Algorithmic Enhancements to Polynomial Matrix Factorisations

Fraser Kenneth Coutts

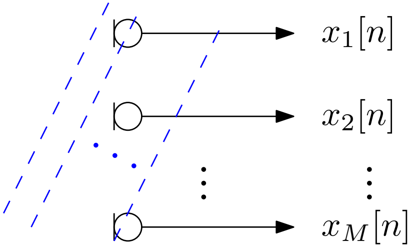
Material adapted from PhD Thesis of the same name.

Thesis available online via EURASIP's library: `https://tinyurl.com/couttsphd`.

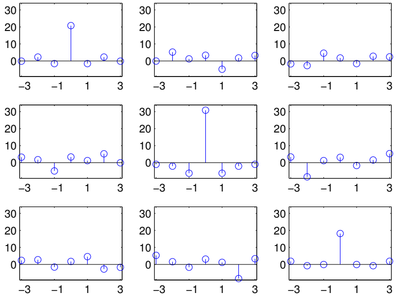
27th June 2019

Presentation Overview

Summary of Background



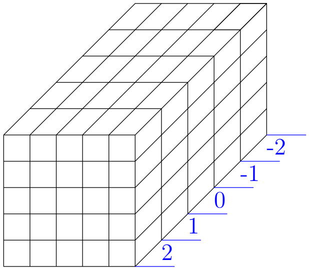
- Space-time covariance matrix: $\mathbf{R}[\tau] = \mathcal{E}\{\mathbf{x}[n]\mathbf{x}^{\mathrm{H}}[n-\tau]\}$
- Matrix of auto- & cross-correlation sequences
- Symmetry $\mathbf{R}[\tau] = \mathbf{R}^{\mathrm{H}}[-\tau]$



- Cross-spectral density $\boldsymbol{R}(z) = \sum_{\tau} \mathbf{R}[\tau] z^{-\tau}$ is a polynomial matrix.
- Parahermitian: $\boldsymbol{R}^{\mathrm{P}}(z) = \boldsymbol{R}^{\mathrm{H}}(1/z^*) = \boldsymbol{R}(z)$

Summary of Background

- $\boldsymbol{R}(z)$ is a matrix with (Laurent) polynomial entries or alternatively a polynomial with matrix-valued coefficients.
- Can be interpreted as a three-dimensional matrix.



$$\boldsymbol{R}(z) = \begin{bmatrix} R_{11}(z) & R_{12}(z) & \cdots & R_{1M}(z) \\ R_{21}(z) & R_{22}(z) & & \vdots \\ \vdots & & \ddots & \vdots \\ R_{M1}(z) & \cdots & \cdots & R_{MM}(z) \end{bmatrix}$$

$$= \sum_{\tau} \mathbf{R}[\tau] z^{-\tau}$$

Summary of Background

- Eigenvalue decomposition (EVD) of a Hermitian covariance matrix $\mathbf{R}$ offers optimality for many narrowband problems:

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{H}} \qquad \mathbf{\Lambda} \text{ is diagonal}$$
$$\mathbf{Q} \text{ is unitary, i.e. } \mathbf{Q}\mathbf{Q}^{\mathrm{H}} = \mathbf{I}$$

- The polynomial matrix EVD (PEVD) is an extension to parahermitian matrices $\boldsymbol{R}(z)$:

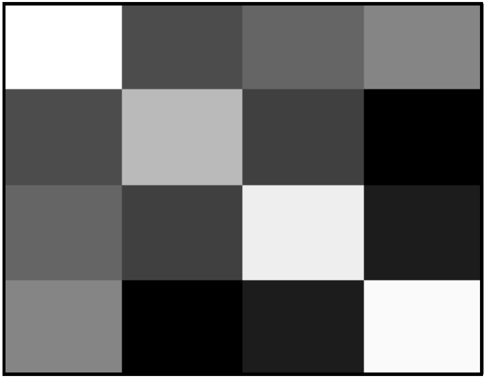
$$\boldsymbol{R}(z) \approx \boldsymbol{Q}(z)\mathbf{\Lambda}(z)\boldsymbol{Q}^{\mathrm{P}}(z) \qquad \mathbf{\Lambda}(z) \text{ is diagonal}$$
$$\boldsymbol{Q}(z) \text{ is paraunitary, } \boldsymbol{Q}(z)\boldsymbol{Q}^{\mathrm{P}}(z) = \mathbf{I}$$

- Diagonalisation of $\mathbf{\Lambda}(z)$ is important for, e.g., decoupling of broadband MIMO systems.
- Polynomial subspace decomposition (i.e. $\boldsymbol{Q}(z)$) used in, e.g., broadband AoA estimation and beamforming.

How to Factorise a Polynomial Matrix?

- Iteratively minimise off-diagonal energy in $\boldsymbol{R}(z)$.

$$\boldsymbol{R}(z) \rightarrow \boldsymbol{\Lambda}(z)$$

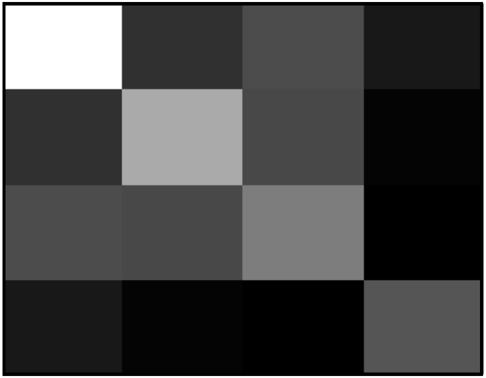


Energy in 'flattened' parahermitian matrix. White $\Rightarrow$ high energy.

How to Factorise a Polynomial Matrix?

- Iteratively minimise off-diagonal energy in $\boldsymbol{R}(z)$.

$$\boldsymbol{R}(z) \rightarrow \boldsymbol{\Lambda}(z)$$

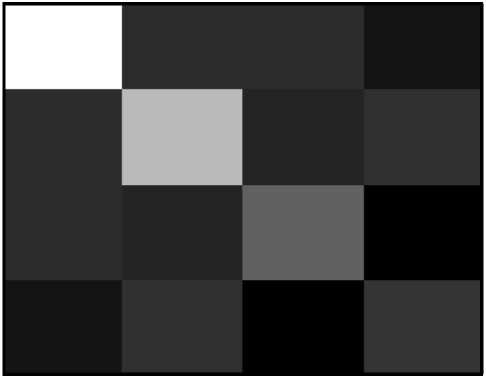


Energy in ‘flattened’ parahermitian matrix. White $\Rightarrow$ high energy.

How to Factorise a Polynomial Matrix?

- Iteratively minimise off-diagonal energy in $\boldsymbol{R}(z)$.

$$\boldsymbol{R}(z) \rightarrow \boldsymbol{\Lambda}(z)$$

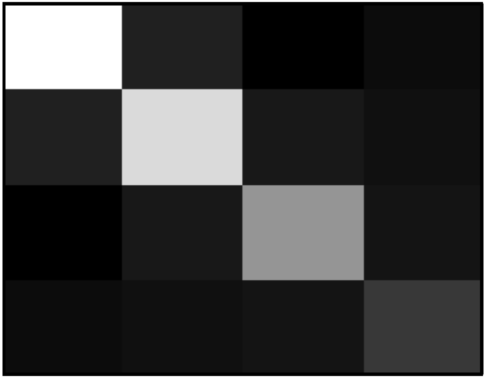


Energy in ‘flattened’ parahermitian matrix. White $\Rightarrow$ high energy.

How to Factorise a Polynomial Matrix?

- Iteratively minimise off-diagonal energy in $\boldsymbol{R}(z)$.

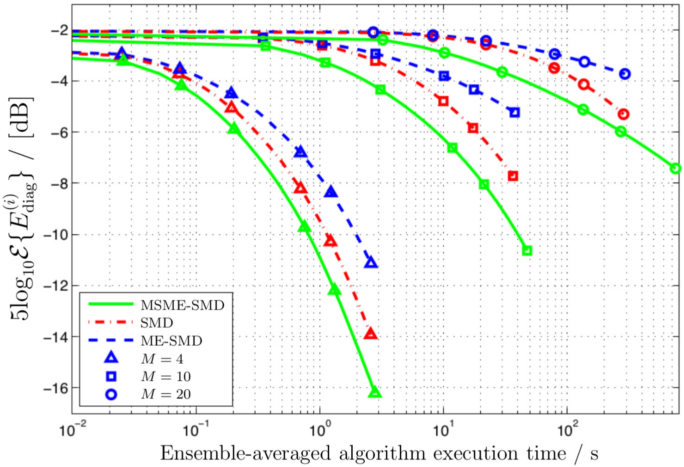
$$\boldsymbol{R}(z) \rightarrow \boldsymbol{\Lambda}(z)$$



Energy in 'flattened' parahermitian matrix. White $\Rightarrow$ high energy.

How to Factorise a Polynomial Matrix?

- Iteratively minimise off-diagonal energy in $\boldsymbol{R}(z)$.
- Key performance metric: off-diagonal energy vs execution time.

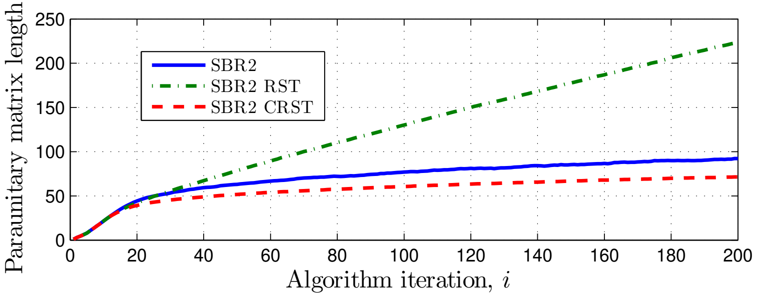


How to Factorise a Polynomial Matrix?

- Several iterative PEVD algorithms successfully minimise off-diagonal energy.
- Most important:
  - Second order sequential best rotation (SBR2);
  - Sequential matrix diagonalisation (SMD).

Requirement for Truncation in Iterative PEVD Algorithms

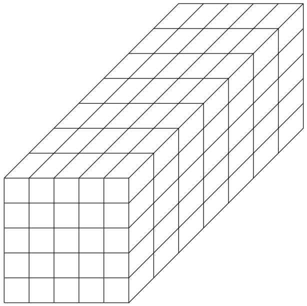
- The ‘shift’ operations performed in SBR2 and SMD lead to an increase in the order of the paraunitary matrix $\boldsymbol{Q}(z)$ at **each iteration**.
- If the order of $\boldsymbol{Q}(z)$ is not restricted in some way, memory usage and computational complexity will scale linearly with the order.



State-of-the-Art Truncation

[Ta *et al.*, ICIS&SP TSP 2007]

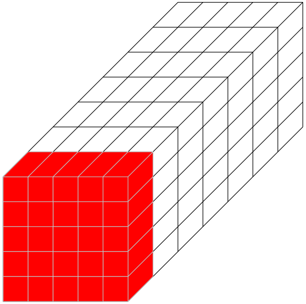
- Used to reduce the polynomial order of the paraunitary $\boldsymbol{Q}(z)$.



State-of-the-Art Truncation

[Ta *et al.*, ICIS&SP TSP 2007]

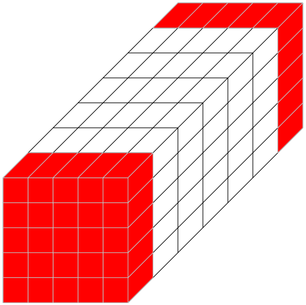
- Used to reduce the polynomial order of the paraunitary $\boldsymbol{Q}(z)$.
- Successively removes the outermost lag of $\boldsymbol{Q}(z)$ with the lowest energy.



State-of-the-Art Truncation

[Ta *et al.*, ICIS&SP TSP 2007]

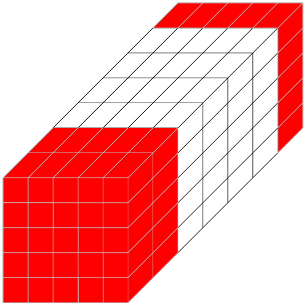
- Used to reduce the polynomial order of the paraunitary $\boldsymbol{Q}(z)$.
- Successively removes the outermost lag of $\boldsymbol{Q}(z)$ with the lowest energy.
- Lags can be removed from either side of $\boldsymbol{Q}(z)$.



State-of-the-Art Truncation

[Ta *et al.*, ICIS&SP TSP 2007]

- Used to reduce the polynomial order of the paraunitary $\boldsymbol{Q}(z)$.
- Successively removes the outermost lag of $\boldsymbol{Q}(z)$ with the lowest energy.
- Lags can be removed from either side of $\boldsymbol{Q}(z)$.

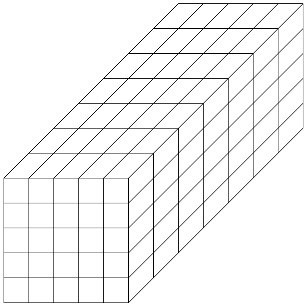


PEVD Ambiguity

- The paraunitary matrix in the PEVD is not unique.
- The same diagonalised parahermitian matrix, $\mathbf{\Lambda}(z)$, can be obtained using different paraunitary $\boldsymbol{F}(z) = \boldsymbol{Q}^{\mathrm{P}}(z)$,
$$\boldsymbol{R}(z) \approx \boldsymbol{F}^{\mathrm{P}}(z)\mathbf{\Lambda}(z)\boldsymbol{F}(z) = \hat{\boldsymbol{F}}^{\mathrm{P}}(z)\mathbf{\Lambda}(z)\hat{\boldsymbol{F}}(z) \quad .$$
- Note that polynomial eigenvectors are in the **rows** of $\boldsymbol{F}(z)$.
- Using a modifying matrix, $\mathbf{\Gamma}(z)$, we can go from $\boldsymbol{F}(z)$ to $\hat{\boldsymbol{F}}(z)$,
$$\hat{\boldsymbol{F}}(z) = \mathbf{\Gamma}(z)\boldsymbol{F}(z) \quad .$$
- For the row-shift truncation we simply use,
$$\mathbf{\Gamma}(z) = \mathrm{diag}\{z^{-\tau_1} \; z^{-\tau_2} \; \ldots \; z^{-\tau_M}\} \quad ,$$
which individually delays or advances each of the M rows of $\boldsymbol{F}(z)$.

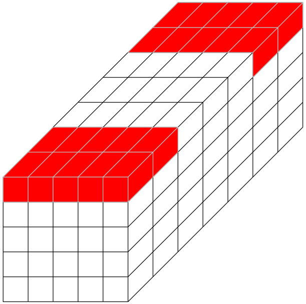
Row-Shift Truncation

- Used to reduce the polynomial order of the paraunitary $\boldsymbol{F}(z)$.



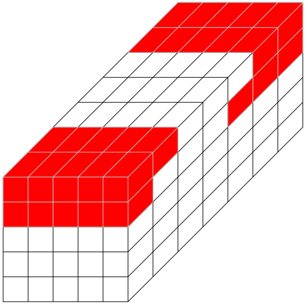
Row-Shift Truncation

- Used to reduce the polynomial order of the paraunitary $\boldsymbol{F}(z)$.
- Works the same as the state-of-the-art but applied to each row individually.



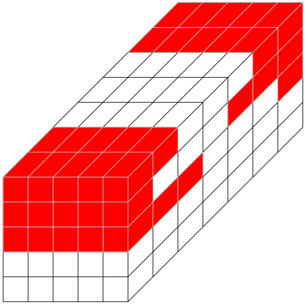
Row-Shift Truncation

- Used to reduce the polynomial order of the paraunitary $\boldsymbol{F}(z)$.
- Works the same as the state-of-the-art but applied to each row individually.
- Each row can be truncated by a different amount.



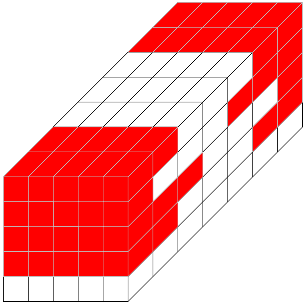
Row-Shift Truncation

- Used to reduce the polynomial order of the paraunitary $\boldsymbol{F}(z)$.
- Works the same as the state-of-the-art but applied to each row individually.
- Each row can be truncated by a different amount.



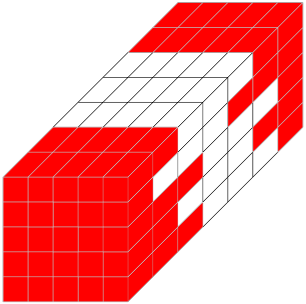
Row-Shift Truncation

- Used to reduce the polynomial order of the paraunitary $\boldsymbol{F}(z)$.
- Works the same as the state-of-the-art but applied to each row individually.
- Each row can be truncated by a different amount.



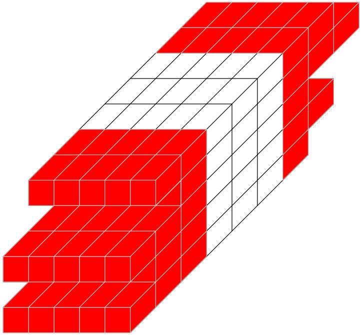
Row-Shift Truncation

- Used to reduce the polynomial order of the paraunitary $\boldsymbol{F}(z)$.
- Works the same as the state-of-the-art but applied to each row individually.
- Each row can be truncated by a different amount.



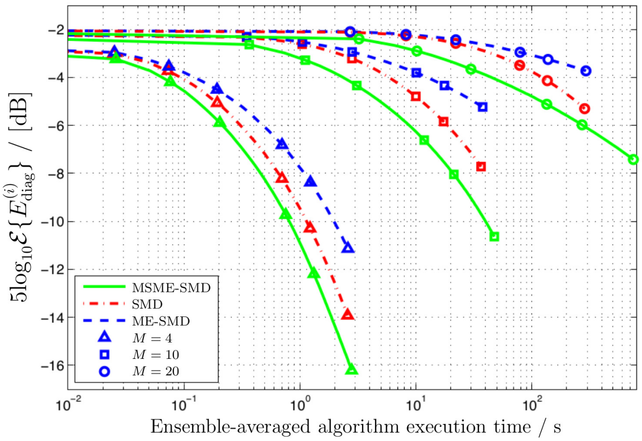
Row-Shift Truncation

- Used to reduce the polynomial order of the paraunitary $\boldsymbol{F}(z)$.
- Works the same as the state-of-the-art but applied to each row individually.
- Each row can be truncated by a different amount.
- Final step aligns rows using $\boldsymbol{\Gamma}(z)$.



Motivation for my Research

- Existing iterative algorithms are slow to converge and not feasible to implement — worse for large spatial dimension M.



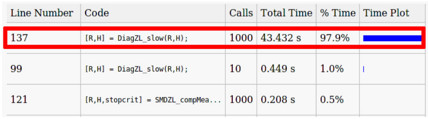
Motivation for my Research

- Existing iterative algorithms are slow to converge and not feasible to implement — worse for large spatial dimension M.
- Aims:
  - Develop techniques to lower complexity (and memory requirements) of existing methods.
  - Design novel, fast algorithms with improved scalability.
  - Investigate frequency-based methods to compute PEVD.

Algorithmic Improvements

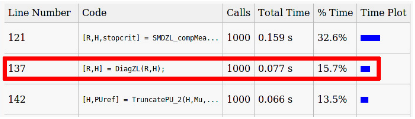
- First step: code profiling and optimisation.

Before optimisation

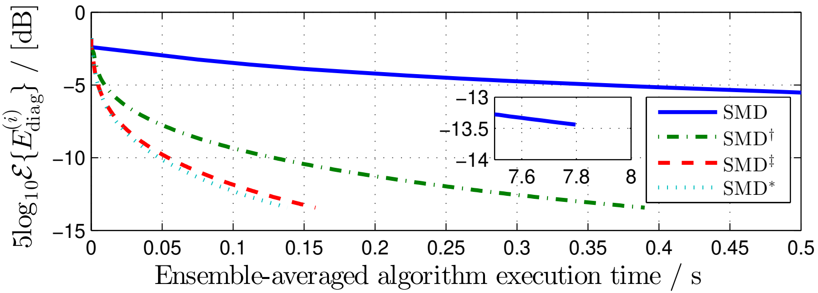


Line Number	Code	Calls	Total Time	% Time	Time Plot
137	`[R,H] = DiagZL_slow(R,H);`	1000	43.432 s	97.9%	▬
99	`[R,H] = DiagZL_slow(R,H);`	10	0.449 s	1.0%	▏
121	`[R,H,stopcrit] = SMDZL_compMea...`	1000	0.208 s	0.5%	

After optimisation



Line Number	Code	Calls	Total Time	% Time	Time Plot
121	`[R,H,stopcrit] = SMDZL_compMea...`	1000	0.159 s	32.6%	▬
137	`[R,H] = DiagZL(R,H);`	1000	0.077 s	15.7%	▬
142	`[H,PUref] = TruncatePU_2(H,Mu,...`	1000	0.066 s	13.5%	▬

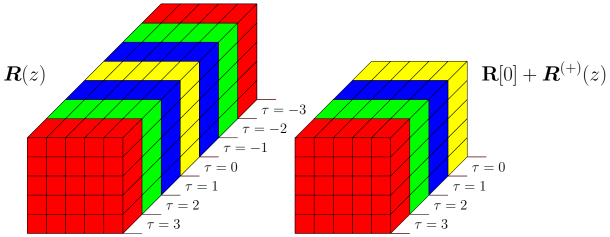


Reduced Parahermitian Matrix Representation

- By segmenting a parahermitian matrix $\boldsymbol{R}(z)$, we can write

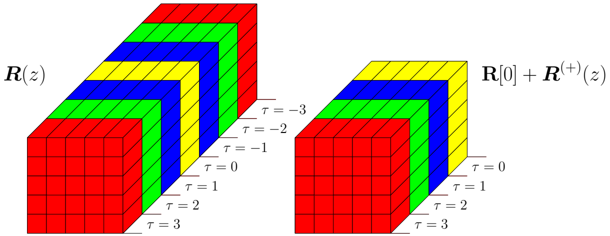
$$\boldsymbol{R}(z) = \boldsymbol{R}^{(-)}(z) + \mathbf{R}[0] + \boldsymbol{R}^{(+)}(z) \ .$$

- $\mathbf{R}[0]$ is the zero lag matrix, $\boldsymbol{R}^{(+)}(z)$ contains terms for positive lag elements only, and $\boldsymbol{R}^{(-)}(z) = \boldsymbol{R}^{(+),\mathrm{P}}(z)$.
- It is therefore sufficient to record a 'half-matrix' version of $\boldsymbol{R}(z)$.



Reduced Parahermitian Matrix Representation

- Columns beyond lag zero ($\tau = 0$) have been discarded.
- Modifications have to be made to the search and shift stages.
- Both columns **and** rows in the reduced matrix are searched.
- A ‘cyclic shift’ approach is employed.

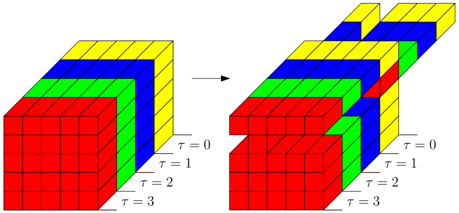


Reduced Parahermitian Matrix Representation

- An example of the shift operation is depicted for the case of $\boldsymbol{R}(z) : \mathbb{C} \rightarrow \mathbb{C}^{5\times 5}$ with parameters $k^{(i)} = 2$ and $\tau^{(i)} = -3$.

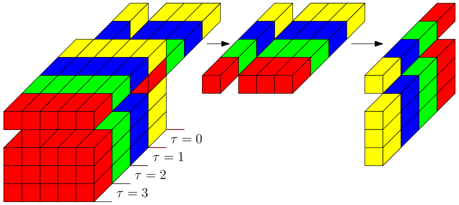
Reduced Parahermitian Matrix Representation

- An example of the shift operation is depicted for the case of $\boldsymbol{R}(z) : \mathbb{C} \rightarrow \mathbb{C}^{5\times 5}$ with parameters $k^{(i)} = 2$ and $\tau^{(i)} = -3$.
- Shifting procedure:
  1. The row is shifted.



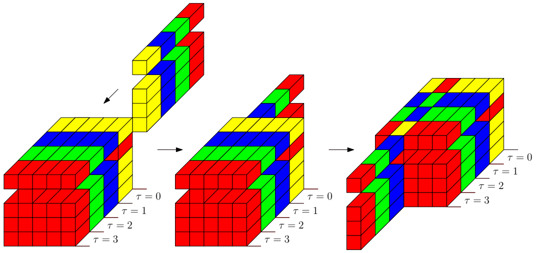
Reduced Parahermitian Matrix Representation

- An example of the shift operation is depicted for the case of $\boldsymbol{R}(z) : \mathbb{C} \rightarrow \mathbb{C}^{5\times 5}$ with parameters $k^{(i)} = 2$ and $\tau^{(i)} = -3$.
- Shifting procedure:
  1. The row is shifted.
  2. Non-diagonal elements in the $k^{(i)}$th row past lag zero are extracted and parahermitian transposed.



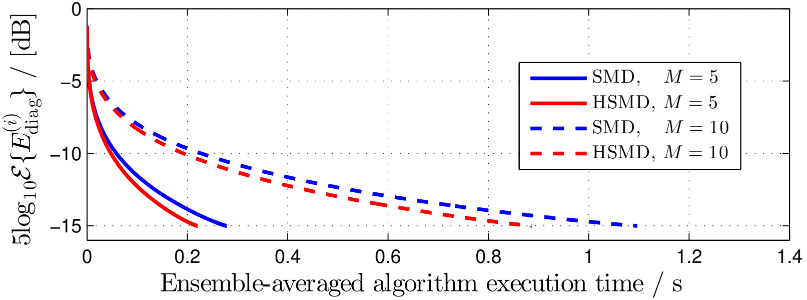
Reduced Parahermitian Matrix Representation

- An example of the shift operation is depicted for the case of $\boldsymbol{R}(z) : \mathbb{C} \rightarrow \mathbb{C}^{5\times5}$ with parameters $k^{(i)} = 2$ and $\tau^{(i)} = -3$.
- Shifting procedure:
  1. The row is shifted.
  2. Non-diagonal elements in the $k^{(i)}$th row past lag zero are extracted and parahermitian transposed.
  3. These elements are appended to the $k^{(i)}$th column at lag zero and this column is shifted in the opposite direction.



Reduced Parahermitian Matrix Representation

- Off-diagonal energy versus algorithm execution time for standard SMD algorithm and 'half-matrix' SMD (HSMD) implementation.
- M is spatial dimension of parahermitian matrix, e.g., number of array elements.



Reduced Parahermitian Matrix Representation

- Approximate resource requirements of standard (SMD) and proposed (HSMD) representations of an $M \times M \times (2N^{(i)} + 1)$ parahermitian matrix at the ith iteration.

Method	Complexity	Storage	Memory Moves
SMD	$4N^{(i)}M^3$	$2N^{(i)}M^2$	$4N^{(i-1)}M$
HSMD	$2N^{(i)}M^3$	$N^{(i)}M^2$	$2N^{(i-1)}M$

- All resource requirements are approximately halved using the proposed approach.

Restricted Update SMD (RU-SMD)

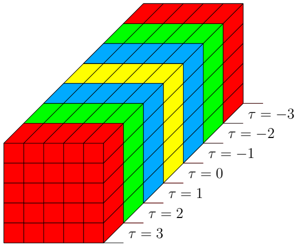
- Restricting the search space of iterative PEVD algorithms to a subset of lags around lag zero of a parahermitian matrix can bring performance gains with little impact on algorithm convergence.
  [Corr *et al.*, ISP 2015 & Coutts *et al.*, Asilomar SSC 2016]
- The restricted update SMD (RU-SMD) algorithm restricts the search space of the SMD algorithm, but also restricts the portion of the parahermitian matrix that is updated at each iteration.
- The update step of SMD is its most computationally costly operation; thus, restricting the complexity of this step is useful.
  [Redif *et al.*, IEEE TSP 2015]
- Aim of RU-SMD is to be **less expensive** and **faster** than SMD.

RU-SMD Overview

- The RU-SMD algorithm computes the PEVD of a parahermitian matrix $\boldsymbol{R}(z)$ over $i = 0 \ldots I$ iterations.
- RU-SMD has two main steps:
  1. Restricted update: iteratively diagonalise parahermitian matrix while monotonically contracting the search and update space.
  2. Matrix regeneration: regenerate parahermitian matrix when search and update space has maximally contracted.
- Steps 1 and 2 are repeated for index $\alpha = 0 \ldots \beta$.
- The space restriction in 1 limits the number of search operations, and reduces the computations required to update the parahermitian matrix.

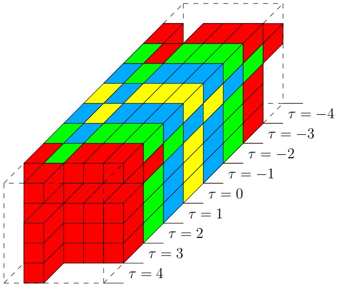
Restricted Update Approach

- Matrix $\boldsymbol{S}^{(i-1)}(z) = \boldsymbol{R}_{(\alpha)}(z) : \mathbb{C} \rightarrow \mathbb{C}^{5\times 5}$ with maximum lag $\tau_{\max}^{(i)} = 3$ input to restricted update step. Note: $\boldsymbol{R}_{(0)}(z) = \boldsymbol{R}(z)$.
- Treat lags $|\tau| > \tau_{\max}^{(i)}$ as invalid.



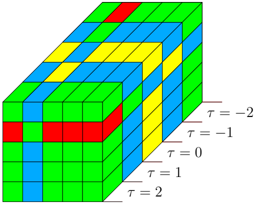
Restricted Update Approach

- Shifting of $k^{(i)}$th row and column energy to lag zero from lags $\pm\tau^{(i)}$ ($k^{(i)} = 2$, $\tau^{(i)} = -1$).
- Invalid values from lags $|\tau| > \tau_{\max}^{(i)}$ are shifted towards lag zero.



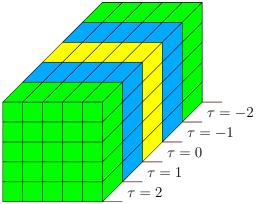
Restricted Update Approach

- Valid central matrix with maximum lag $(\tau^{(i)}_{\max} - |\tau^{(i)}|) = 2$, $\boldsymbol{S}^{(i)\prime\prime}(z)$, is extracted.
- Lags $|\tau| > (\tau^{(i)}_{\max} - |\tau^{(i)}|)$ are invalid.



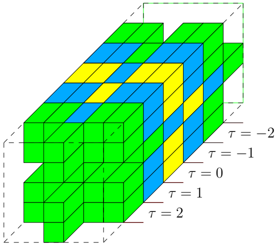
Restricted Update Approach

- Update step: $\boldsymbol{S}^{(i)}(z) = \mathbf{Q}^{(i)}\boldsymbol{S}^{(i)\prime\prime}(z)\mathbf{Q}^{(i),\mathrm{H}}$. Matrix $\mathbf{Q}^{(i)}$ is obtained from EVD of lag zero.
- Lags $|\tau| > (\tau_{\max}^{(i)} - |\tau^{(i)}|)$ are invalid.



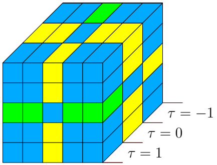
Restricted Update Approach

- Shifting of $k^{(i+1)}$th row and column energy to lag zero from lags $\pm\tau^{(i+1)}$ ($k^{(i+1)} = 3$, $\tau^{(i+1)} = -1$).
- Invalid values from lags $|\tau| > (\tau_{\max}^{(i)} - |\tau^{(i)}|)$ are shifted towards lag zero.



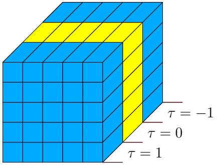
Restricted Update Approach

- Valid central matrix with maximum lag $(\tau_{\max}^{(i)} - |\tau^{(i)}| - |\tau^{(i+1)}|) = 1$, $\boldsymbol{S}^{(i+1)\prime\prime}(z)$, is extracted.
- Lags $|\tau| > (\tau_{\max}^{(i)} - |\tau^{(i)}| - |\tau^{(i+1)}|)$ are invalid.



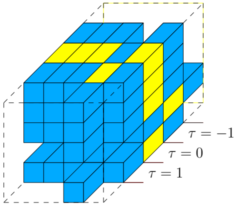
Restricted Update Approach

- Update step: $\boldsymbol{S}^{(i+1)}(z) = \mathbf{Q}^{(i+1)}\boldsymbol{S}^{(i+1)\prime\prime}(z)\mathbf{Q}^{(i+1),\mathrm{H}}$. Matrix $\mathbf{Q}^{(i+1)}$ is obtained from EVD of lag zero.
- Lags $|\tau| > (\tau_{\max}^{(i)} - |\tau^{(i)}| - |\tau^{(i+1)}|)$ are invalid.



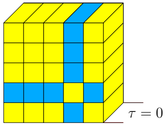
Restricted Update Approach

- Shifting of $k^{(i+2)}$th row and column energy to lag zero from lags $\pm\tau^{(i+2)}$ ($k^{(i+2)} = 4$, $\tau^{(i+2)} = -1$).
- Invalid values from lags $|\tau| > (\tau_{\max}^{(i)} - |\tau^{(i)}| - |\tau^{(i+1)}|)$ are shifted towards lag zero.



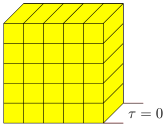
Restricted Update Approach

- Valid central matrix with maximum lag $(\tau_{\max}^{(i)} - |\tau^{(i)}| - |\tau^{(i+1)}| - |\tau^{(i+2)}|) = 0$, $\boldsymbol{S}^{(i+2)\prime\prime}(z)$, is extracted.
- Lags $|\tau| > (\tau_{\max}^{(i)} - |\tau^{(i)}| - |\tau^{(i+1)}| - |\tau^{(i+2)}|)$ are invalid.



Restricted Update Approach

- Update step: $\boldsymbol{S}^{(i+2)}(z) = \mathbf{Q}^{(i+2)}\boldsymbol{S}^{(i+2)\prime\prime}(z)\mathbf{Q}^{(i+2),\mathrm{H}}$. Matrix $\mathbf{Q}^{(i+2)}$ is obtained from EVD of lag zero.
- Only zero lag remains: matrix must now be regenerated.



Matrix Regeneration

- Paraunitary matrix $\boldsymbol{F}_{(\alpha)}(z)$ is the product of all shift and update operations performed during the αth execution of the restricted update step.
- This step has iteratively reduced the off-diagonal energy in input parahermitian matrix $\boldsymbol{R}_{(\alpha)}(z)$.
- $\boldsymbol{R}_{(\alpha+1)}(z) = \boldsymbol{F}_{(\alpha)}(z)\boldsymbol{R}_{(\alpha)}(z)\boldsymbol{F}^{\mathrm{P}}_{(\alpha)}(z)$ is the regenerated matrix, and is more diagonal than $\boldsymbol{R}_{(\alpha)}(z)$.

Matrix Regeneration

- Following regeneration of the parahermitian matrix, a matrix $\boldsymbol{G}_{(\alpha+1)}(z)$ is also updated, which is a product of the paraunitary matrices generated for indices $0\ldots\alpha$:

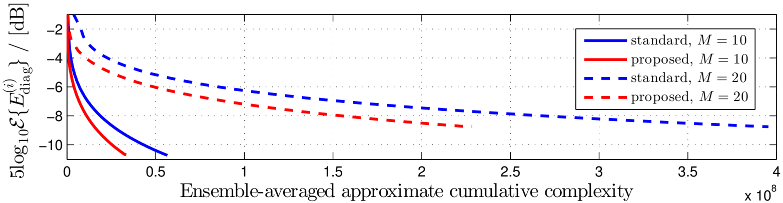
$$\boldsymbol{G}_{(\alpha+1)}(z) = \boldsymbol{F}_{(\alpha)}(z)\cdots\boldsymbol{F}_{(0)}(z) = \left(\prod_{x=0}^{\alpha}\boldsymbol{F}_{(\alpha-x)}(z)\right) .$$

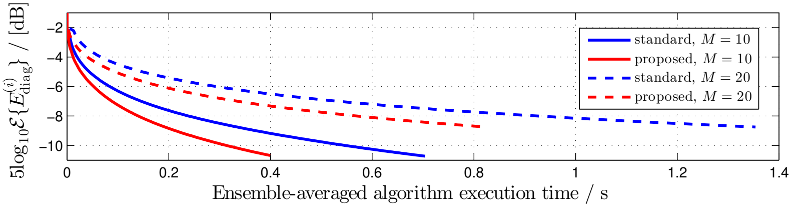
- $\boldsymbol{R}_{(\alpha+1)}(z)$ is then input to the $(\alpha+1)$th instantiation of the restricted update step and iterations of RU-SMD continue.

- If the total number of RU-SMD algorithm iterations exceeds some user-defined value I, or if the energy in the shifted column falls below a user-defined threshold, the algorithm ends with $\boldsymbol{\Lambda}(z) = \boldsymbol{R}_{(\alpha+1)}(z)$ and $\boldsymbol{Q}(z) = \boldsymbol{G}_{(\alpha+1)}(z)$.

RU-SMD Performance

- Algorithm execution time and complexity requirements reduced.



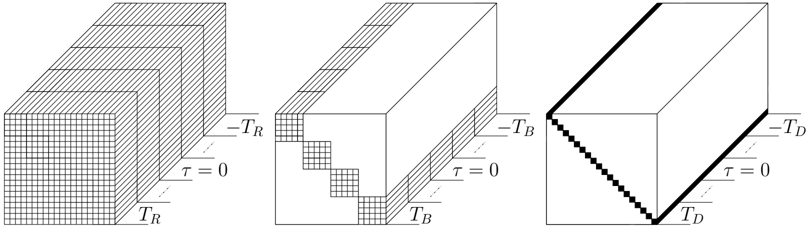


Summary of Algorithmic Improvements

- Improvements to algorithm implementation efficiency.
- Half-matrix form for parahermitian matrix.
- Restricted update approach.
- For the interested reader, see my thesis!
- Problem: none of these address increased algorithmic complexity (proportional to M^3) as spatial dimension M increases.

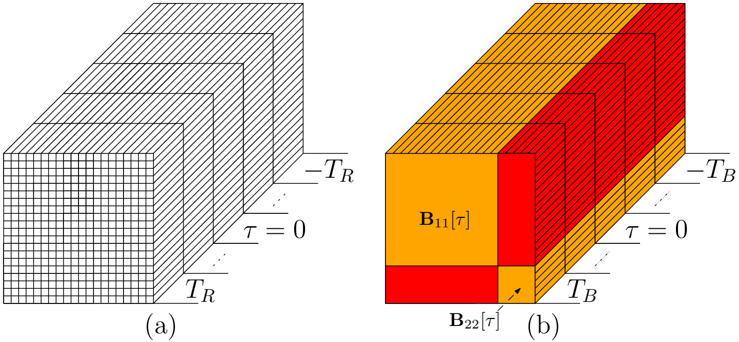
‘Divide-and-Conquer’ (DaC) Approach to the PEVD

- Traditional PEVD algorithms are tasked with diagonalising an entire $M \times M$ parahermitian matrix via sequential operations.
- DaC method first ‘divides’ the matrix into a number of smaller, independent parahermitian matrices, before diagonalising — or ‘conquering’ — each matrix separately.
- DaC scheme can substantially reduce PEVD complexity, which is typically proportional to M^3.



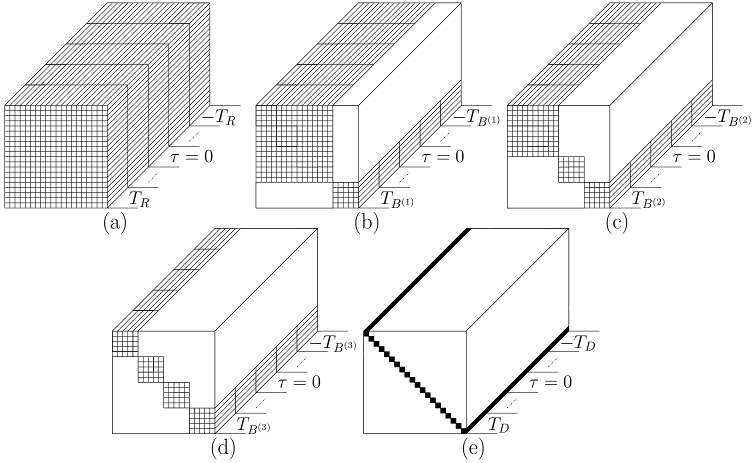
‘Divide-and-Conquer’ (DaC) Approach to the PEVD

- Iteratively minimising energy in red regions yields a block diagonal parahermitian matrix.
- Remaining $\boldsymbol{B}_{11}(z)$ and $\boldsymbol{B}_{22}(z)$ are independent parahermitian matrices and can be diagonalised separately.



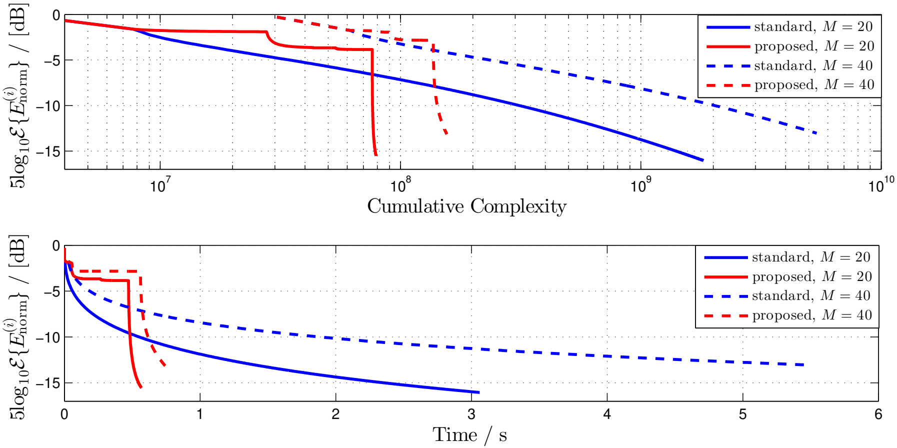
‘Divide-and-Conquer’ (DaC) Approach to the PEVD

- Several block diagonalisation steps yield a block diagonal matrix.



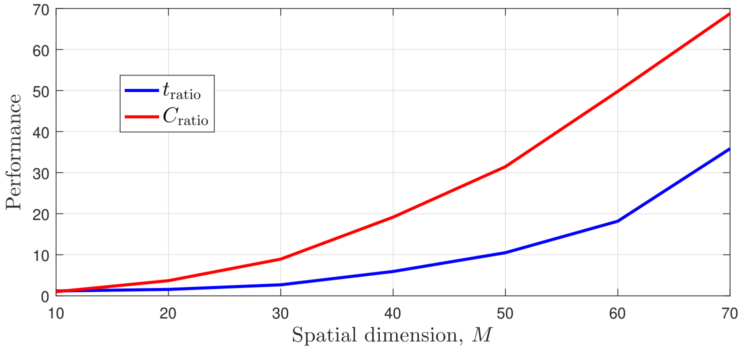
‘Divide-and-Conquer’ (DaC) Approach to the PEVD

- SMD (standard) versus DaC SMD (DC-SMD, proposed).



‘Divide-and-Conquer’ (DaC) Approach to the PEVD

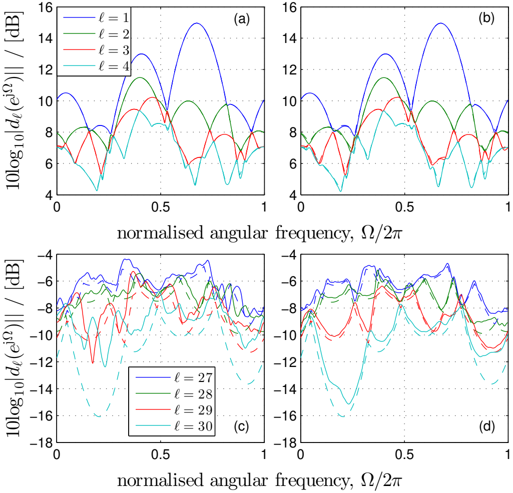
- Divide-and-conquer strategy becomes increasingly useful as spatial dimension M increases.



$$t_{\text{ratio}} = \frac{\mathcal{E}\{\text{Execution time}_{\text{SMD},-10\,\text{dB},M}\}}{\mathcal{E}\{\text{Execution time}_{\text{DC}-\text{SMD},-10\,\text{dB},M}\}} \qquad C_{\text{ratio}} = \frac{\mathcal{E}\Big\{\text{Complexity}_{\text{SMD},-10\,\text{dB},M}\Big\}}{\mathcal{E}\Big\{\text{Complexity}_{\text{DC}-\text{SMD},-10\,\text{dB},M}\Big\}}$$

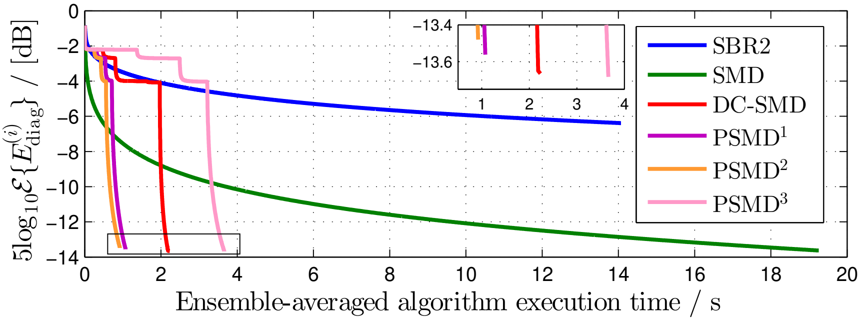
'Divide-and-Conquer' (DaC) Approach to the PEVD

- Power spectral densities of the (a,b) first and (c,d) last four eigenvalues obtained from (a,c) SMD and (b,d) DC-SMD.



‘Divide-and-Conquer’ (DaC) Approach to the PEVD

- Independent parahermitian matrices $\Rightarrow$ parallel processing.
- Task: combine parallelised DaC strategies for the PEVD with developed complexity and memory reduction techniques.
- **Parallel**-Sequential Matrix Diagonalisation (PSMD).



‘Divide-and-Conquer’ (DaC) Approach to the PEVD

- Independent parahermitian matrices $\Rightarrow$ parallel processing.
- Task: combine parallelised DaC strategies for the PEVD with developed complexity and memory reduction techniques.
- **Parallel**-Sequential Matrix Diagonalisation (PSMD).

Method	E.val. Res.	MSE	Paraun. Err.	L_Q
SBR2 [1]	1.1305	1.293×10^{-6}	2.448×10^{-8}	133.8
SMD [2]	0.0773	3.514×10^{-6}	6.579×10^{-8}	165.5
DC-SMD	0.0644	6.785×10^{-6}	1.226×10^{-14}	360.4
PSMD^1	0.0658	6.918×10^{-6}	4.401×10^{-15}	279.3
PSMD^2	0.0661	8.346×10^{-6}	1.303×10^{-8}	156.0
PSMD^3	0.0245	7.618×10^{-7}	1.307×10^{-14}	307.6

[1] J. G. McWhirter et al. An EVD Algorithm for Para-Hermitian Polynomial Matrices. *IEEE Trans. on Signal Process.*, 55(5):2158–2169, May 2007.

[2] S. Redif et al. Sequential Matrix Diagonalisation Algorithms for Polynomial EVD of Parahermitian Matrices. *IEEE Trans. on Signal Process.*, 63(1):81–89, Jan. 2015.

DFT-Based PEVD Algorithms

- We have seen one of the two main categories of PEVD algorithm:

1. Iterative time (lag) based methods.
    - Directly manipulate polynomial coefficients and seek to iteratively diagonalise parahermitian matrix $\boldsymbol{R}(z)$.
    - Encourage spectral majorisation of eigenvalues.
    - Established algorithms: SBR2 [1] and SMD [2].
    - Recent low-complexity, divide-and-conquer algorithm in [3] for large arrays.

[1] J. G. McWhirter et al. An EVD Algorithm for Para-Hermitian Polynomial Matrices. *IEEE Trans. on Signal Process.*, 55(5):2158–2169, May 2007.

[2] S. Redif et al. Sequential Matrix Diagonalisation Algorithms for Polynomial EVD of Parahermitian Matrices. *IEEE Trans. on Signal Process.*, 63(1):81–89, Jan. 2015.

[3] F. K. Coutts et al. Divide-and-conquer sequential matrix diagonalisation for parahermitian matrices. *In Proc. Sensor Signal Processing for Defence*, Dec. 2017.

DFT-Based PEVD Algorithms

- An alternative strategy:

2. Fixed order frequency based methods.
   - Transform the problem into a pointwise-in-frequency standard matrix decomposition.
   - Able to provide a spectrally majorised decomposition, or attempt to approximate maximally smooth, analytic eigenvalues.
   - Formulation in [4] performs well for finite order problems [5], but requires an a priori guess of the polynomial order of $\boldsymbol{Q}(z)$.

[4] M. Tohidian et al. A DFT-based approximate eigenvalue and singular value decomposition of polynomial matrices. *EURASIP J. Adv. Signal Process.*, 2013:93, December 2013.
[5] F. K. Coutts et al. A Comparison of Iterative and DFT-Based Polynomial Matrix Eigenvalue Decompositions. *In Proc. IEEE 7th Int. Workshop Comp. Advances in Multi-Sensor Adaptive Process.*, Dec. 2017.

Fixed Order Frequency Based Methods

- An existing approach obtains an approximate PEVD via K independent EVDs in the discrete frequency domain:

$$\mathbf{R}[k] = \mathbf{Q}[k]\boldsymbol{\Lambda}[k]\mathbf{Q}^{\mathrm{H}}[k]\,,\ k = 0\ldots K-1\,,$$

$$\mathbf{R}[k] = \boldsymbol{R}(z)|_{z=\mathrm{e}^{\mathrm{j}\Omega_k}} = \sum\nolimits_{\tau}\mathbf{R}[\tau]\mathrm{e}^{-\mathrm{j}\Omega_k\tau}\,, \qquad \Omega_k = 2\pi k/K$$

- Can rearrange the eigenvalues and -vectors at each frequency bin.
- Ambiguity in eigenvector phase leads to discontinuities in phase between eigenvectors in adjacent frequency bins.
- Phase alignment step alters phases to minimise discontinuities.
- Following the permutation (if desired) and phase alignment of $\mathbf{Q}[k]$, $\mathbf{Q}[\tau]$ and $\boldsymbol{\Lambda}[\tau]$ are computed via the inverse DFT.

Fixed Order Frequency Based Methods

- While analytic polynomial eigenvalues and eigenvectors have been shown to exist as absolutely convergent Laurent series in [6] there is currently no way of knowing the length of the series a priori.
- When converting $\mathbf{\Lambda}[k]$ and $\mathbf{Q}[k]$ to the lag domain, the order of the IDFT restricts the series' length to K.
- For unsufficiently large K, this can result in energy from ignored high order polynomial coefficients corrupting the fixed set of K coefficients (i.e., time domain aliasing).

[6] S. Weiss et al. On the Existence and Uniqueness of the Eigenvalue Decomposition of a Parahermitian Matrix. *IEEE Trans. on Signal Process.*, 66(10):2659–2672, May 2018.

Proposed Approach Overview

- We propose an iterative frequency-based scheme:

$$\mathbf{R}[k] = \mathbf{Q}[k]\mathbf{\Lambda}[k]\mathbf{Q}^{\mathrm{H}}[k]\,,\ k = 0 \ldots K_i - 1\,.$$

- K_i is increased within the set $K_i \in \{2^{l+i} \,|\, l, i \in \mathbb{Z}^+,\, 2^l > L\}$.
- L is the polynomial order of $\boldsymbol{R}(z)$.
- $i = 0, 1, \ldots, I-1$ records current iteration.
- Use method for reordering the eigenvalues and -vectors from [4].
- Use a phase alignment function from [7], which uses Powell's 'dogleg' algorithm [8] to maximise eigenvector smoothness.
- Iterations continue while decomposition MSE above threshold ϵ.

[4] M. Tohidian et al. A DFT-based approximate eigenvalue and singular value decomposition of polynomial matrices. *EURASIP J. Adv. Signal Process.*, 2013:93, December 2013.

[7] F. K. Coutts et al. Enforcing Eigenvector Smoothness for a Compact DFT-based Polynomial Eigenvalue Decomposition. *In Proc. IEEE Workshop on Sensor Array and Multichannel Sig. Process.*, July. 2018.

[8] M. J. D. Powell. A new algorithm for unconstrained optimization. *Nonlinear programming*, 31–65, 1970.

Proposed Approach Overview

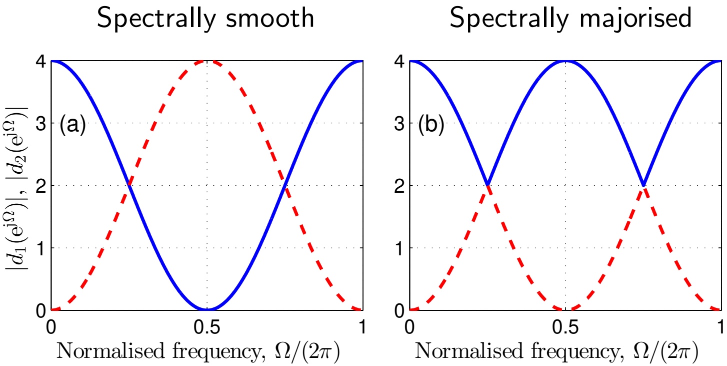
- We propose an iterative frequency-based scheme:

$$\mathbf{R}[k] = \mathbf{Q}[k]\mathbf{\Lambda}[k]\mathbf{Q}^{\mathrm{H}}[k]\,,\ k = 0 \ldots K_i - 1\,.$$

- K_i is increased within the set $K_i \in \{2^{l+i} \,|\, l, i \in \mathbb{Z}^+,\, 2^l > L\}$.
- In iteration i, $\mathbf{Q}[k]$ and $\mathbf{\Lambda}[k]$ for $k = 0, 1, \ldots, K_i - 1$ are identical for $k = 0, 2, 4, \ldots, K_{i+1} - 2$ in the $(i+1)$th iteration.
- Phase alignment step exploits this to aid optimisation.
- $\mathbf{Q}[\tau]$ and $\mathbf{\Lambda}[\tau]$ are computed via the inverse DFT following the permutation (if desired) and phase alignment of $\mathbf{Q}[k]$.
- MSE computed between $\hat{\boldsymbol{R}}(z) = \boldsymbol{Q}(z)\boldsymbol{\Lambda}(z)\boldsymbol{Q}^{\mathrm{P}}(z)$ and $\boldsymbol{R}(z)$.

Smooth Decomposition

- In a smooth (analytic) decomposition, the eigenvalues and -vectors are arranged such that discontinuities between adjacent frequency bins are minimised.



Smooth Decomposition

- In a smooth (analytic) decomposition, the eigenvalues and -vectors are arranged such that discontinuities between adjacent frequency bins are minimised.
- For a smooth decomposition, the eigenvectors in adjacent frequency bins are rearranged using the inner product

  $$\mathsf{c}_{mn}[k] = \mathbf{q}_m^{\mathrm{H}}[k-1]\mathbf{q}_n[k] \ ,$$

  where, $\mathbf{q}_m[k]$ is the mth column of $\mathbf{Q}[k]$.
- $\mathsf{c}_{mn}[k] \approx 1$ if $\mathbf{q}_m[k-1]$ & $\mathbf{q}_n[k]$ aligned; $\mathsf{c}_{mn}[k] \approx 0$ otherwise.
- Goal: reorder columns of $\mathbf{Q}[k]$ such that $\mathsf{c}_{mm}[k] \approx 1 \ \forall \ m$.
- $\mathbf{\Lambda}[k]$ rearranged according to the reordering of the eigenvectors.

Phase Alignment

- Phase alignment of eigenvectors in adjacent frequency bins is vital for a compact-order decomposition.
- A matrix $\mathbf{C}^{(P)}$ [7] can be used to calculate the total power in the derivatives of a function up to and including the Pth derivative.
  - Function 'smoothness' is characterised by a low total power.
- Phase of the mth eigenvector at frequency bin k can be adjusted by an angle θ_k according to $\mathbf{q}_m[k] \leftarrow e^{\mathrm{j}\theta_k}\mathbf{q}_m[k]$.
- We compute a vector of phases $\boldsymbol{\theta} = [\theta_0, \cdots, \theta_{K_i-1}]^{\mathrm{T}}$ s.t. the mth eigenvector $\mathbf{q}_m[k] \; \forall \; k$ is maximally smooth.

[7] F. K. Coutts et al. Enforcing Eigenvector Smoothness for a Compact DFT-based Polynomial Eigenvalue Decomposition. *In Proc. IEEE Workshop on Sensor Array and Multichannel Sig. Process.*, July. 2018.

Phase Alignment

- An objective function has been derived in [7] that measures the smoothness of all elements of $\mathbf{q}_m[k]$ and takes the form

$$f(\boldsymbol{\theta}) = \mathbb{R}\{\boldsymbol{u}^{\mathrm{H}}\boldsymbol{\Gamma}\boldsymbol{u}\}\ .$$

- $\boldsymbol{u}^{\mathrm{H}} = [e^{\mathrm{j}\theta_0}, \cdots, e^{\mathrm{j}\theta_{K_i-1}}]$, $\boldsymbol{\Gamma} = \sum_{n=0}^{M-1} \mathrm{diag}\{\mathbf{v}_n\}\mathbf{C}_{(P)}\mathrm{diag}\{\mathbf{v}_n^{\mathrm{H}}\}$, and $\mathbf{v}_n = [\mathbf{q}_{m,n}[0], \cdots, \mathbf{q}_{m,n}[K_i - 1]]$.
- $\mathbf{q}_{m,n}[k]$ denotes the nth element (row) of eigenvector $\mathbf{q}_m[k]$.
- We employ relatively low cost Powell's iterative 'dogleg' trust region strategy [8] for the unconstrained minimisation of $f(\boldsymbol{\theta})$.
- For $i > 0$, can use previous $\boldsymbol{\theta}$ to give a more informed initial guess.

[7] F. K. Coutts et al. Enforcing Eigenvector Smoothness for a Compact DFT-based Polynomial Eigenvalue Decomposition. *In Proc. IEEE Workshop on Sensor Array and Multichannel Sig. Process.*, July. 2018.
[8] M. J. D. Powell. A new algorithm for unconstrained optimization. *Nonlinear programming*, 31–65, 1970.

DFT-Based PEVD Performance

- DFT-based PEVD algorithm capable of outperforming existing methods.
- Suitable for scenarios with a high number of sensors.
- Example below has $\boldsymbol{R}(z) : \mathbb{C} \to \mathbb{C}^{5\times 5}$ of order 38 with ground truth polynomial eigenvalues that are **not** spectrally majorised.

Method	MSE	Paraun. Err.	E_{diag}	Time / s	L_Q	Complexity
proposed	5.750×10^{-29}	2.887×10^{-22}	0	0.08854	64	$\mathcal{O}\left(ML^3\right)$
SBR2 [1]	1.815×10^{-6}	3.303×10^{-8}	10^{-6}	37.64	600.0	$\mathcal{O}\left(M^2L\right)$
SMD [2]	9.321×10^{-7}	3.847×10^{-8}	10^{-6}	11.34	357.9	$\mathcal{O}\left(M^3L\right)$

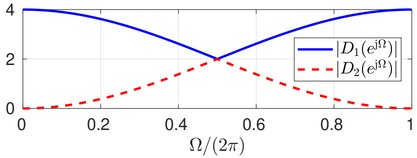
[1] J. G. McWhirter et al. An EVD Algorithm for Para-Hermitian Polynomial Matrices. *IEEE Trans. on Signal Process.*, 55(5):2158–2169, May 2007.

[2] S. Redif et al. Sequential Matrix Diagonalization Algorithms for Polynomial EVD of Parahermitian Matrices. *IEEE Transactions on Signal Processing*, Jan. 2015.

DFT-Based PEVD Performance

- Decompose the theoretical parahermitian matrix

$$\boldsymbol{R}(z) = \begin{bmatrix} 2 & z^{-1}+1 \\ z+1 & 2 \end{bmatrix} .$$



- Eigenvectors & eigenvalues are neither of finite order nor rational.
- To decompose $\boldsymbol{R}(z)$ via an exact PEVD would require polynomial matrices of infinite length.

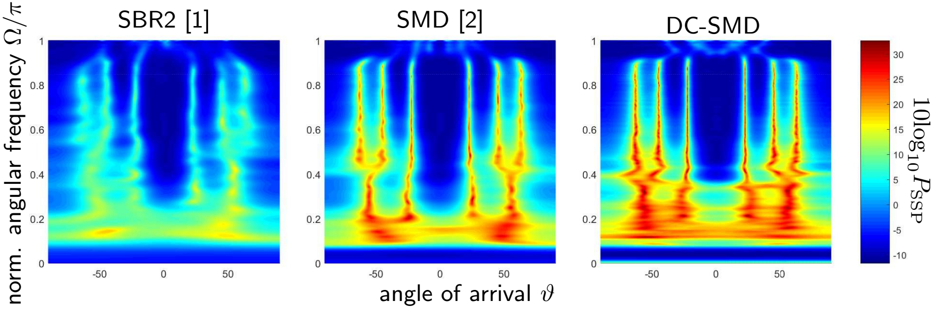
Method	MSE	Paraun. Err.	E_{diag}	Time / s	L_Q
proposed	7.077×10^{-9}	1.381×10^{-4}	0	0.1196	64
SMD, μ_1	4.362×10^{-25}	2.466×10^{-16}	10^{-6}	0.6256	345
SMD, μ_2	2.909×10^{-10}	9.546×10^{-8}	10^{-6}	0.1995	83
SBR2	2.909×10^{-10}	9.546×10^{-8}	10^{-6}	0.1724	83

Angle of Arrival Estimation (AoA)

- Accuracy of noise-only subspace strongly dependent on quality of PEVD.
- Each PEVD algorithm will produce a different AoA estimation performance.
  - The results you have seen used the SBR2 algorithm.
- Divide-and-conquer strategies offer very fast diagonalisation performance, and are able to resolve weaker polynomial eigenvalues.
  - Does this translate to better AoA estimation performance if simulation time is fixed?

Angle of Arrival Estimation Results

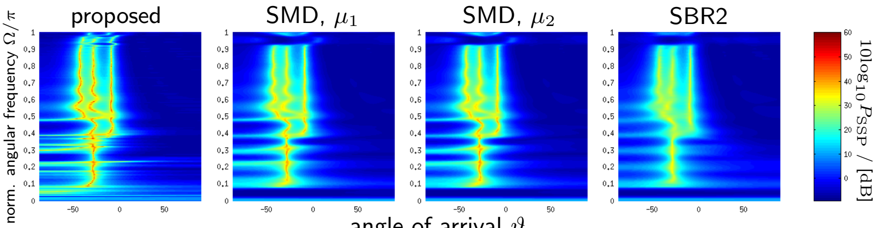
- 'Divide-and-conquer' (DaC) approach to the PEVD:
- 6 sources sharing frequency range $\Omega \in [0.1\pi, 0.9\pi]$, $20\,\mathrm{dB}$ SNR.

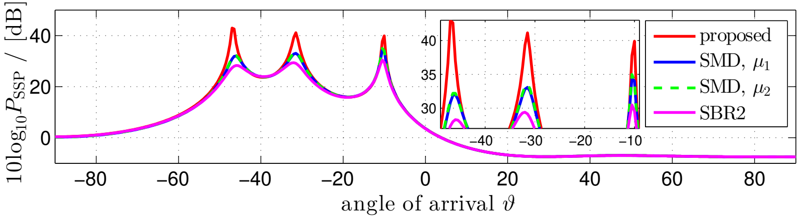


[1] J. G. McWhirter et al. An EVD Algorithm for Para-Hermitian Polynomial Matrices. *IEEE Trans. on Signal Process.*, 55(5):2158–2169, May 2007.
[2] S. Redif et al. Sequential Matrix Diagonalization Algorithms for Polynomial EVD of Parahermitian Matrices. *IEEE Transactions on Signal Processing*, Jan. 2015.

Angle of Arrival Estimation Results

- Frequency-Based PEVD Algorithms:
- 3 sources with different frequency ranges, $20\,\mathrm{dB}$ SNR.

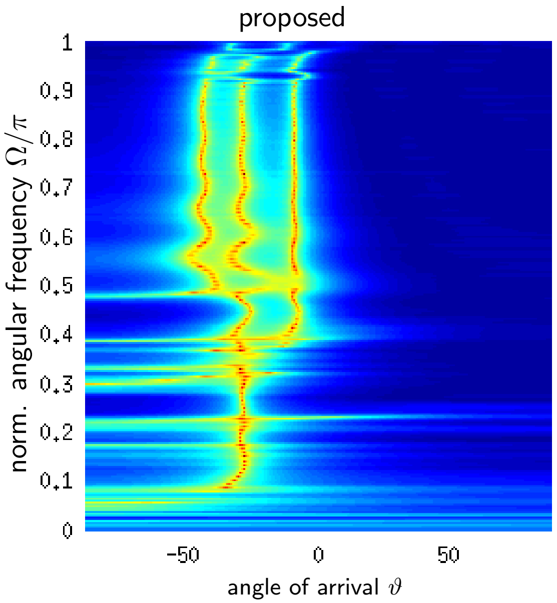




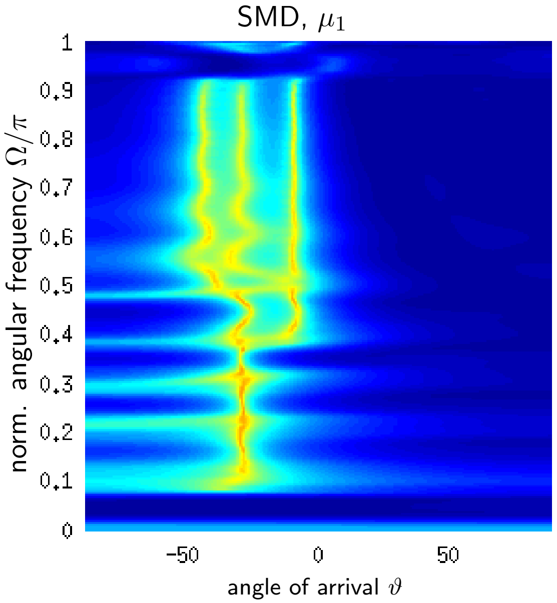
Conclusion

- Summarised existing iterative PEVD algorithms.
- Introduced some methods used to lower the computational cost of these algorithms.
- Gave overview of ‘divide-and-conquer’ approach to the PEVD.
- Explained DFT-based PEVD approaches and their advantages.
- Demonstrated the effects of PEVD algorithm choice on AoA estimation results.

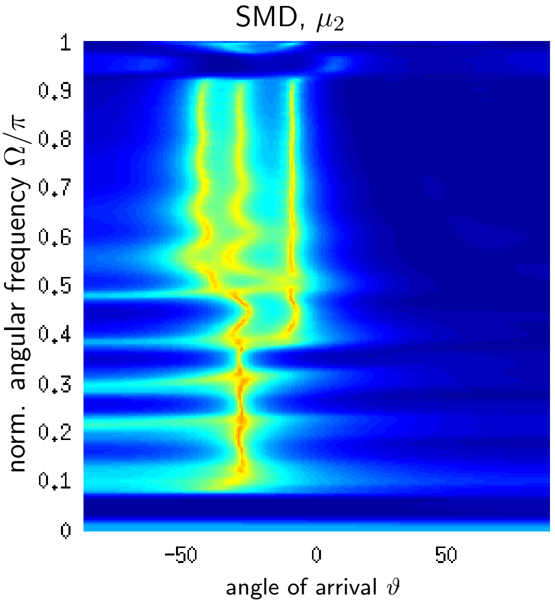
Simulation Scenario 3 Zoomed



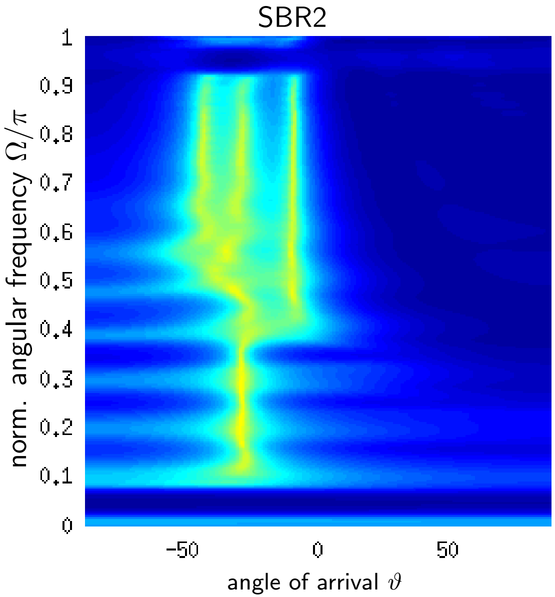
Simulation Scenario 3 Zoomed



Simulation Scenario 3 Zoomed

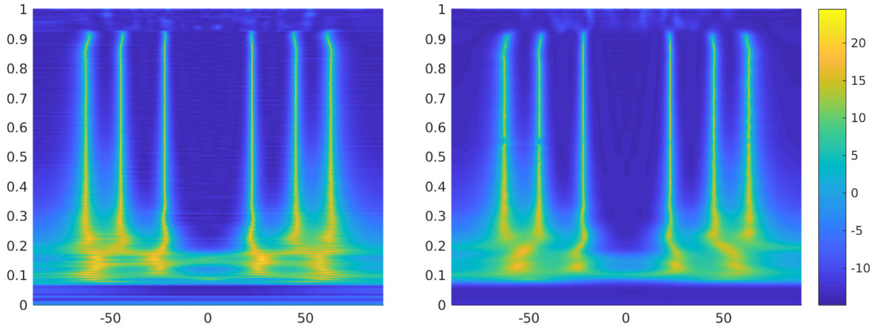


Simulation Scenario 3 Zoomed



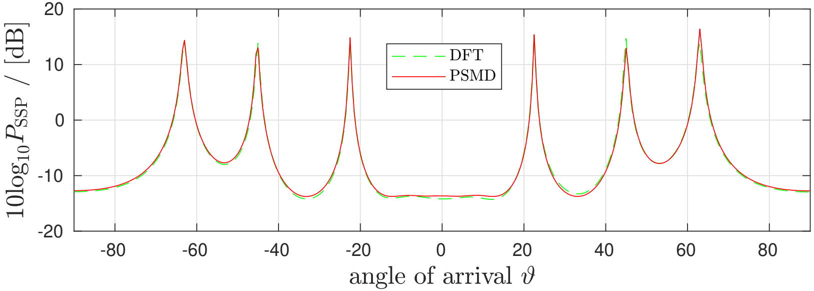
AoA Comparison for DFT-Based and PSMD Algorithms

- Both algorithms executed for 1.5 seconds
- $M = 24$
- 6 sources
- DFT-based (left) vs PSMD (right)



AoA Comparison for DFT-Based and PSMD Algorithms

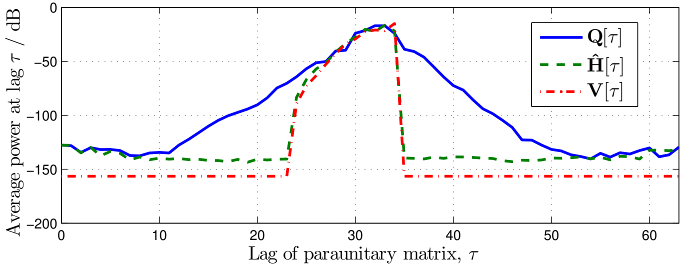
- Both algorithms executed for 1.5 seconds
- $M = 24$
- 6 sources
- Evaluated at $\Omega = \pi/2$



Approximating a minimum-order solution to the PEVD

$$\boldsymbol{R}(z) = \boldsymbol{V}(z)\boldsymbol{\Lambda}(z)\boldsymbol{V}^{\mathrm{P}}(z) \approx \boldsymbol{Q}(z)\boldsymbol{D}(z)\boldsymbol{Q}^{\mathrm{P}}(z)$$
$$\boldsymbol{Q}(z) = \boldsymbol{H}(z)\boldsymbol{U}(z)$$

- Task: find all-pass filter bank $\boldsymbol{U}(z) = \mathrm{diag}\{u_1(z), \ldots, u_M(z)\}$.
- $u_m(z)$ defined by the greatest common divisor [9] of $\boldsymbol{q}_m(z)$.



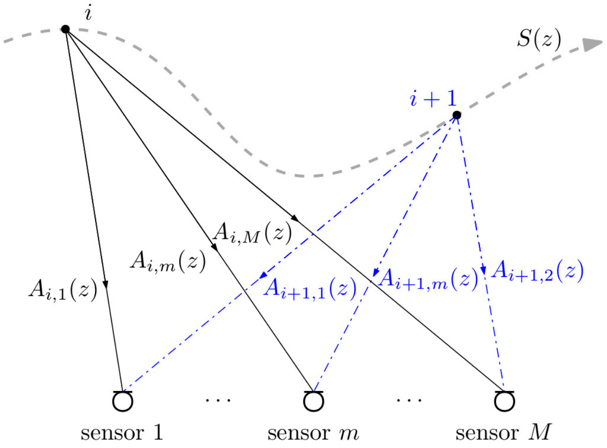
[9] F. C. Chang. Polynomial GCD derived through monic polynomial subtractions. *ISRN Applied Mathematics*, 2011.

Future Work

- Impact of divide-and-conquer algorithms on spectral majorisation.
- Investigate scenarios with 'naturally' (up to permutations) block diagonal matrices.
- Alternative permutation/optimisation strategies for the DFT-based PEVD.
- Minimum-order solutions and impulse response estimation.
- Deploying developed algorithms in the real world.

Future Work

- Minimum-order solutions and impulse response estimation:



[9] F. C. Chang. Polynomial GCD derived through monic polynomial subtractions. *ISRN Applied Mathematics*, 2011.

[10] S. Weiss et al. Identification of broadband source-array responses from sensor second order statistics. *IEEE Sensor Signal Processing for Defence Conference*, Dec. 2017

Future Work

- Minimum-order solutions and impulse response estimation:
  - Cross-spectral density matrix $\boldsymbol{R}_i(z)$ defined by transfer functions, $\boldsymbol{a}_i(z)$, and source power spectral density, $\boldsymbol{S}(z)$:

$$\boldsymbol{R}_i(z) = \boldsymbol{a}_i(z)S(z)\boldsymbol{a}_i^{\mathrm{P}}(z) + \sigma_n^2\mathbf{I}_M \approx \boldsymbol{q}_i(z)d_i(z)\boldsymbol{q}_i^{\mathrm{P}}(z) + \sigma_n^2\mathbf{I}_M$$

$$\boldsymbol{a}_i^{\mathrm{P}}(z)\boldsymbol{a}_i(z) = A_i(z) = A_i^{(+)}(z)A_i^{(+),\mathrm{P}}(z)\,, \qquad \boldsymbol{a}_{i,\mathrm{norm}}(z) = \frac{\boldsymbol{a}_i(z)}{A_i^{(+)}(z)}$$

$$\boldsymbol{R}_i(z) \approx \boldsymbol{a}_{i,\mathrm{norm}}(z)A_i^{(+)}(z)S(z)A_i^{(+),\mathrm{P}}(z)\boldsymbol{a}_{i,\mathrm{norm}}^{\mathrm{P}}(z) + \sigma_n^2\mathbf{I}_M$$

$$\boldsymbol{q}_i(z) = \frac{\boldsymbol{a}_i(z)}{A_i^{(+)}}\,, \qquad d_i(z) = A_i^{(+)}(z)S(z)A_i^{(+),\mathrm{P}}(z)$$

[9] F. C. Chang. Polynomial GCD derived through monic polynomial subtractions. *ISRN Applied Mathematics*, 2011.

[10] S. Weiss et al. Identification of broadband source-array responses from sensor second order statistics. *IEEE Sensor Signal Processing for Defence Conference*, Dec. 2017

Future Work

- Minimum-order solutions and impulse response estimation:

$$\boldsymbol{q}_i(z) = \frac{\boldsymbol{a}_i(z)}{A_i^{(+)}}\,, \qquad d_i(z) = A_i^{(+)}(z) S(z) A_i^{(+)\mathrm{P}}(z)$$

$$\hat{S}(z) = \mathrm{GCD}\left\{d_1(z), \ldots, d_I(z)\right\}$$

  - Phase ambiguity in $\boldsymbol{q}_i(z) = \hat{\boldsymbol{q}}_i(z) u_i(z)$ can be eliminated through determination of greatest common divisor (GCD) $u_i(z)$ [9].

$$\hat{\boldsymbol{a}}_i(z) = \hat{A}_i^{(+)}(z) \hat{\boldsymbol{q}}_i(z)$$

  - Magnitude **and** phase of transfer functions recovered.
  - Next problem: identifying GCD of noisy eigenvalues.

[9] F. C. Chang. Polynomial GCD derived through monic polynomial subtractions. *ISRN Applied Mathematics*, 2011.

[10] S. Weiss et al. Identification of broadband source-array responses from sensor second order statistics. *IEEE Sensor Signal Processing for Defence Conference*, Dec. 2017