triangleq \sum_{n=1}^{N} |s_{n,n}^{(i)}[0]|^2,$$

$$\alpha_2\{\boldsymbol{S}^{(i)}(z)\} \triangleq \sum_{m=1}^{M}\sum_{n=1}^{N} |s_{m,n}^{(i)}[0]|^2,$$

$$\alpha_3\{\boldsymbol{S}^{(i)}(z)\} \triangleq \sum_{\substack{m=1 \\ m\neq n}}^{M}\sum_{n=1}^{N} |s_{m,n}^{(i)}[0]|^2,$$

$$\alpha_4\{\boldsymbol{S}^{(i)}(z)\} \triangleq \sum_{\tau}\sum_{m=1}^{M}\sum_{n=1}^{N} |s_{m,n}^{(i)}[\tau]|^2.$$

It follows that $\alpha_2\{\boldsymbol{S}^{(i)}(z)\} = \alpha_1\{\boldsymbol{S}^{(i)}(z)\} + \alpha_3\{\boldsymbol{S}^{(i)}(z)\}$ and $\alpha_1\{\boldsymbol{S}^{(i)}(z)\} \leq \alpha_4\{\boldsymbol{S}^{(i)}(z)\}$. The quantity $\alpha_1\{\cdot\}$ remains invariant under the simultaneous application of left and right delay matrices, i.e., $\alpha_1\{\boldsymbol{S}^{(i+\frac{1}{2})}(z)\} = \alpha_1\{\boldsymbol{B}_{\mathrm{l}}^{(i)}(z)\boldsymbol{S}^{(i)}(z)\boldsymbol{B}_{\mathrm{r}}^{(i)}(z)\}$.

The right multiplication moves the n_i-th column's coefficients from lag τ_i to 0 and moves $s_{n_i,n_i}^{(i-1)}[0]$ to $-\tau_i$; the left multiplication by $z^{-\tau_i}$ on row n_i restores it to the lag 0. Similarly, $\alpha_2\{\cdot\}$ is invariant under unitary transformations., i.e., $\alpha_2\{\boldsymbol{S}^{(i)}(z)\} = \alpha_2\{\mathbf{U}^{(i),\mathrm{H}}\boldsymbol{S}^{(i-\frac{1}{2})}(z)\mathbf{V}^{(i)}\}$ in the SVD step. Lastly, $\alpha_4\{\cdot\}$ is invariant to the application of a paraunitary matrix, i.e., $\alpha_4\{\boldsymbol{S}^{(i)}(z)\} = \alpha_4\{\boldsymbol{U}^{(i),\mathrm{P}}(z)\boldsymbol{S}^{(i)}(z)\boldsymbol{V}^{(i)}(z)\}$. This holds because $\boldsymbol{B}_{\mathrm{l}}^{(i)}(z)$ and $\boldsymbol{B}_{\mathrm{r}}^{(i)}(z)$ are paraunitary and only reindex lag coefficients, and

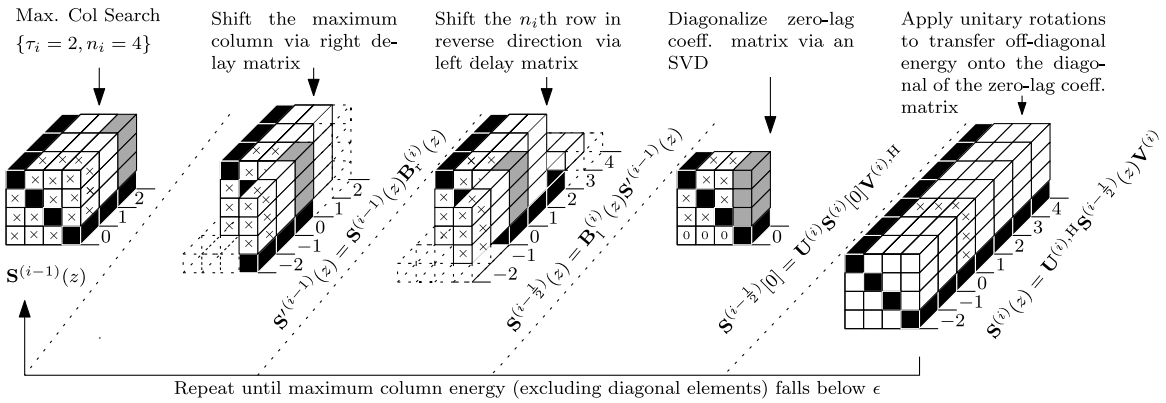

Fig. 1. The zero-lag coefficient matrix $\boldsymbol{S}^{(i-1)}[0]$ at the start of ith iteration is diagonal with off-diagonal elements shown as ×, denoting they are eliminated or made zero in previous iteration. Diagonal entries are shown as black. The column with the largest off-diagonal energy is determined, here shown in darkgray with $n_i = 4, \tau_i = 2$, and then shifted to the zero-lag position via delay matrix, in this case $\boldsymbol{B}_{\mathrm{r}}^{(i)} = \mathrm{diag}\{1, 1, 1, z^2\}$, multiplied from the right. This causes the last diagonal element to be shifted out of the zero-lag coefficient matrix, which is brought back to the zero-lag position by shifting the n_ith row in reverse direction by the same amount of delay, i.e. $\boldsymbol{B}_{\mathrm{l}}^{(i)} = \mathrm{diag}\{1, 1, 1, z^{-2}\}$. This causes the maximum column elements to get shifted onto the zero-lag coefficient matrix without disturbing its diagonal elements. Unitary rotation are applied to the entire matrix such that the zero-lag coefficient matrix is a diagonal matrix. These unitary rotation matrices are actually the left- and right-singular vector matrices of the zero-lag coefficient of $\boldsymbol{S}^{(i-\frac{1}{2})}(z)$ which is obtained by its SVD. The process is repeated until the column energy is below ϵ or maximum I are expended.

multiplication on the left and on the right by $\mathbf{U}^{(i),H}$ and $\mathbf{V}^{(i)}$ leaves the Frobenius norm of each lag slice unchanged: $\|\mathbf{U}^{(i),H}\,\boldsymbol{S}^{(i-\frac{1}{2})}[\tau]\,\mathbf{V}^{(i)}\|_{\mathrm{F}} = \|\boldsymbol{S}^{(i-\frac{1}{2})}[\tau]\|_{\mathrm{F}} \quad \forall\,\tau$.

At the end of $(i-1)$th iteration, $\alpha_1 = \alpha_2$ with $\alpha_3 = 0$ due to zero off-diagonal energy in the zero-lag $\mathbf{S}^{(i-1)}[0]$. In the ith iteration, post application of left and right delay matrices $\boldsymbol{B}_{\mathrm{l}}^{(i)}(z)$ and $\boldsymbol{B}_{\mathrm{r}}^{(i)}(z)$, respectively, α_1 and α_4 remains unchanged, i.e., $\alpha_1\{\boldsymbol{S}^{(i-\frac{1}{2})}(z)\} = \alpha_1\{\boldsymbol{S}^{(i-1)}(z)\}$ and $\alpha_4\{\boldsymbol{S}^{(i-\frac{1}{2})}(z)\} = \alpha_4\{\boldsymbol{S}^{(i-1)}(z)\}$. Since the delay operations shifted the n_ith column at lag τ_i, and the n_ith row at lag $-\tau_i$ to the zero-lag coefficient matrix of $\mathbf{S}^{(i-\frac{1}{2})}[\tau]$, we have

$$\alpha_2\{\boldsymbol{S}^{(i-\frac{1}{2})}(z)\} = \alpha_1\{\boldsymbol{S}^{(i-\frac{1}{2})}(z)\} + \gamma^2_{c(n_i,\tau_i)} + \gamma^2_{r(n_i,-\tau_i)},$$

and

$$\alpha_3\{\boldsymbol{S}^{(i-\frac{1}{2})}(z)\} = \gamma^2_{c(n_i,\tau_i)} + \gamma^2_{r(n_i,-\tau_i)},$$

where $\gamma^2_{c(n_i,\tau_i)} = \sum_{\substack{m=1\\ m\neq n_i}}^{M} |s^{(i-1)}_{m,n_i}[\tau_i]|^2$, and $\gamma^2_{r(n_i,-\tau_i)} = \sum_{\substack{n=1\\ n\neq n_i}}^{N} |s^{(i-1)}_{n_i,n}[-\tau_i]|^2$. Unlike the GSBR2 algorithm, where the diagonal energy gets increased by the squared magnitude of the largest off-diagonal element, GSMD increases the diagonal energy in each iteration at least by the squared off-diagonal norm of the largest off-diagonal column. Therefore, the proposed algorithm achieves faster convergence. Once $\mathbf{U}^{(i)}$ and $\mathbf{V}^{(i)}$, the left and right singular vectors of $\mathbf{S}^{(i-\frac{1}{2})}[0]$, are applied to each lag, the off-diagonal energy in the zero-lag gets transferred onto the diagonal resulting in

$$\begin{aligned}
\alpha_1\{\boldsymbol{S}^{(i)}(z)\} &= \alpha_1\{\boldsymbol{S}^{(i-1)}(z)\} + \gamma^2_{c(n_i,\tau_i)} + \gamma^2_{r(n_i,-\tau_i)}\\
\alpha_2\{\boldsymbol{S}^{(i)}(z)\} &= \alpha_2\{\boldsymbol{S}^{(i-\frac{1}{2})}(z)\}\\
\alpha_3\{\boldsymbol{S}^{(i)}(z)\} &= 0.
\end{aligned}$$

This shows that diagonal energy of the zero-lag, i.e., $\alpha_1\{\boldsymbol{S}^{(i)}(z)\}$ increases monotonically with each iteration while the total energy, i.e., $\alpha_4\{\boldsymbol{S}^{(i)}(z)\}$ remains the same. It therefore forms the upper bound for $\alpha_1\{\boldsymbol{S}^{(i)}(z)\}$, i.e., $\alpha_1\{\boldsymbol{S}^{(i)}(z)\} \leq \alpha_4\{\boldsymbol{S}^{(i)}(z)\}\ \forall i$. Therefore, $\alpha_1\{\boldsymbol{S}^{(i)}(z)\}$ must have a supremum β. Hence for $\varepsilon > 0$, there must be an iteration number I for which we have

$$\beta - \alpha_1\{\boldsymbol{S}^{(I)}(z)\} < \varepsilon.$$

After some subsequent non-zero iterations $\ell > 0$, it is easy to prove that

$$\gamma^2_{c(n_{I+\ell},\tau_{I+\ell})} + \gamma^2_{r(n_{I+\ell},-\tau_{I+\ell})} < \beta - \alpha_1\{\boldsymbol{S}^{(I)}(z)\} < \varepsilon.$$

This proves that after sufficiently large number of iterations, the maximum off-diagonal energy is bounded by $\varepsilon > 0$.

It must be noted that although $\alpha_1\{\boldsymbol{S}^{(i)}(z)\}$ increase monotonically increases in each iteration, the maximum off-diagonal column energy may not necessarily decrease monotonically. This means that the overall off-diagonal energy at subsequent iterations can be larger than that of the previous iterations. Due to the iterative nature of the proposed algorithm, the difference between $\alpha_4\{\boldsymbol{S}^{(i)}(z)\}$, which is invariant to iterations, and β will generally be non-zero unless $\boldsymbol{A}(z)$ is a diagonal matrix. Increasing the number of iterations improves the diagonalization and the gap between the total energy β and the diagonal energy $\alpha_1\{\boldsymbol{S}^{(I)}(z)\}$ decreases further. Consequently, achieving a smaller target ϵ requires more iterations compared to a larger one. In algorithmic sense, the parameter ϵ implicitly governs the number of iterations needed by the algorithm.

Note that the proposed method does not rely on any additional assumptions for convergence, except that $\alpha_4\{\boldsymbol{A}(z)\}$ remains finite. This quantity is invariant to delays, unitary transformations, and, therefore, to the iteration index. As long as this condition holds, the proposed algorithm will iteratively converge toward a diagonal matrix.

4.4. Polynomial order growth and truncation

The growth in polynomial order of all three matrices must be controlled to reduce the computational complexity of the algorithm akin to the case with SMD and SBR2 [39,40]. This will subsequently minimize implementation costs when $\hat{\boldsymbol{V}}(z)$ and $\hat{\boldsymbol{U}}(z)$ are realized via DSPs or FPGAs in applications. Since $\mathbf{S}^{(i)}[\tau]$ is a Laurent polynomial but not para-Hermitian symmetric, it is truncated by discarding lags from $\tau = \tau_{\min}$, where $\tau_{\min}$ represents the least maximum lag for which $\mathbf{S}^{(i)}[\tau]_{\tau<\tau_{\min}} = 0$, to $\tau = \tau_{\mathrm{trun}-}$, and from $\tau = \tau_{\mathrm{trun}+}$ to $\tau = \tau_{\max}$, $\tau_{\max}$ is the minimum lag for which $\boldsymbol{S}^{(i)}[\tau]|_{\tau>\tau_{\max}} = 0$, via a parameter μ_{PH} that determines $\tau_{\mathrm{trun}-}$ and $\tau_{\mathrm{trun}+}$ as

$$\frac{\sum_{\tau=\tau_{\min}}^{\tau_{\mathrm{trun}-}} \|\mathbf{S}^{(i)}[\tau]\|_{\mathrm{F}}^2}{\alpha_4\{\boldsymbol{A}(z)\}} > \frac{\mu_{\mathrm{PH}}}{2},\quad \frac{\sum_{\tau=\tau_{\mathrm{trun}+}}^{\tau_{\max}} \|\mathbf{S}^{(i)}[\tau]\|_{\mathrm{F}}^2}{\alpha_4\{\boldsymbol{A}(z)\}} > \frac{\mu_{\mathrm{PH}}}{2}. \tag{16}$$

Unlike SBR2/SMD, the truncation here is not symmetric due to the absence of para-Hermitian symmetry.

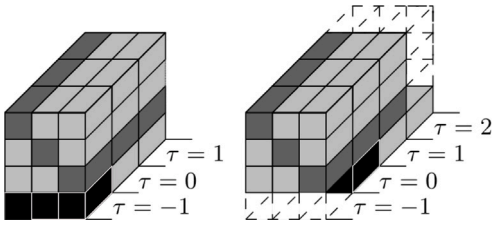

Fig. 2. An example case shown for RGSMD with $\mathbf{S}^{(i)}[\tau] \in \mathbb{C}^{4\times 3}$ of polynomial order 2 or 3 time lags. The dominant row (in dark black) has the row index $m_i = 4$, which exceed the numbers of columns, and reside at lag $\tau = -1$. Shifting it to the *tau* = 0 does not affect the diagonal elements (in dark gray), making the right delay matrix an identity. This causes the order of the shifted matrix to increase by $|\tau_i| = 1$ instead of by $2|\tau_i|$.

In the proposed approach, columns are either delayed or advanced, making the right paraunitary matrix a Laurent polynomial matrix; the same applies to rows, and therefore, to the left paraunitary matrix. As iterations progress, the coefficients of all three intermediate matrices decay at both ends. Unlike the SMD and SBR2 where column is strictly advanced, and row is delayed in every iteration, the proposed method and its variants, discussed below, have the flexibility to advance and delay both the columns and rows. Therefore, along with $\boldsymbol{S}^{(i)}(z)$, the coefficients of matrices $\boldsymbol{U}^{(i)}(z), \boldsymbol{V}^{(i)}(z)$ can decay on either ends necessitating truncation of exterior lags on either end. This is carried out via a μ_{PU} parameter

$$\frac{\sum_{\tau=\tau_{\min}}^{\tau_{\text{trun}-}} \|\mathbf{U}^{(i)}[\tau]\|_{\text{F}}^2}{\sum_{\tau=\tau_{\min}}^{\tau=\tau_{\max}} \|\mathbf{U}^{(i)}[\tau]\|_{\text{F}}^2} > \mu_{\text{PU}}, \quad \frac{\sum_{\tau=\tau_{\text{trun}+}}^{\tau_{\max}} \|\mathbf{U}^{(i)}[\tau]\|_{\text{F}}^2}{\sum_{\tau=\tau_{\min}}^{\tau=\tau_{\max}} \|\mathbf{U}^{(i)}[\tau]\|_{\text{F}}^2} > \mu_{\text{PU}}, \tag{17}$$

at each iteration of the proposed algorithm.

5. Variant of the proposed algorithm

Depending on the spatial dimensions and the energy distribution across the rows and columns of a polynomial matrix, various variants of the GSMD algorithm can be developed. We highlight several variants that transfer different amounts of energy in each iteration, which we will analyze through simulations to compare their convergence speed and computational cost.

5.1. Dominant row variant

In this variant, instead of focusing on columns, the algorithm identifies the dominant row with the maximum off-diagonal ℓ_2-norm, (5) at each iteration, defined by:

$$\{\tau_i, m_i\} = \underset{\tau,m}{\operatorname{argmax}} \left\{ \sum_{n=1,n\neq m}^{N} |s_{m,n}^{(i-1)}[\tau]|^2 \right\}^{\frac{1}{2}}. \tag{18}$$

By concentrating on the dominant row, this approach creates a row-based GSMD variant (RGSMD). If ℓ_2-norm is replaced with an $\ell_\infty-$norm in (18) as adopted in (15) for columns, we obtain the maximum element RGSMD (ME-RGSMD) variant. These row-based variants differ from the column-based variant in terms of the polynomial order growth of the intermediate matrices $\boldsymbol{S}^{(i)}(z), \boldsymbol{U}^{(i)}(z)$ and $\boldsymbol{V}^{(i)}(z)$. For instance, if in a certain iteration, the dominant row index exceeds the number of columns, i.e., $m_i > N$, the diagonal element will not be shifted due to the application of $\boldsymbol{B}_{\text{l}}^{(i)}(z)$ from the left, therefore, making $\boldsymbol{B}_{\text{r}}^{(i)}(z)$ an identity. See an illustrative example in Fig. 2. As a result, in such an iteration, the polynomial order of $\boldsymbol{S}^{(i)}(z)$ and $\boldsymbol{U}^{(i)}(z)$ would grow by $|\tau_i|$, while that of $\boldsymbol{V}^{(i)}(z)$ remains the same. While if there is a case with $m_i \leq N$ in any iteration, the RGSMD variant works similarly to the CGSMD, and the ME-RGSMD works similarly to the ME-CGSMD.

5.2. Dominant row based extra shift variant

This variant introduces an additional shift in the dominant row approach, allowing for more flexibility in transferring energy, particularly when dealing with the lower submatrix of $\boldsymbol{A}(z)$. This is achieved at a cost of minimum additional computation and not increase in polynomial order. That is, after determining $\{\tau_i, m_i\}$ via (18), this variant transfers the entire lower $(M-N)\times N$ submatrix, which does not contain any diagonal element, from $\tau = \tau_i$ to the zero-lag along with the m_ith row. The submatrix transfer may enable greater energy transfer to the zero-lag diagonal, potentially accelerating the diagonalization process. This accomplished via the left delay matrix, which varies depending upon m_i as follows:

$$\begin{aligned} m_i \leq N \;:\; \boldsymbol{B}_{\text{l}}^{(i)}(z) &= \operatorname{diag}\left\{ \mathbf{1}_{m_i-1}, z^{\tau_i}, \mathbf{1}_{N-m_i}, \mathbf{1}_{M-N} z^{\tau_i} \right\}, \\ \boldsymbol{B}_{\text{r}}^{(i)}(z) &= \operatorname{diag}\left\{ \mathbf{1}_{m_i-1}, z^{-\tau_i}, \mathbf{1}_{N-m_i-1} \right\} \\ m_i > N \;:\; \boldsymbol{B}_{\text{l}}^{(i)}(z) &= \operatorname{diag}\left\{ \mathbf{1}_N,\ \mathbf{1}_{M-N} z^{\tau_i} \right\},\ \boldsymbol{B}_{\text{r}}^{(i)}(z) = \mathbf{I}_N \end{aligned} \tag{19}$$

In case $m_i \leq N$, the corresponding m_ith column is shifted in the reverse direction to bring back the diagonal elements to their original position. This variant is termed as extra-shift RGSMD (ES-RGSMD) variant. The entire procedure of this variant is same as outlined in Algorithm 1 with left and right delay matrices replaced with (19) and column norm is replaced with row norm. If extra shifts are performed within the framework of ME-RGSMD, it is denoted with abbreviation ES-ME-RGSMD. The polynomial order growth of intermediate matrices in these variants is same as that in the RGSMD/ME-RGSMD.

5.3. Hybrid column-row GSMD variant

This hybrid variant combines both column and row-based approaches, allowing the algorithm to choose between transferring energy via the dominant row or column based on which provides a higher energy transfer in each iteration. In the hybrid column-row GSMD variant, the algorithm can switch between the row-based (RGSMD) and column-based (CGSMD) or between ES-RGSMD and CGSMD approaches depending on the relative off-diagonal energy norms.

To accomplish this feat, this variant performs two searches in each iteration: one to identify the dominant column index n_i and its lag $\tau_{i,c}$, while the second identifies the dominant row parameters, i.e., m_i and $\tau_{i,r}$. The additional subscripts in the lag parameters distinguish between row and column lags, with the scope of this notation limited to this subsection. After identifying both, if the off-diagonal norm of the m_ith row situated at $\tau = \tau_{i,r}$ exceeds that of the n_ith column at $\tau = \tau_{i,c}$, the row is transferred, making this iteration similar to an RGSMD iteration. Otherwise, the n_ith column at $\tau = \tau_{i,c}$ is transferred to the zero-lag, resulting in a CGSMD-like iteration. This variant is known as hybrid column-row GSMD (HCR-GSMD). If instead of RGSMD, the algorithm switches between ES-RGSMD and CGSMD, slightly more energy is expected to be transferred onto the zero-lag in each iteration. This type of variant is known as extra-shift HCR-GSMD (ES-HCR-GSMD).

5.4. Multiple maximum element shift (MSME-gsmd)

This variant extends the GSMD algorithm by allowing multiple off-diagonal elements to be shifted in a single iteration. Since this variant involves multiple maximum-element shifts within a single iteration, we introduce an additional subscript to keep track of both column and row indices of each element being shifted. Here, as in the previous sections, i denotes the iteration number, and j denotes the sequence of maximum element shifts within the same iteration.

First a maximum search is performed within the upper $N\times N$ submatrix of $\boldsymbol{S}^{(i-1)}(z)$ in the ith iteration. Let us assume the first maximum off-diagonal element is located at coordinates $\{m_{(i,1)}, n_{(i,1)}, \tau_{(i,1)}\}$. This

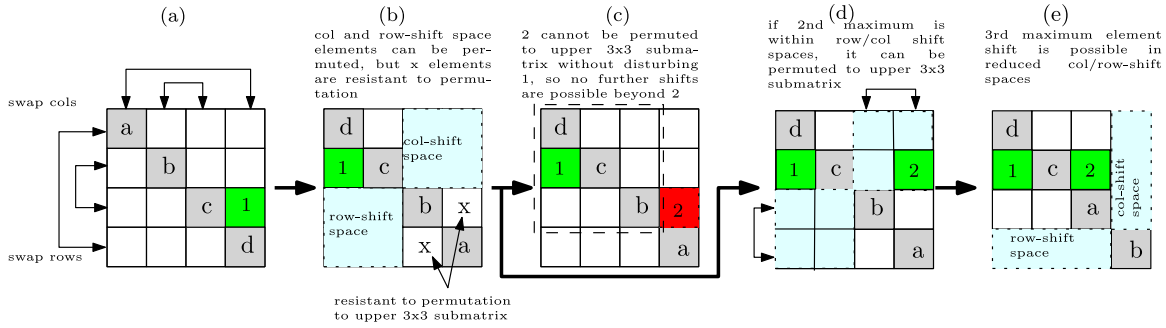

Fig. 3. Example of MSME-GSMD with a zero-lag coefficient matrix (diagonal elements in light gray). (a) The first maximum off-diagonal element (green, labeled 1) is shifted to the zero-lag position and permuted into the upper 2×2 submatrix using row/column shifts. The corresponding row- and column-shift spaces are highlighted in light blue in (b). Elements outside col/row-shift spaces (marked x) are resistant to permutation into the upper 3×3 submatrix. If the second maximum off-diagonal element (red, labeled 2 in (c)) lies outside the shift spaces, it cannot be moved into the upper submatrix without disturbing element 1, and therefore, no further shifts beyond 2 are possible in this iteration. (d) If the second maximum lies within the col or row-shift spaces, as highlighted in green with label 2, it can be shifted to the zero-lag position and easily permuted to the upper 3×3 submatrix if not already lies in it without affecting the upper 2×2 submatrix. This allows a 3rd maximum element shift in reduced col/row-shift space, as shown in (e), in the same iteration making total possible shifts $N-1=3$.

element is time-shifted from $\tau_{i,1}$ to $\tau = 0$ via the left- and right-shift matrices in (7) and (9), respectively, resulting in

$$\boldsymbol{S}^{(i-1,\frac{1}{2})}(z) = \boldsymbol{B}_{\mathrm{l}}^{(i,1)}(z)\boldsymbol{S}^{(i-1)}(z)\boldsymbol{B}_{\mathrm{r}}^{(i,1)}(z)\,.$$

Next, a permutation is applied to move this maximum off-diagonal element to the upper 2×2 matrix, specifically to the second row and first column, i.e., at index $(2,1)$. However, permutations are not applied if it is already an off-diagonal element of the upper 2×2 submatrix. In general if $(m_{(i,1)}, n_{(i,1)}) \notin \{(1,2),(2,1)\}$, the left and right permutation matrices for the first shift can be expressed as the product of the two modified identity matrices:

$$\mathbf{P}_{\mathrm{l}}^{(i,1)} = \mathbf{I}_M^{(m_{(i,1)}\leftrightarrow 2)} \cdot \mathbf{I}_M^{(n_{(i,1)}\leftrightarrow 1)}, \qquad \mathbf{P}_{\mathrm{r}}^{(i,1)} = \mathbf{I}_N^{(m_{i,1}\leftrightarrow 2)} \cdot \mathbf{I}_N^{(n_{(i,1)}\leftrightarrow 1)},$$

where $\mathbf{I}_N^{(m_{(i,1)}\leftrightarrow 2)}$ denotes an $N \times N$ identity matrix with its $m_{i,1}$-th column swapped with the second column. This permutation is applied to all lags which produces

$$\boldsymbol{S}^{(i-1,1)}(z) = \mathbf{P}_{\mathrm{l}}^{(i,1)}\boldsymbol{S}^{(i-1,\frac{1}{2})}(z)\mathbf{P}_{\mathrm{r}}^{(i,1)},$$

to ensure that subsequent shifts within the same iteration do not affect the first maximum off-diagonal element, provided the upper left 2×2 submatrix remains unchanged. An example Fig. 3(a) illustrates an off-diagonal element (green 1) permuted to $(2,1)$ by swapping the first column with the fourth, and second column with the third, and similarly for rows.

After the first maximum off-diagonal shift and permutation, the subsequent search is performed within two reduced search areas: the column-shift and row-shift spaces. These two reduced search spaces after the first maximum off-diagonal shift and permutation are $\boldsymbol{S}_{1:2,3:N}^{(i-1,1)}(z)$ and $\boldsymbol{S}_{3:M,1:2}^{(i-1,1)}(z)$, respectively, as shown in Fig. 3(b). Any off-diagonal element in the column-shift space can be translated to $\tau = 0$ through a column shift, and via a row shift if it is within the row-shift space. Selecting an off-diagonal element outside these regions reduces the possible number of maximum element shifts within a single iteration, as elements outside these spaces cannot be transferred to the upper 3×3 section without disturbing the upper 2×2 matrix. An example in Fig. 3(b) shows light yellow elements, marked as x, outside the column and row-shift spaces if being selected as the maximum element, limiting the number of shifts to two as shown in Fig. 3(c) with maximum element showed in red as the second shift, while otherwise three shifts are possible as evident in Fig. 3(d) and (e).

In general, for subsequent maximum off-diagonal element shifts, i.e., $j > 1$, within the ith iteration, the maximum off-diagonal element coordinates $\{m_{(i,j)}, n_{(i,j)}, \tau_{(i,j)}\}$ is determined within

$$\begin{bmatrix} \mathbf{0}_{j\times j} & \boldsymbol{S}_{1:j,j+1:N}^{(i-1,j-1)} \\ \boldsymbol{S}_{j+1:N,1:j}^{(i-1,j-1)} & \mathbf{0}_{(N-j)\times(N-j)} \end{bmatrix}.$$

If $m_{(i,j)} < n_{(i,j)}$, the element being in column-shift space is time-shifted via left and right shift matrices

$$\boldsymbol{B}_{\mathrm{r}}^{(i,j)}(z) = \mathrm{diag}\left\{\mathbf{1}_{n_{(i,j)}-1}, z^{\tau_{(i,j)}}, \mathbf{1}_{N-n_{(i,j)}-1}\right\}, \tag{20a}$$

$$\boldsymbol{B}_{\mathrm{l}}^{(i,j)}(z) = \mathrm{diag}\left\{\mathbf{1}_{n_{(i,j)}-1}, z^{-\tau_{(i,j)}}, \mathbf{1}_{M-n_{(i,j)}-1}\right\}, \tag{20b}$$

with permutation matrices

$$\mathbf{P}_{\mathrm{l}}^{(i,j)} = \mathbf{I}_M^{(n_{(i,j)}\leftrightarrow j+1)}; \ \mathbf{P}_{\mathrm{r}}^{(i,j)} = \mathbf{I}_N^{(n_{(i,j)}\leftrightarrow j+1)}.$$

However, if $m_{(i,j)} > n_{(i,j)}$, shift matrices are

$$\boldsymbol{B}_{\mathrm{r}}^{(i,j)}(z) = \mathrm{diag}\left\{\mathbf{1}_{m_{(i,j)}-1}, z^{-\tau_{(i,j)}}, \mathbf{1}_{N-m_{(i,j)}-1}\right\}, \tag{21a}$$

$$\boldsymbol{B}_{\mathrm{l}}^{(i,j)}(z) = \mathrm{diag}\left\{\mathbf{1}_{m_{(i,j)}-1}, z^{\tau_{(i,j)}}, \mathbf{1}_{M-m_{(i,j)}-1}\right\}, \tag{21b}$$

with $\mathbf{P}_{\mathrm{l}}^{(i,j)} = \mathbf{I}_M^{(m_{(i,j)}\leftrightarrow j+1)}$; $\mathbf{P}_{\mathrm{r}}^{(i,j)} = \mathbf{I}_N^{(m_{(i,j)}\leftrightarrow j+1)}$. The update goes as

$$\boldsymbol{S}^{(i-1,j)}(z) = \mathbf{P}_{\mathrm{l}}^{(i,j)}\boldsymbol{B}_{l}^{(i,j)}(z)\boldsymbol{S}^{(i-1,j-1)}(z)\boldsymbol{B}_{r}^{(i,j)}(z)\mathbf{P}_{\mathrm{r}}^{(i,j)}.$$

With a total of $j = 1,\ldots,(N-1)$ maximum element shifts with embedded permutations, the lower $(M-N)\times N$ submatrix which remains unchanged excepts that elements within are swapped column wise due to permutations. Two different approaches can be applied to transfer additional rows:

5.4.1. Same lag transfer

The same lag transfer approach determines the lag with the maximum energy in the entire lower submatrix, that is

$$\tau_{(i,N)} = \underset{\tau}{\mathrm{argmax}} \ \|\mathbf{S}_{N+1:M,1:N}^{(i-1,N-1)}[\tau]\|_{\mathrm{F}}^2, \tag{22}$$

which is then transferred to $\tau = 0$ via $\boldsymbol{B}_{\mathrm{l}}^{(i,N)} = \mathrm{diag}\{\mathbf{1}_N, \mathbf{1}_{M-N} z^{\tau_{(i,N)}}\}$ and $\boldsymbol{B}_{\mathrm{r}}^{(i,N)}(z) = \mathbf{I}_N$ as

$$\boldsymbol{S}^{(i-1,N)}(z) = \boldsymbol{B}_{\mathrm{l}}^{(i-1,N-1)}(z)\boldsymbol{S}^{(i-1,N-1)}(z).$$

Thereafter, the SVD of the zero-lag is performed and its left and right singular matrices are applied to all lags

$$\boldsymbol{S}^{(i)}(z) = \mathbf{U}^{(i),\mathrm{H}}\boldsymbol{S}^{(i-1,N)}(z)\mathbf{V}^{(i)},$$

where $\mathbf{S}^{(i-1,N)}[0] = \mathbf{U}^{(i)}\mathbf{D}^{(i)}\mathbf{V}^{(i),\mathrm{H}}$ which completes the ith iteration of this variant. We denote this approach as same lag MSME-GSMD (SL-MSME-GSMD). The polynomial growth of $\boldsymbol{U}^{(i)}(z)$, $\boldsymbol{V}^{(i)}(z)$ and $\boldsymbol{\Sigma}^{(i)}(z)$ in the ith iteration is $\sum_{j=1}^{N}\tau_{(i,j)}$, $\sum_{j=1}^{N-1}\tau_{(i,j)}$ and $2\sum_{j=1}^{N-1}\tau_{(i,j)} + \tau_{(i,N)}$, respectively.

5.4.2. Individual rows transfer

Instead of the same lag transfer as in (22), we determine the time lag for maximum energy within each individual row of the lower $(M-N)\times N$ matrix as

$$\tau_{(i,j)} = \underset{\tau}{\operatorname{argmax}} \ \|\mathbf{s}_{j+1,:}^{(i-1,N-1)}[\tau]\|_2^2, \ j = N,\dots,(M-1). \tag{23}$$

Thereafter, each of these rows are transferred to $\tau = 0$ from its respective maximum lag $\tau_{(i,j)}, j = N,\dots,M$ via a left shift matrix $\boldsymbol{B}_{\mathrm{l}}^{(i,N)}(z) = \operatorname{diag}\{\mathbf{1}_N, z^{\tau_{(i,N)}},\dots,z^{\tau_{(i,M)}}\}$ with right shift matrix equal an identity. The rest of the procedure is the same as that discussed above for SL-MSME-GSMD. This separate row transfers may experience high polynomial order growth as compared to SL-MSME-GSMD but will often transfer more energy onto the diagonal in each iteration. It should be noted that these extra rows within the lower submatrix are shifted from different lags once all the necessary rows and columns in the upper $N\times N$ matrix are shifted. Hence they will increase the polynomial order growth of $\boldsymbol{S}^{(i)}(z)$ and $\boldsymbol{U}^{(i)}(z)$ unlike HCR-GSMD and ES-RGSMD where the low sub-matrix is shifted in parallel with the dominant row shift. For performance comparison, we denote this approach as different lag MSME-GSMD (DL-MSME-GSMD). The polynomial order of $\boldsymbol{U}^{(i)}(z)$, $\boldsymbol{V}^{(i)}(z)$ and $\boldsymbol{\Sigma}^{(i)}(z)$ at the end of ith iteration grows by $\sum_{j=1}^{M-1}\tau_{(i,j)}$, $\sum_{j=1}^{N-1}\tau_{(i,j)}$ and $2\sum_{j=1}^{N-1}\tau_{(i,j)}+\sum_{j=N}^{M-1}\tau_{(i,j)}$, respectively.

5.5. Convergence

Since each variant transfers a different amount of energy to the diagonal of the zero-lag coefficient matrix in each iteration, $\alpha_1\{\boldsymbol{S}^{(i)}(z)\}$ monotonously increases for all variants. Thus, the convergence proof presented above is valid for each variant.

6. Simulations and results

6.1. Performance metrics

6.1.1. Off-diagonal energy ratio

To assess the performance of various PSVD algorithms, we use the off-diagonal energy ratio

$$\beta^{(i)} = 5\log_{10}\left\{\bar{\alpha}_3\{\boldsymbol{S}^{(i)}(z)\}/\alpha_4\{\hat{\boldsymbol{S}}^{(i)}(z)\}\right\}, \tag{24}$$

where $\bar{\alpha}_3\{\boldsymbol{S}^{(i)}(z)\} = \sum_{\tau}\sum_{\substack{m=1\\ m\neq n}}^{M}\sum_{n=1}^{N}|s_{m,n}^{(i)}[\tau]|^2$. A lower value of $\beta^{(I)}$ indicates better diagonalization performance, as it signifies less off-diagonal energy. This metric is particularly important in applications such as broadband MIMO systems, where efficient diagonalization improves system performance and reduces interference. Ideally, $\beta^{(I)}$ should approach $-\infty$. In practice this is not achievable; an off-diagonal energy ratio below -10 dB is desirable [2,7]. .

6.1.2. Normalized error of estimated singular values

The accuracy of the diagonalization and the estimated singular values is assessed by comparing the singular values of the sample points of $\boldsymbol{A}(z)$ on the unit circle, i.e., $\boldsymbol{A}(\mathrm{e}^{\mathrm{j}\Omega_k})$, computed via a standard SVD, i.e., $\boldsymbol{A}(\mathrm{e}^{\mathrm{j}\Omega_k}) = \mathbf{U}_k\mathbf{D}_k\mathbf{V}_k{}^{\mathrm{H}}$ where $\mathbf{D}_k = \operatorname{diag}\{\sigma_1(\mathrm{e}^{\mathrm{j}\Omega_k}),\dots,\sigma_N(\mathrm{e}^{\mathrm{j}\Omega_k})\}$, with $\sigma_1(\mathrm{e}^{\mathrm{j}\Omega_k})\geq\cdots\geq\sigma_N(\mathrm{e}^{\mathrm{j}\Omega_k})$, and $\bar{\boldsymbol{\Sigma}}(\mathrm{e}^{\mathrm{j}\Omega_k})$ for $k=0,\dots,(K-1)$. The accuracy metric ξ is expressed as:

$$\xi = \frac{\sum_{k=0}^{K-1}\|\mathbf{D}_k - \operatorname{abs}\{\bar{\boldsymbol{\Sigma}}(\mathrm{e}^{\mathrm{j}\Omega_k})\}\|_F^2}{K\sum_{k=0}^{K-1}\|\mathbf{D}_k\|_{\mathrm{F}}^2},$$

where $\bar{\boldsymbol{\Sigma}}(\mathrm{e}^{\mathrm{j}\Omega_k})$ is equivalent to $\hat{\boldsymbol{\Sigma}}(\mathrm{e}^{\mathrm{j}\Omega_k})$ but its off-diagonal elements set to zero. The metric is computed at DFT size K that exceeds both the polynomial order of $\boldsymbol{A}(z)$ and $\hat{\boldsymbol{\Sigma}}(z)$.

6.1.3. Normalized reconstruction error

If $\hat{\boldsymbol{A}}(z) = \hat{\boldsymbol{U}}(z)\bar{\boldsymbol{\Sigma}}(z)\hat{\boldsymbol{V}}^{\mathrm{P}}(z)$ is the reconstructed polynomial matrix from the PSVD factors where $\bar{\boldsymbol{\Sigma}}(z)$ is same as $\hat{\boldsymbol{\Sigma}}(z)$ but its off-diagonal element forcefully made zero, we consider the reconstruction error $\boldsymbol{E}(z) = \boldsymbol{A}(z) - \hat{\boldsymbol{A}}(z)$ and define a normalized error metric

$$\zeta = \frac{\sum_{\tau}\|\mathbf{E}[\tau]\|_{\mathrm{F}}^2}{\sum_{\tau}\|\mathbf{A}[\tau]\|_{\mathrm{F}}^2}, \tag{25}$$

where $\mathbf{E}[\tau] \bullet\!\!-\!\!\circ\, \boldsymbol{E}(z)$.

6.1.4. Computational and implementation

Complexity The computational complexity of an algorithm is evaluated by its execution time, which is measured using the **tic** and **toc** functions in MATLAB′. In contrast, the implementation complexity can be quantified by the polynomial order of the PSVD factors [7] produced by each algorithm.

6.2. Worked example

In this worked example, we investigate the PSVD computation of a simple polynomial matrix

$$\boldsymbol{A}(z) = \begin{bmatrix} 1+2z^{-1}+3z^{-2} & z^{-1}-2z^{-2} & 4+z^{-1}+0.5z^{-2} \\ 2-z^{-1}+z^{-2} & 3+5z^{-1} & 1+z^{-1}+z^{-2} \\ 5+3z^{-1}+z^{-2} & 2+z^{-1}-z^{-2} & 5+3z^{-1}-z^{-2} \end{bmatrix} \tag{26}$$

via the CGSMD algorithm, as a demonstration, to analyze the zero-lag coefficient diagonal energy increase in each iteration and compare it against the GSBR2. We iterative both algorithm for 25 iteration, the resulting $\alpha\{\boldsymbol{S}^{(i)}(z)\}$, defined in Section 4.3, is shown in Fig. 4. The profile clearly shows that the GSMD algorithm is able to transfer more energy onto the diagonal of the zero-lag coefficient matrix than the GSBR2 algorithm because it diagonalize the entire zero-lag coefficient matrix through an SVD instead of zeroing a single element like GSBR2. Using the same example matrix as before, we now analyze the effect of ϵ on the GSMD algorithm. Therefore, we permit as many iterations as needed until the off-diagonal column norm falls below ϵ, with $\epsilon \in \{10^{-1},10^{-2},10^{-3},10^{-4},10^{-5}\}$, to study its impact on the total number of iterations. The truncation parameter μ_{PH} and μ_{PU} in (16) and (17) are both set to 10^{-6}. For the CGSMD variant, as an illustrative case, the algorithm requires 73, 224, 268, 309, and 352 iterations, respectively, to satisfy these thresholds. Smaller values of ϵ lead to more iterations and improved diagonalization. This is also reflected in the normalized singular value errors, which in this case are $\{18.2,\ 2.81,\ 2.76,\ 2.75,\ 2.74\}\times 10^{-7}$, progressively decreasing and indicating increasingly accurate singular value estimates.

6.3. Ensemble test setting

To evaluate the performance of the proposed GSMD algorithm variants, simulations were conducted on 2000 randomly generated polynomial matrices $\boldsymbol{A}(z)\in\mathbb{C}^{6\times 4}$ of polynomial order 10 with Gaussian-distributed coefficients. These matrices resemble polynomial matrices estimated from real-world data in the sense that the singular values of both remain majorised and the PSVD factors are often of infinite polynomial order [22,24,25,34], and therefore makes the ensemble test valid for evaluating PSVD methods. It is important to mention here that the DFT-based PSVD approach is excluded in this comparison due to its high computational cost [18]. Moreover, it is suitable only for cases involving short polynomial-order ground-truth singular vectors, which is typically not the case with randomly generated or estimated polynomial matrices [24]. For similar reasons, the iterative PQRD approach, which is not a PSVD approach but can be used to compute the PSVD, and the 2-PEVDs approach are also not considered. The GSBR2 approach represents the sole dedicated PSVD approach available in the literature.

All the proposed algorithms and the GSBR2 are simulated with $\epsilon = \mu_{\mathrm{PH}} = 10^{-6}$, $\mu_{\mathrm{PU}} = 10^{-5}$ for a maximum of $I = 250$ iterations unless the off-diagonal energy threshold is satisfied. All simulations are performed in MATLAB′ 2022b on a Dell (G15) i7 machine.

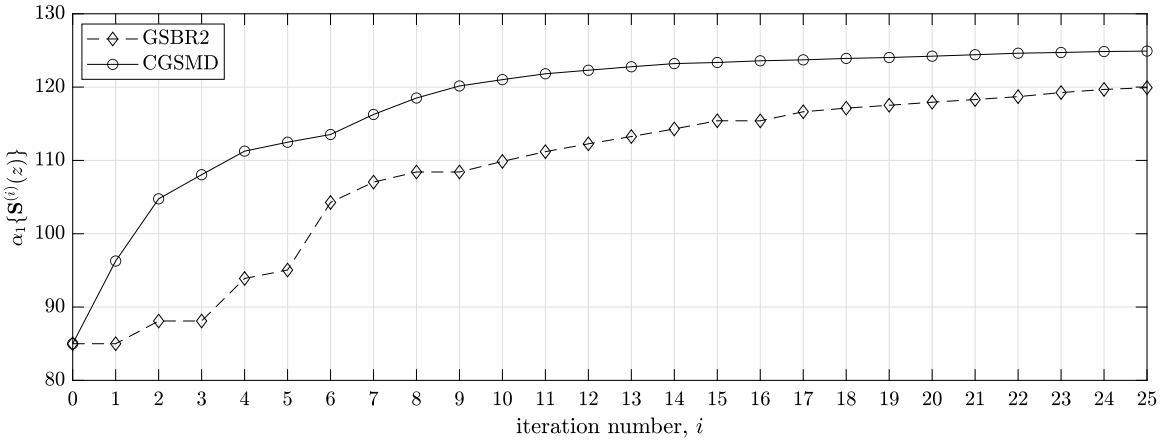

Fig. 4. Zero-lag coefficient matrix diagonal energy, i.e. $\alpha_1\{\boldsymbol{S}^{(i)}(z)\}$, versus iteration number for example $\boldsymbol{A}(z)$ in (26).

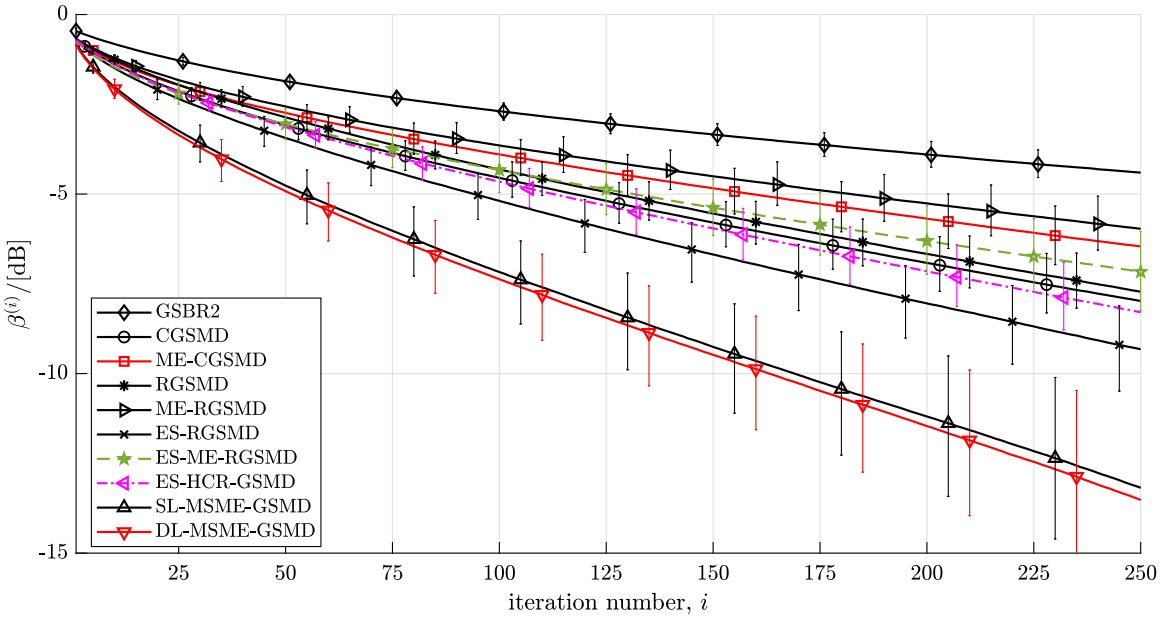

Fig. 5. $\beta^{(i)}$ versus iteration number i for the GSBR2 and proposed algorithms over the entire ensemble. HCR-GSMD shows almost identical profile to CGSMD, and so is omitted for clarity. Solid lines show the median; error bars denote the 10th and 90th percentiles.

6.4. Ensemble results

6.4.1. Off-diagonal energy versus iteration index

A key merit of an effective iterative PSVD algorithm is its ability to progressively transfer more energy onto the diagonal with each iteration. This behavior can be quantified by monitoring the reduction of the off-diagonal energy. To this end, we plot $\beta^{(i)}$ as defined in (24), where more negative values indicate stronger diagonalization, against the iteration index for the entire ensemble (see Fig. 5). The ensemble median is shown in bold, while the 10th and 90th percentiles are depicted as error bars. The results clearly demonstrate that all GSMD-based variants outperform GSBR2 in terms of reducing off-diagonal energy across iterations. For GSBR2, the final $\beta^{(i)}$ after 250 iterations remains above -5 dB, whereas all proposed variants reach below -5 dB before the 175th iteration. Among them, DL-MSME-GSMD and SL-MSME-GSMD exhibit the fastest convergence, achieving $\beta^{(I)} < -12.5$ dB, as they transfer the largest fraction of energy per iteration. By contrast, ME-RGSMD and ME-CGSMD converge more slowly, though they still consistently outperform GSBR2. Overall, these findings suggest that GSBR2 would require at least 50–100 additional iterations to reduce the off-diagonal energy to the level achieved even by the slowest of the proposed GSMD variants.

6.4.2. Computational complexity

To evaluate the computational complexity of the proposed algorithms, we present two sets of results. In the first set, we examine the relationship between diagonalization performance and execution time by plotting the ensemble median of $\beta^{(i)}$ against the ensemble median of the elapsed system time T_i, as shown in Fig. 6. This curve is obtained by measuring both $\beta^{(i)}$ and T_i at each iteration, which allows us to view diagonalization progress as a function of the total time required to complete i iterations. The profile in Fig. 6 shows that all proposed variants achieve significantly better diagonalization than GSBR2 within the same time budget. For example, at $t = 0.1$ s, the proposed algorithms outperform GSBR2 by at least 1.65 dB in metric $\beta^{(i)}$. While some variants incur higher computational cost per iteration, they still achieve substantially greater diagonalization within the same execution time, making them computationally more efficient overall. Among them, RGSMD and ES-RGSMD provide the fastest diagonalization with respect to execution time, while ME-CGSMD and ME-RGSMD are the slowest, though they still perform better than GSBR2.

The second set of results, summarized in Table 1, reports the overall execution time, including both the mean and the standard deviation across the entire ensemble. With the exception of SL-MSME-GSMD and

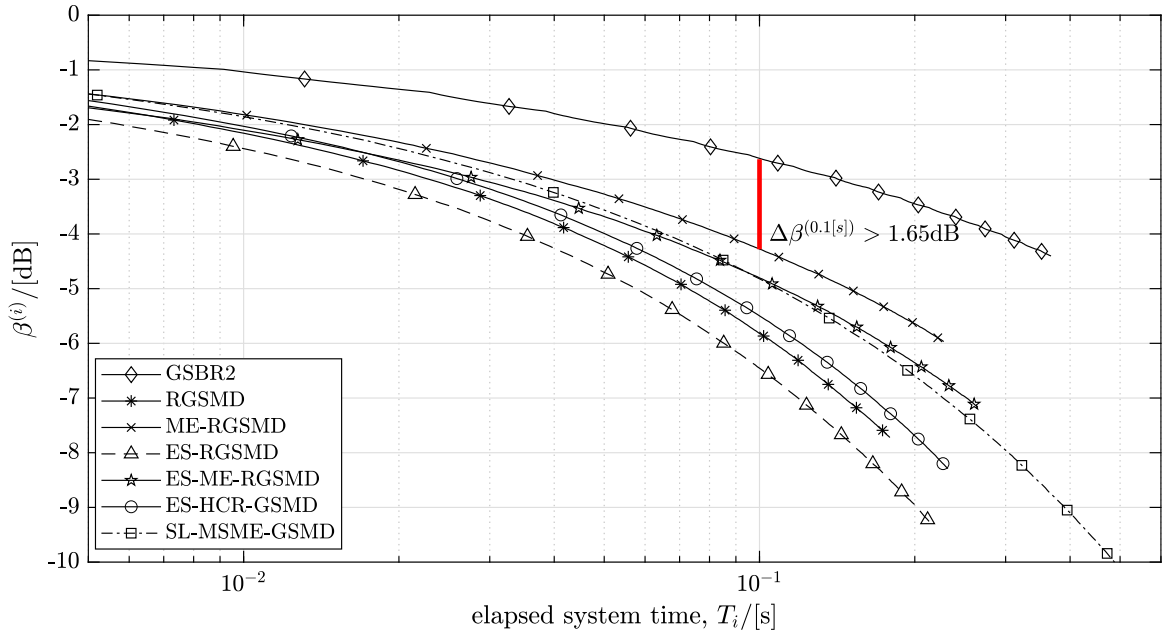

Fig. 6. Normalized off-diagonal energy versus the ensemble median of the elapsed system time. HCR–GSMD and ES-HCR-GSMD show behavior identical to CGSMD. DL-MSME-GSMD is indistinguishable from SL-MSME-GSMD, and the ME-CGSMD profile is close to ME-RGSMD, so these traces are omitted for clarity.

Table 1
Comparison of PSVD algorithms over an ensemble of 2000 randomized 6×4-dimension polynomial matrices.

Algorithms/Metrics	Computation Time [s]	Norm. Recons. Error [ζ]	Norm. Error of $\hat{\sigma}(z)$ [ξ]	$\mathcal{O}\{\hat{\boldsymbol{U}}(z)\}$	$\mathcal{O}\{\hat{\boldsymbol{V}}(z)\}$
GSBR2	0.3857 ± 0.0878	0.139 ± 0.019	$(7.00 \pm 2.22) \times 10^{-5}$	92 ± 16	87 ± 17
CGSMD	0.2410 ± 0.0476	0.066 ± 0.013	$(0.33 \pm 0.15) \times 10^{-5}$	84 ± 10	82 ± 12
ME-CGSMD	0.2620 ± 0.0535	0.093 ± 0.022	$(0.78 \pm 0.35) \times 10^{-5}$	93 ± 19	89 ± 21
RGSMD	0.1892 ± 0.0398	0.057 ± 0.011	$(0.42 \pm 0.18) \times 10^{-5}$	81 ± 11	70 ± 10
ME-RGSMD	0.2381 ± 0.0524	0.096 ± 0.023	$(1.13 \pm 0.51) \times 10^{-5}$	97 ± 23	85 ± 21
ES-RGSMD	0.2282 ± 0.0470	0.055 ± 0.011	$(0.19 \pm 0.11) \times 10^{-5}$	96 ± 29	75 ± 10
ES-ME-RGSMD	0.2772 ± 0.0573	0.082 ± 0.021	$(0.49 \pm 0.28) \times 10^{-5}$	116 ± 46	88 ± 24
HCR-GSMD	0.2412 ± 0.0496	0.065 ± 0.013	$(0.32 \pm 0.15) \times 10^{-5}$	86 ± 10	80 ± 12
ES-HCR-GSMD	0.2439 ± 0.0508	0.064 ± 0.013	$(0.29 \pm 0.14) \times 10^{-5}$	86 ± 10	81 ± 13
SL-MSME-GSMD	0.8997 ± 0.1997	0.079 ± 0.019	$(0.49 \pm 0.28) \times 10^{-6}$	121 ± 33	113 ± 34
DL-MSME-GSMD	0.9726 ± 0.2143	0.079 ± 0.019	$(0.48 \pm 0.27) \times 10^{-6}$	122 ± 35	114 ± 35

DL-MSME-GSMD, all proposed variants complete $I = 250$ iterations in roughly two-thirds of the time required by GSBR2. RGSMD is the fastest overall, followed by ES-RGSMD.

6.4.3. Normalized error metrics

Since none of the algorithms reached the target off-diagonal energy threshold ($\epsilon = 10^{-6}$) within the allowed number of iterations, the normalized error metric ξ at $I = 250$ is expected to be smaller for algorithms that achieve stronger diagonalization. As shown in Table 1, the multiple-shift variants are clearly the most effective, achieving the lowest normalization errors for the singular values. In contrast, GSBR2's performance is the worst: its singular value error is about six times larger than that of ME-RGSMD, the least effective of the proposed methods. The same pattern holds for the normalized reconstruction error metric, which indirectly shows the accuracy of both the estimated eigenvectors and the singular values, where GSBR2 again ranks last, while RGSMD and ES-RGSMD emerge as the most accurate.

6.4.4. Paraunitary matrices polynomial orders

The polynomial orders of $\hat{\boldsymbol{U}}(z)$ and $\hat{\boldsymbol{V}}(z)$, which indicate the implementation complexity of filter banks in hardware for signal processing applications, are also reported in Table 1. RGSMD produces eigenvectors with the lowest polynomial orders, followed by CGSMD, both of which outperform GSBR2 in this regard. In contrast, the multiple-shift variants result in the largest polynomial orders among all algorithms.

6.5. Scalability and limitation

6.5.1. Small to moderately large spatial dimension simulation

For an $M \times M$ polynomial matrix, the overall complexity is cubic in M, primarily due to the ordinary matrix SVD computation and the subsequent application of unitary transformations to $\boldsymbol{S}^{(i-\frac{1}{2})}[\tau]$ for every lag τ in the ith iteration. The temporal order L of the polynomial matrix has little impact, apart from the fact that complexity scales linearly with it. Therefore, we focus on comparing the execution time of the proposed algorithm against that of the GSBR2 algorithm for moderately high spatial dimensions, in order to assess scalability with respect to M in the Matlab. For this, we generate an ensemble of 200 polynomial matrices of size $M \times M$, with $M \in \{4, 8, 12, 16, 20\}$. The algorithmic parameters are kept the same as in the earlier ensemble test. The extra-shift (ES) variants are not applicable in this case due to the square spatial dimension. Since SL-MSME-GSMD and DL-MSME-GSMD behave identically under these conditions, the set of considered algorithms reduces to CGSMD, RGSMD, HCR-GSMD, and their maximum-element variants (ME-CGSMD and ME-RGSMD). However, as ME-CGSMD and ME-RGSMD are less effective, we restrict our simulations to four algorithms: CGSMD, RGSMD, HCR-GSMD, and MSME-GSMD (with SL and DL being identical here).

The ensemble results are shown in Fig. 7, with computation time in (a) and the normalized error of the estimated singular values, i.e., ξ, in (b) plotted against the spatial dimension. Apart from MSME-GSMD, the average execution time of the CGSMD, RGSMD, and HCR-GSMD is

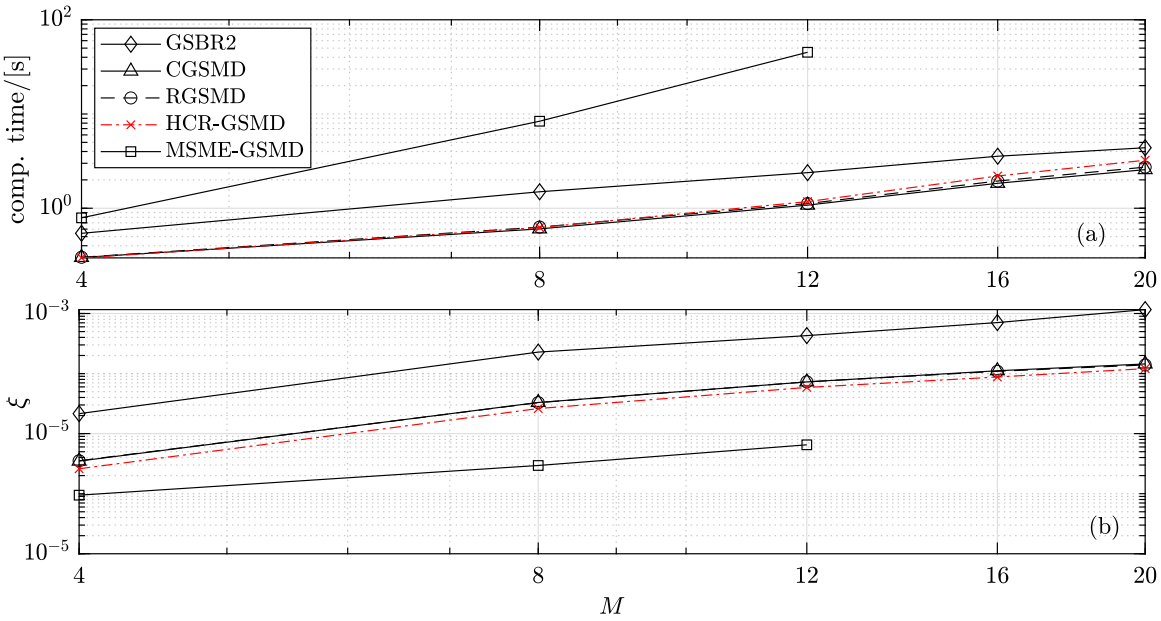

Fig. 7. (a) Execution time, and (b) normalized error for estimated singular values for higher spatial dimension polynomial matrices (MSME-GSMD is simulated only for $M = 4, 8, 12$ as the execution exceeded a minute).

consistently lower than that of the GSBR2 algorithm for all values of M. Among CGSMD, RGSMD, and HCR-GSMD, there is no appreciable difference in execution time; they perform identically in terms of computational complexity. The complexity of MSME-GSMD, however, grows rapidly with M, such that for $M > 12$, a diagonalization on average requires approximately one minute or more. Note that the reported computation times correspond to a total of 250 iterations. The superior performance in computational sense is further supported by the fact that normalized singular value error metric ξ for the proposed algorithms is consistently lower than that of GSBR2.

6.5.2. Higher or extreme spatial dimension polynomial matrices

For higher-dimensional spatial matrices, the divide-and-conquer approach [41,42], initially adopted for the SMD algorithm in [43], can also be applied to the GSMD algorithms. Since the core principles of SMD and GSMD are closely related, similar improvements in execution time can be expected for GSMD, as reported for SMD in [43], where a reduction of up to 66% was achieved for parahermitian matrices of spatial dimensions $M = 20, 40$.

6.5.3. Polynomial order growth limitation

The paraunitary and parahermitian polynomial matrix truncation methods used for SBR2 and SMD are not particularly suitable for GSMD or GSBR2 [8,9,39,40]. The reason is that, unlike SMD and SBR2—where columns are delayed and rows are advanced, and the partially diagonalized matrix remains parahermitian, simplifying the truncation procedure—in the proposed method, as well as in GSBR2, columns can be either advanced or delayed, and the partially diagonalized matrix does not exhibit any symmetry. Therefore, a new truncation strategy is required, which can significantly improve the accuracy of singular vectors and reduce the reconstruction error.

6.6. Radar plot for conclusive comparison

For a direct performance comparison, a radar chart, illustrated in Fig. 8, is constructed considering five different metrics: convergence speed with respect to iterations (CSI), computational efficiency (CE), convergence speed w.r.t time (CST), accuracy speed w.r.t time (AST), and accuracy speed w.r.t iterations (ASI). These metrics are computed as follows:

$$CSI = \frac{-4 - \text{average}\{\beta^{(I)}\}}{I}, CE = \frac{I}{\text{average}\{T_I\}}, CST = \frac{-4 - \text{average}\{\beta^{(I)}\}}{\text{average}\{T_I\}}$$
$$AST = \frac{\text{average}\{\xi\} - 10^{-4}}{\text{average}\{T_I\}}, ASI = \frac{\text{average}\{\xi\} - 10^{-4}}{I}, \quad (27)$$

and thereafter normalized w.r.t. maximum. A higher values of these metrics indicate better performance. These metrics allow users to select the most appropriate algorithm based on the specific application requirements. For time-sensitive applications, where fast convergence and high accuracy in minimal time are essential, algorithms with better AST and CST metrics are preferable. In contrast, in less time-critical environments, iteration-based metrics like CSI and ASI might be more relevant.

The MSME variants transfer the most energy onto the diagonal of the zero-lag coefficient matrix in every iteration, making them optimal according to the CSI metric but the worst in terms of CE. Conversely, RGSMD has the lowest execution time per iteration, yielding the best CE among all variants, although it is outperformed by ES-RGSMD in CST. This observation aligns with the results shown in Fig. 6. The AST metric, which evaluates the accuracy of the estimated singular values relative to the elapsed system time, shows RGSMD as the best performer than the ES-RGSMD over the constructed ensemble, while the MSME variants show poor performance, requiring more time to estimate accurate singular values. However, when the number of iterations is considered instead of the execution time, MSME variants achieve the most accurate singular value estimates with fewer iterations, as reflected in a high ASI metric. Finally, GSBR2 exhibits the poorest performance in all aspects excepts that its average execution time per iteration is lower than the MSME variants. Nevertheless, due to its poor performance in the other metrics, GSBR2 demonstrates the worst overall performance, as evidenced by the smallest enclosed area in its radar chart.

In summary, the CGSMD, RGSMD, ES-RGSMD, ES-HCR-GSMD, and HCR-GSMD, which is not shown due similar performance to ES-HCR-GSMD, variants demonstrate comparable performance to one another, with the ES-RGSMD emerging as the leading method. In scenarios

where computation time is of lesser concern, such as in an offline diagonalization of a polynomial matrix, the MSME-GSMD variants are preferable, as they achieve superior diagonalization with the fewest iterations.

7. Application of the GSMD algorithm to broadband MIMO channel equalization

To demonstrate the practical utility of the proposed PSVD algorithm, we apply it to decouple a broadband MIMO channel. Broadband MIMO systems, characterized by multi-path propagation and frequency-selective fading, and therefore can be modeled via tap-delay-line, i.e., a polynomial matrix with sufficient time-lags. With broadband MIMO channel $\boldsymbol{C}[n], n = 0, \ldots, (L-1)$ with L number of taps, $s_m[n], i = 1, \ldots, M$ denoting the source signals, $x_\ell[n], \ell = 1, \ldots, N$ the received signals and $v_\ell[n], \ell = 1, \ldots, N$ the additive Gaussian noise, in z-domain we have

$$\begin{bmatrix} x_1(z) \\ \vdots \\ x_N(z) \end{bmatrix} = \begin{bmatrix} c_{11}(z) & \cdots & c_{1M}(z) \\ \vdots & \cdots & \vdots \\ c_{N1}(z) & \cdots & c_{NM}(z) \end{bmatrix} \begin{bmatrix} s_1(z) \\ \vdots \\ s_M(z) \end{bmatrix} + \begin{bmatrix} v_1(z) \\ \vdots \\ v_M(z) \end{bmatrix} \tag{28}$$

An SVD can only diagonalize $\boldsymbol{C}(z)$ for one particular value of z but not $\forall\ z$, whereas a PSVD can diagonalize $\boldsymbol{C}(z)$ for all values of z into $\boldsymbol{U}(z)\boldsymbol{\Sigma}(z)\boldsymbol{V}^{\mathrm{P}}(z)$, where $\boldsymbol{U}(z)$ and $\boldsymbol{V}(z)$ are para-unitary filter banks, and $\boldsymbol{\Sigma}(z)$ is a diagonal polynomial matrix containing decoupled frequency selective channel gains. The transmitter precodes the source signals $\boldsymbol{s}(z)$ using $\boldsymbol{V}(z)$ and receiver applies $\boldsymbol{U}^{\mathrm{P}}(z)$ to the received signal

$$\boldsymbol{U}^{\mathrm{P}}(z)\boldsymbol{x}(z) = \boldsymbol{U}^{\mathrm{P}}(z)\boldsymbol{U}(z)\boldsymbol{\Sigma}(z)\boldsymbol{V}^{\mathrm{P}}(z)\boldsymbol{V}(z)\boldsymbol{s}(z) + \boldsymbol{U}^{\mathrm{P}}(z)\boldsymbol{v}(z) \tag{29}$$

eliminating channel interference such that

$$\boldsymbol{y}(z) = \boldsymbol{\Sigma}(z)\boldsymbol{s}(z) + \boldsymbol{v}'(z) \tag{30}$$

where $\boldsymbol{y}(z) = \boldsymbol{U}^{\mathrm{P}}(z)\boldsymbol{x}(z)$ and $\boldsymbol{v}'(z) = \boldsymbol{U}^{\mathrm{P}}(z)\boldsymbol{v}(z)$. Due to paraunitary filter banks both in transmitter and receiver, neither the transmit power is increased, nor the channel noise. The resulting ISI in $\boldsymbol{\Sigma}(z)$ can be removed zero-forcing equalizer. By concentrating energy along the diagonal of $\boldsymbol{\Sigma}(z)$, the PSVD effectively isolates the independent subchannels, enabling simpler receiver design.

A representative example of a 3×3 broadband MIMO channel $C(z)$ of polynomial order 20 is considered to illustrate the performance of the proposed algorithm. The absolute weights at different time lags are depicted in Fig. 9. For the decoupling application, channel state information (CSI) is assumed to be known at both the transmitter and receiver. The precoder and decoder paraunitary matrices are derived using the MSME-RGSMD-DL algorithm, employing a maximum of 400 iterations with $\epsilon = 10^{-6}$ and $\mu_{\mathrm{PU}} = \mu_{\mathrm{PH}} = 10^{-6}$. The elements of the resulting diagonalized matrix $\hat{\boldsymbol{\Sigma}}[n]$ are shown in Fig. 10, demonstrating that the considered convolutive MIMO channel is effectively decoupled. Upon evaluating $\hat{\boldsymbol{\Sigma}}(z)$ on the unit circle, i.e., $z = \mathrm{e}^{\mathrm{j}\Omega}$, we can compare the estimated singular values against the ground truth as shown in Fig. 11. This shows that estimated singular values closely matches the ground-truth and therefore, suggests accurate decoupling of the MIMO channel. Moreover, it also highlights the proposed GSMD algorithm's capability to reliably estimate singular values across the frequency spectrum. This accuracy ensures that the channel matrix is well-conditioned for equalization, likely leading to enhanced interference suppression and improved signal recovery. Such characteristics are essential in broadband MIMO systems, where spectral nulls and poorly estimated singular values can degrade overall performance.

The sources, drawn from three independent BPSK constellations, each of length 10^5, are filtered through $\boldsymbol{V}(z)$ and subsequently convolutively mixed with $\boldsymbol{C}(z)$. Additive white Gaussian noise (AWGN) is then introduced to achieve various levels of signal-to-noise ratio (SNR), ranging from 0 to 10 dB in increments of 2 dB. The received signal is

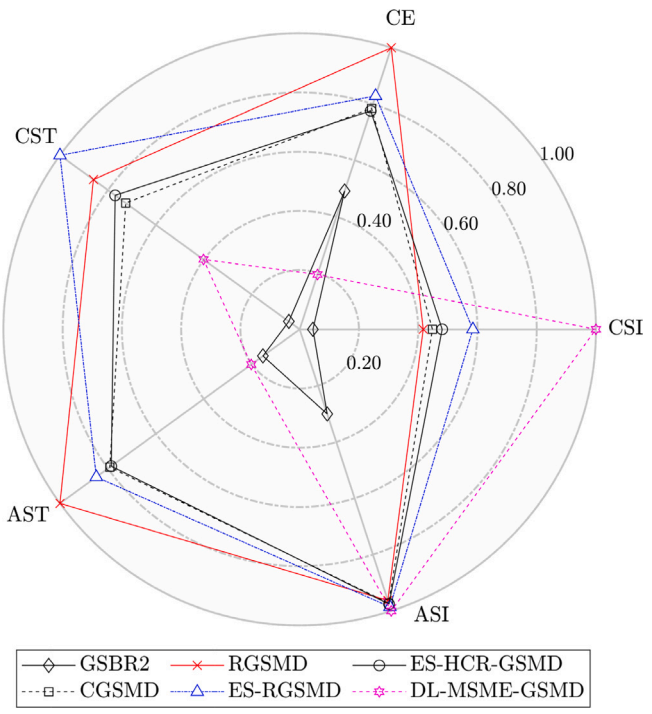

Fig. 8. Radar chart comparing selected top-performing GSMD variants against the baseline GSBR2 algorithm across five performance metrics defined in (27). ME-CGSMD, ME-RGSMD, and ES-ME-RGSMD are omitted due to poor performance, while performances of HCR-GSMD and SL-MSME-GSMD closely match those of ES-HCR-GSMD and DL-MSME-GSMD, respectively, and so are omitted for clarity.

further processed by a paraunitary matrix $\boldsymbol{U}^{\mathrm{P}}(z)$. After the channel is fully decoupled into three single-input single-output (SISO) channels, equalization is performed using a maximum likelihood sequence estimation (MLSE) equalizer, which employs the Viterbi algorithm [44]. The bit error rate (BER) is computed for multiple trials for each source, and the average BER is plotted as a function of the SNR, as illustrated in Fig. 12. The results indicate that the BER for the largest mode or singular value is the lowest, whereas the BER for the smallest singular value remains the highest across all selected SNR values. This simulation demonstrates the efficacy of the proposed algorithm in accurately decoupling convolutive broadband MIMO channels and achieving reliable communication. Unlike the PQRD-based approach [13,15], which requires back-substitution to reconstruct SISO channels, the GSMD algorithm directly decouples the MIMO channel into a series of SISO channels, effectively mitigating error propagation.

The computational efficiency of the GSMD algorithm renders it particularly well-suited for real-time broadband MIMO systems. While GSMD significantly improves accurate singular value estimation and, consequently, BER performance, its efficiency ensures practical feasibility for time-sensitive applications. Unlike approaches such as GSBR2, which often entail significantly higher computational overhead and require more iterations, GSMD achieves effective diagonalization with fewer iterations and controlled growth of the polynomial order. This balance between performance and complexity positions GSMD as a compelling choice for advanced communication systems, including 5G and mmWave MIMO. Furthermore, its computational efficiency translates directly into reduced latency, a critical requirement for time-sensitive applications where rapid channel state adaptation is essential. The algorithm's suitability for implementation on reconfigurable FPGAs further enhances its applicability in real-time operations, making it an excellent candidate for low-latency, high-throughput communication scenarios (see Figs. 7 and 8).

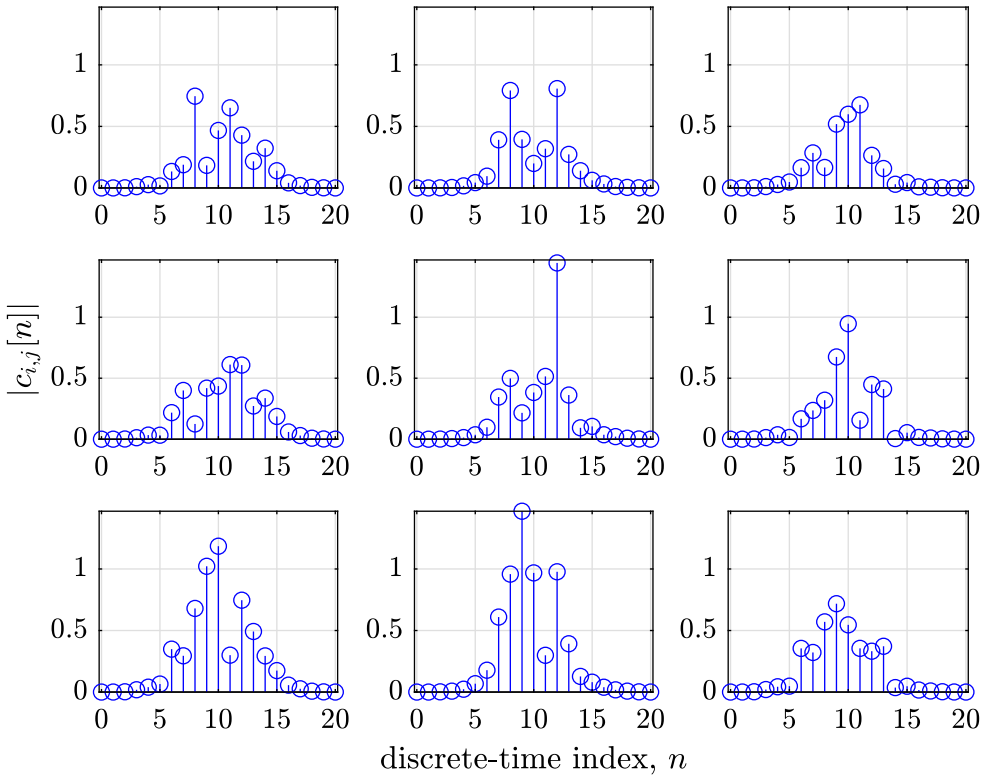

Fig. 9. A 3×3 broadband MIMO channel $\mathbf{C}[n]$ with 21 taps, i.e., $n = 0, \ldots, 20$.

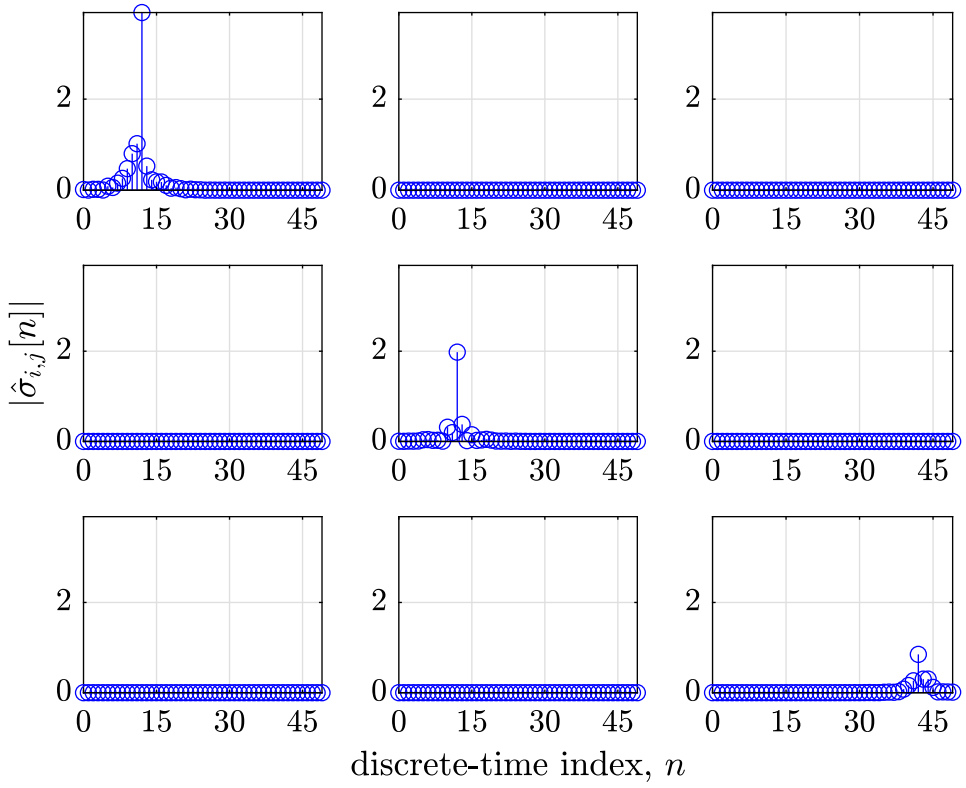

Fig. 10. PSVD results for the example 3×3 broadband MIMO channel, illustrating $\hat{\boldsymbol{\Sigma}}[n]$, produced via MSME-GSMD-DL version of the GSMD algorithm, versus n.

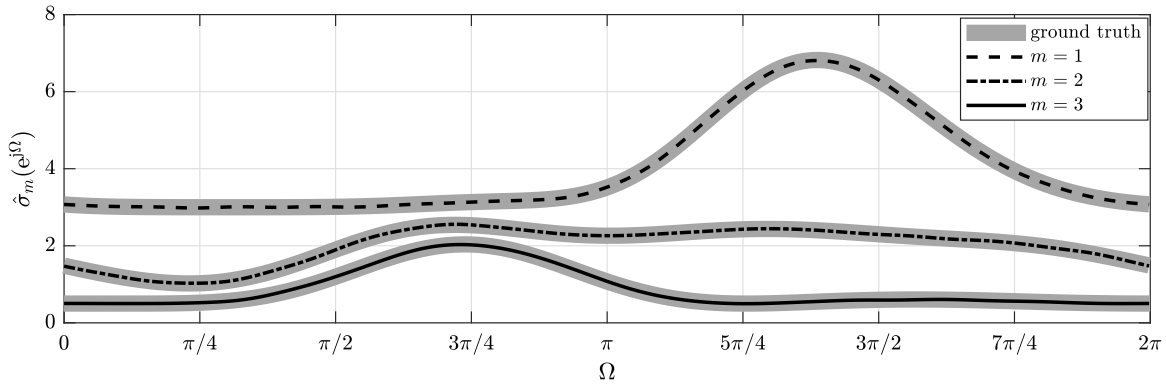

Fig. 11. GSMD estimated singular values, evaluated on the unit circle, for the MIMO channel matrix example with ground-truth underlaid in gray.

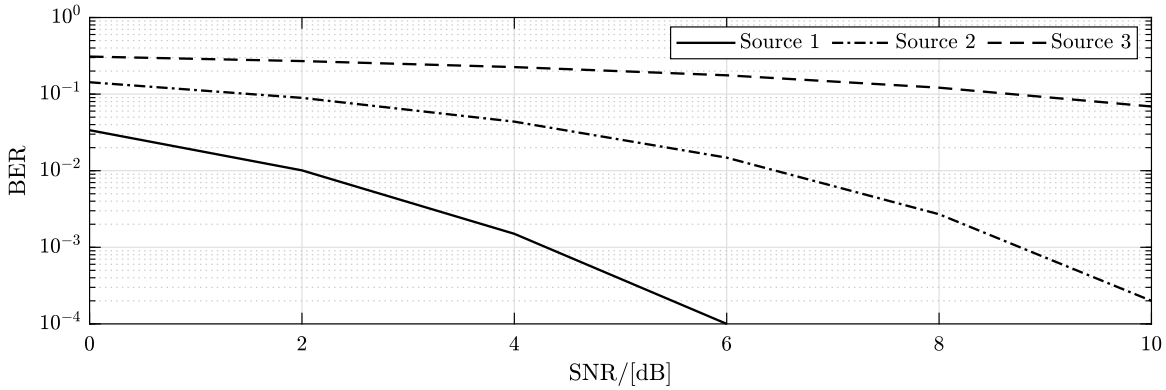

Fig. 12. Average BERs for BPSK sources estimated through an MLSE equalizer for SISO channel decoupled from a convolutive MIMO channel using a PSVD algorithm at multiple SNR values.

8. Conclusion

In this article, we have proposed sequential matrix diagonalization (SMD) based algorithms for computing the singular value decomposition (SVD) of polynomial matrices. The proposed generalized SMD (GSMD) framework encompasses multiple algorithmic variants, each offering distinct trade-offs between convergence speed and computational complexity. The algorithms shift off-diagonal elements onto the zero-lag coefficient matrix of the polynomial matrix using time-shift operations, which are subsequently transferred onto the diagonal via ordinary SVD. A formal proof of convergence has been provided and holds for all proposed variants.

Comprehensive ensemble tests on 2000 randomly generated polynomial matrices demonstrated that the proposed algorithms significantly outperform the generalized SBR2 (GSBR2) algorithm, which represents the sole existing dedicated polynomial SVD (PSVD) algorithm in the literature. Key performance advantages include: (i) achieving 50 to 100 iterations faster convergence for equivalent diagonalization levels, (ii) reducing computation time by approximately one-third while maintaining superior accuracy, and (iii) producing lower polynomial orders in the decomposition factors for several non-MSME variants. Notably, the proposed algorithms can eliminate an entire column and row in each iteration, in contrast to GSBR2, which eliminates a maximum of one element per iteration. This fundamental difference in energy transfer efficiency explains why the proposed GSMD algorithms demonstrate greater relative performance gains over GSBR2 than those previously reported for SMD over SBR2.

Among the proposed variants, ES-RGSMD emerged as the best overall performer, balancing computational efficiency with diagonalization accuracy, while the MSME-GSMD variants achieved the highest diagonalization quality when computational time is not critical. The algorithms are scalable to spatial dimensions up to 20×20, with a divide-and-conquer approach enabling extension to higher dimensions for the non-MSME variants. Our findings validate the effectiveness of the proposed GSMD algorithm not only for broadband MIMO channel equalization but also for wider applications including polynomial generalized SVD beamforming for frequency selective MIMO [45], broadband blind source separation [46], and polynomial root MUSIC [47]. The direct decoupling of MIMO channels into SISO channels without back-substitution represents a significant practical advantage over PQRD-based approaches.

Future work will explore integrating GSMD into adaptive communication systems, including 5G, vehicular-to-everything (V2X), and mmWave networks, which require real-time adjustments to channel conditions under dynamic environments. Additionally, GSMD can be directly applied to the polynomial Procrustes problem for approximating the best paraunitary matrix to a given polynomial matrix [48,49], with applications in determining lossless transfer function matrices for acoustic systems and artificial reverberation of multichannel audio [50, 51].

CRediT authorship contribution statement

Faizan A. Khattak: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Soydan Redif:** Writing – review & editing, Visualization, Validation, Formal analysis,

Conceptualization. **Mohammed Bakhit:** Writing – review & editing, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors would like to express their gratitude to Ian K. Proudler and Stephan Weiss from the University of Strathclyde for their valuable discussions and insights related to this work. We also thank John McWhirter from Cardiff University for reviewing our draft.

Data availability

No data was used for the research described in the article.

References

[1] N. Bose, S. Mitra, Generalized inverse of polynomial matrices, IEEE Trans. Autom. Control 23 (3) (1978) 491–493.
[2] V.W. Neo, S. Redif, J.G. McWhirter, J. Pestana, I.K. Proudler, S. Weiss, P.A. Naylor, Polynomial eigenvalue decomposition for multichannel broadband signal process, IEEE Signal Process. Mag. 40 (7) (2023) 18–37.
[3] J. Foster, J. McWhirter, S. Lambotharan, I. Proudler, M. Davies, J. Chambers, Polynomial matrix QR decomposition for the decoding of frequency selective multiple-input multiple-output communication channels, IET Signal Process. 6 (7) (2012) 704–712.
[4] D. Cescato, H. Bölcskei, QR decomposition of laurent polynomial matrices sampled on the unit circle, IEEE Trans. Inform. Theory 56 (9) (2010) 4754–4761.
[5] V.W. Neo, C. Evers, P.A. Naylor, Enhancement of noisy reverberant speech using polynomial matrix eigenvalue decomposition, IEEE/ACM Trans. Audio Speech Lang. Process. 29 (2021) 3255–3266.
[6] M.A. Alrmah, S. Weiss, S. Lambotharan, An extension of the MUSIC algorithm to broadband scenarios using a polynomial eigenvalue decomposition, in: 2011 19th Eur. Signal Process. Conf, 2011, pp. 629–633.
[7] S. Redif, S. Weiss, J.G. McWhirter, Relevance of polynomial matrix decompositions to broadband blind signal separation, Signal Process. 134 (2017) 76–86.
[8] J.G. McWhirter, P.D. Baxter, T. Cooper, S. Redif, J. Foster, An EVD algorithm for para-hermitian polynomial matrices, IEEE Trans. Signal Process. 55 (5) (2007) 2158–2169.
[9] S. Redif, S. Weiss, J.G. McWhirter, Sequential matrix diagonalization algorithms for polynomial EVD of parahermitian matrices, IEEE Trans. Signal Process. 63 (1) (2015) 81–89.
[10] S. Weiss, I.K. Proudler, F.K. Coutts, Eigenvalue decomposition of a parahermitian matrix: Extraction of analytic eigenvalues, IEEE Trans. Signal Process. 69 (2021) 722–737.
[11] S. Weiss, I.K. Proudler, F.K. Coutts, F.A. Khattak, Eigenvalue decomposition of a parahermitian matrix: Extraction of analytic eigenvectors, IEEE Trans. Signal Process. 71 (2023) 1642–1656.
[12] M. Tohidian, H. Amindavar, A.M. Reza, A DFT-based approximate eigenvalue and singular value decomposition of polynomial matrices, EURASIP J. Adv. Signal Process. 2013 (1) (2013) 93.
[13] J.A. Foster, J.G. McWhirter, M.R. Davies, J.A. Chambers, An algorithm for calculating the QR and singular value decompositions of polynomial matrices, IEEE Trans. Signal Process. 58 (3) (2010) 1263–1274.
[14] D. Cescato, H. Bölcskei, Algorithms for interpolation-based QR decomposition in MIMO-OFDM systems, IEEE Trans. Signal Process. 59 (4) (2011) 1719–1733.
[15] D. Hassan, S. Redif, S. Lambotharan, I.K. Proudler, Sequential polynomial QR decomposition and decoding of frequency selective MIMO channels, in: 2018 26th Eur. Signal Process. Conf., EUSIPCO, 2018, pp. 460–464, (ISSN: 2076-1465).
[16] J.G. McWhirter, An algorithm for polynomial matrix SVD based on generalised kogbetliantz transformations, in: 2010 18th Eur. Signal Process. Conf., 2010, pp. 457–461.
[17] Z. Wang, A. Sandmann, J.G. McWhirter, A. Ahrens, Multiple shift SBR2 algorithm for calculating the SVD of broadband optical MIMO systems, in: 2016 39th Int. Conf. on Telecommunications and Signal Process, TSP, 2016, pp. 433–436.
[18] M. Bakhit, F.A. Khattak, I.K. Proudler, S. Weiss, G. Rice, Compact order polynomial singular value decomposition of a matrix of analytic functions, in: 2023 IEEE Int. Workshop on Computational Adv. in Multi-Sensor Adaptive Process, CAMSAP, 2023, accepted.
[19] F.A. Khattak, I.K. Proudler, S. Weiss, Extension of power method to para-hermitian matrices: Polynomial power method, in: 2023 31st Eur. Signal Process. Conf., EUSIPCO, 2023, pp. 1564–1568.
[20] G.E. Forsythe, P. Henrici, The cyclic Jacobi method for computing the principal values of a complex matrix, Trans. Am. Math. Soc. 94 (1) (1960) 1–23.
[21] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., The Johns Hopkins University Press, 1996.
[22] S. Weiss, I.K. Proudler, G. Barbarino, J. Pestana, J.G. McWhirter, On properties and structure of the analytic singular value decomposition, IEEE Trans. Signal Process. 72 (2024) 2260–2275.
[23] F.A. Khattak, M. Bakhit, I.K. Proudler, S. Weiss, Extraction of analytic singular values of a polynomial matrix, in: 2024 32nd European Signal Processing Conference, EUSIPCO, 2024, pp. 1297–1301.
[24] F.A. Khattak, S. Weiss, I.K. Proudler, J.G. McWhirter, Space-time covariance matrix estimation: Loss of algebraic multiplicities of eigenvalues, in: 2022 56th Asilomar Conf. on Signals, Systems, and Computers, 2022, pp. 975–979, (ISSN: 2576-2303).
[25] M. Bakhit, F.A. Khattak, I.K. Proudler, S. Weiss, Impact of estimation errors of a matrix of transfer functions onto its analytic singular values and their potential algorithmic extraction, in: 2024 IEEE High Performance Extreme Computing Virtual Conf, accepted.
[26] F.A. Khattak, I.K. Proudler, J.G. McWhirter, S. Weiss, Generalised sequential matrix diagonalisation for the SVD of polynomial matrices, in: Sensor Signal Process. for Defence Conf., SSPD, 2023, pp. 1–5.
[27] P. Vaidyanathan, Theory of optimal orthonormal subband coders, IEEE Trans. Signal Process. 46 (6) (1998) 1528–1543.
[28] P.P. Vaidyanathan, Multirate Systems and Filter Banks, Prentice-Hall, Inc., USA, 1993.
[29] S.L. Campbell, Control problem structure and the numerical solution of linear singular systems, Math. Control Signals Systems 1 (1) (1988) 73–87.
[30] V. Balakrishnan, S. Boyd, On computing the worst-case peak gain of linear systems, Systems Control Lett. 19 (4) (1992) 265–269.
[31] A. Bunse-Gerstner, R. Byers, V. Mehrmann, N.K. Nichols, Numerical computation of an analytic singular value decomposition of a matrix valued function, Numer. Math. 60 (1) (1991) 1–39.
[32] V. Mehrmann, W. Rath, Numerical methods for the computation of analytic singular value decompositions, Electron. Trans. Numer. Anal. 1 (1993) 72–88.
[33] V. Janovský, D. Janovská, K. Tanabe, Computing the analytic singular value decomposition via a pathfollowing, in: Numerical Mathematics and Advanced Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 954–962.
[34] S. Weiss, J. Pestana, I.K. Proudler, On the existence and uniqueness of the eigenvalue decomposition of a parahermitian matrix, IEEE Trans. Signal Process. 66 (10) (2018) 2659–2672.
[35] J. Foster, J. McWhirter, J. Chambers, A polynomial matrix QR decomposition with application to MIMO channel equalisation, in: 2007 Conf. Record of the Forty-First Asilomar Conf. on Signals, Systems and Computers, 2007, pp. 1379–1383, (ISSN: 1058-6393).
[36] Z. Wang, J.G. McWhirter, J. Corr, S. Weiss, Multiple shift second order sequential best rotation algorithm for polynomial matrix EVD, in: 2015 23rd Eur. Signal Process. Conf., EUSIPCO, 2015, pp. 844–848.
[37] P. Vaidyanathan, Theory of optimal orthonormal subband coders, IEEE Trans. Signal Process. 46 (6) (1998) 1528–1543.
[38] J. McWhirter, Z. Wang, A novel insight to the sbr2 algorithm for diagonalising para-hermitian matrices, in: IMA Conf. on Mathematics in Signal Process 2016, Institute of Mathematics, 2016.
[39] J. Corr, K. Thompson, S. Weiss, I.K. Proudler, J.G. McWhirter, Row-shift corrected truncation of paraunitary matrices for PEVD algorithms, in: 2015 23rd Eur. Signal Process. Conf., EUSIPCO, 2015, pp. 849–853.
[40] J. Corr, K. Thompson, S. Weiss, I. Proudler, J. McWhirter, Shortening of paraunitary matrices obtained by polynomial eigenvalue decomposition algorithms, in: 2015 Sensor Signal Process. for Defence, SSPD, 2015, pp. 1–5.
[41] J.J. Cuppen, A divide and conquer method for the symmetric tridiagonal eigenproblem, Numer. Math. 36 (2) (1980) 177–195.
[42] D. Gill, E. Tadmor, An o(n^2) method for computing the eigensystem of n*n symmetric tridiagonal matrices by the divide and conquer approach, SIAM J. Sci. Stat. Comput. 11 (1) (1990) 161–173.
[43] F. Coutts, J. Corr, K. Thompson, I. Proudler, S. Weiss, Divide-and-conquer sequential matrix diagonalisation for parahermitian matrices, in: 2017 Sensor Signal Process. for Defence Conf., SSPD, 2017, pp. 1–5.
[44] A. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Trans. Inform. Theory 13 (2) (1967) 260–269.
[45] D. Hassan, S. Redif, J.G. McWhirter, S. Lambotharan, Polynomial GSVD beamforming for two-user frequency-selective mimo channels, IEEE Trans. Signal Process. 69 (2021) 948–959.
[46] S. Redif, J. Pestana, I.K. Proudler, Analysis of broadband GEVD-based blind source separation, in: ICASSP 2019-2019 IEEE Int. Conf. on Acoustics, Speech and Signal Process., ICASSP, 2019, pp. 8028–8032.
[47] W. Coventry, C. Clemente, J. Soraghan, Enhancing polynomial MUSIC algorithm for coherent broadband sources through spatial smoothing, in: 25th Eur. Signal Process. Conf., EUSIPCO, 2017, pp. 2448–2452.

[48] J.C. Gower, G.B. Dijksterhuis, Procrustes Problems, Oxford University Press, 2004.
[49] S. Weiss, S.J. Schlecht, O. Das, E. De Sena, Polynomial procrustes problem: Paraunitary approximation of matrices of analytic functions, in: EUSIPCO, Helsinki, Finland, 2023.
[50] S.J. Schlecht, E.A.P. Habets, On lossless feedback delay networks, IEEE Trans. Signal Process. 65 (6) (2017) 1554–1564.
[51] S.J. Schlecht, Allpass feedback delay networks, IEEE Trans. Signal Process. 69 (2021) 1028–1038.