Direction of Arrival Estimation based on AI/ML Methods: SIAM Mathematical Problems in Industry Workshop 2025

Hesham Abdelfattah
University of Cincinnati

Toluwanimi Akinwande
Claremont Graduate University

Ghan Bhatt
Tennessee State University

Harshit Bhatt
North Carolina State University

Kendra Calman
Cal Poly Pomona

Sayoni Chakraborty
The University of Texas at Dallas

Jessie Chen
North Carolina State University

Kausik Das
University of Michigan

Ansh Desai
University of Delaware

Pak-Wing Fok
University of Delaware

Marissa Gee
Kenyon College

Geoffrey Hewitt
Claremont Graduate University

Taras Lakoba
University of Vermont

Zhihua Li
University of Iowa

Emeka Mazi
Georgia State University

Kristiana Nakaj
Virginia Tech

Rebecca Rodrigues
Rochester Institute of Technology

Isabelle Sanz
Georgia Institute of Technology

Teodor Vakarelsky
Bulgarian Academy of Sciences

Yifan Wu
Harvey Mudd College

September 7, 2025

Raytheon Project Supervisors: Al Samuel, Rich Scholes, and Pranav Prabhakaran Sudha

1 Introduction

Subspace based techniques [Sch82] in Direction of Arrival (DoA) estimation problems constitute a class of common and well-understood methods for parameter estimation. Specific subspace methods for DoA estimation include MUSIC (MUltiple SIgnal Classification) [SS02], Matrix Pencil [SP95, HS02], and the ESPRIT [RK89] algorithms.

In Direction of Arrival (DoA) problems, an array of antennas is used to detect the incidence angles of a plane-wave signal. Such arrays are common in applications such as radar systems [GRY+07] and astronomy [DTVB09]. A key output of the MUSIC algorithm is a noise subspace matrix that can be analyzed to compute angles or incoming directions of the incident waves. Recent results from Raytheon using COSnet, a neural network, have yielded "surrogate" matrices that outperform the MUSIC matrices in many situations. The task we have been given at this Mathematical Problems in Industry (MPI) workshop is to characterize these COSnet derived matrices, and understand their improved performance.

In section 2, we set up the DoA problem mathematically and describe MUSIC and COSnet's approaches to inferring angles of arrival. Section 3 contains our results and is the largest section of this report. Collectively, we generated about about 10 different ideas for solving DoA problems using MUSIC, and COSnet. These are described in 11 subsections:

- Section 3.1: Numerical Comparisons between MUSIC vs COSnet
- Section 3.2: Metric for, and dependence on the antenna separation of, angular resolution
- Section 3.3: A VarPro-Style Minimization Method
- Section 3.4: Integral Equation Approaches
- Section 3.5: Sparse Recovery of Sources
- Section 3.6: Upper Limits of Source Detection
- Section 3.7: Spectral Analysis of MUSIC and COSnet Matrices
- Section 3.8: Fast Computation of the COSnet Matrix
- Section 3.9: Fourth Order Statistics
- Section 3.10: Alternative Methods for Subspace Identification
- Section 3.11: Formal Description of Solving DoA via Machine Learning

In section 4, we conclude our report and summarize our findings.

2 Description of Methods

2.1 DoA Estimation via MUSIC

The MUSIC method [SS02] provides a way to estimate angles of incidence given the array signals measured at K time instances. Mathematically,

$$x_k = As_k + n_k, \qquad k = 1, \ldots, K, \tag{1}$$

where $x_k \in \mathbb{R}^N$ is the kth measured signal in each of the N antennas,

$$A = [a(\theta_1)\ a(\theta_2)\ \ldots\ a(\theta_M)], \tag{2}$$

$$a(\theta) = (1, e^{j2\pi d \sin\theta}, e^{j2\pi 2d \sin\theta}, e^{j2\pi 3d \sin\theta}, \ldots, e^{j2\pi(N-1)d \sin\theta})^T, \tag{3}$$

where $j = \sqrt{-1}$; d is the physical separation between the antennas; θ_m, $m = 1, \ldots, M$ are the angles of each of the M signals (a.k.a. sources); s_k is a complex signal amplitude; and n_k is the noise in the measurement of x_k, assumed to be Gaussian in this report. Additionally, N is the number of receivers or antennas, K is the number of time snapshots, and superscript 'T' denotes transposition. The matrix A in (2) consists of M column vectors, side-by-side.

MUSIC is based on analyzing the *sample correlation matrix*

$$\hat{R} = \frac{1}{K} \sum_{k=1}^{K} x_k x_k^H, \tag{4}$$

where superscript 'H' denotes Hermitian conjugation. Because $\hat{R}$ is positive definite, one can perform an eigenvalue decomposition $\hat{R} = V \Lambda V^H$ where V is the matrix of orthonormal eigenvectors and Λ is a diagonal matrix of positive eigenvalues. The matrix V can be partitioned into the signal subspace V_S and the noise subspace V_N so that

$$V = [V_S \quad V_N], \tag{5}$$

where the columns of V_S (V_N) span the signal (noise) subspace. The dimension of the column space of V_S is M, for M well-separated sources. Steering vectors oriented in the arrival direction θ^* are orthogonal to the noise subspace: $V_N^H a(\theta^*) = 0$. Therefore, the maxima of the function $1/|g_{MUSIC}(\theta)|$ where

$$g_{MUSIC}(\theta) = a^H(\theta) V_N V_N^H a(\theta), \tag{6}$$

yield the direction angles. The method is computationally efficient since only a 1D line search is required to determine $\vec{\theta} = (\theta_1, \ldots, \theta_M)$.

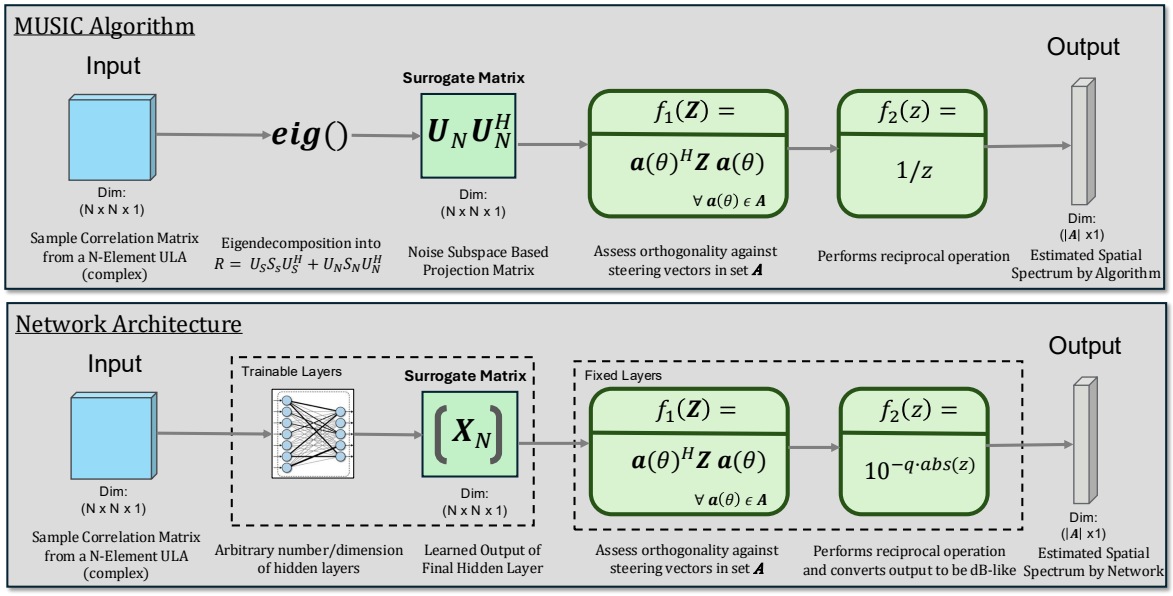

Figure 1: Comparison of MUSIC and COSnet for DoA estimation.

2.2 DoA estimation via COSnet

COSnet is a neural network developed by Raytheon that has been trained to predict angles of arrival based on signal data: see Figure 1. Training data is created by generating noisy x_k through eq. (1) for known $\vec{\theta}$. A learned output of the hidden trainable layers is an $N \times N$ complex-valued matrix X_{ML} which is used as a surrogate to $X_{MUSIC} = V_N V_N^H$ from eq. (6).

Specifically, for a given sample correlation matrix $\hat{R}$ from eq. (4), COSnet tries to construct a function

$$g_{ML}(\theta) = a^H(\theta) X_{ML} a(\theta). \tag{7}$$

that produces the right incidence angles. It achieves this by adjusting the entries of X_{ML} to minimize the cross-entropy of the two functions $f_2(\theta) = 10^{-q|g_{ML}(\theta)|}$ for some $q > 0$, and $f_0(\theta) = \sum_{j=1}^{M} \delta(\theta - \theta_j)$. In other words, COSnet is trained to find X_{ML} that makes f_0 as close to f_2 as possible. The process is summarized in Fig. 1. Note that because eigendecomposition is not involved, X_{ML} will not generally be positive definite. In summary, pairs of (Y_j, Θ_j) where the jth data set $Y_j = [x_{1,j}, \ldots, x_{K,j}]$ and $\Theta_j = (\theta_{1,j}, \ldots, \theta_{M,j})$ are provided as training data to COSnet until it can predict a $g_{ML}(\theta)$ for a provided $x_1, \ldots, x_K$: see Fig. 2 (left).

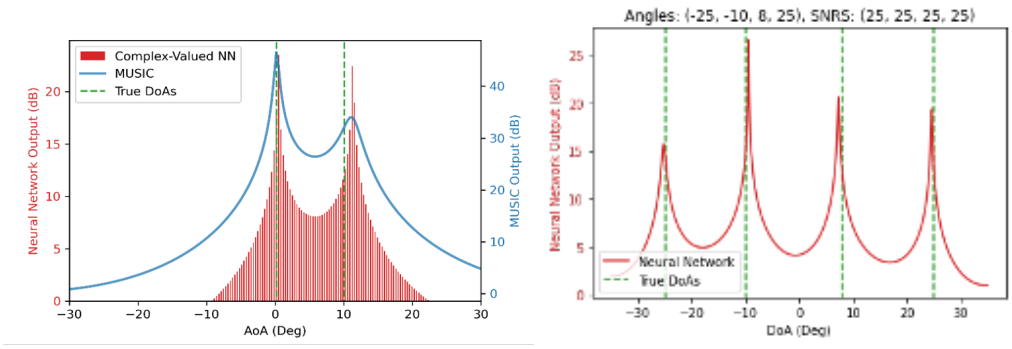

Figure 2: Left: Comparison of $1/|g_{MUSIC}(\theta)|$ (solid blue) and $1/|g_{ML}(\theta)|$ (red bars) to solve the two source DoA problem. Right: COSnet can solve the $M = N = 4$ case (4 sources, 4 receivers).

An interesting feature of COSnet is that it can accurately estimate angle directions when the

number of sources equals the number of receivers ($M = N$): see Figure 2 (right). For N antennas, the maximum number of sources that MUSIC can theoretically resolve is $M = N - 1$. The reason for this is that when $M = N$, $V = V_S$ in eq. (5). Since V_S is orthonormal with N linearly independent columns, $V_S V_S^H = I$ and because V_N and V_S are related through $V_N V_N^H = I - V_S V_S^H$,$V_N V_N^H$ must be the zero matrix and $g_{MUSIC}(\theta)$ in eq. (6) is the zero function. Based on some analysis from our group, we believe that COSnet should be able to resolve up to $M = 2N - 2$ sources. For more details, see section 3.6.2.

3 Results

3.1 Numerical Comparisons between MUSIC and COSnet

In this section, we compare the performance of MUSIC and COSnet under different conditions. First, we take $M = 2$ sources each with a different SNR (Signal to Noise). We change the difference in SNR and see which method is more accurate. Then, we study the $M = 3$ case. With three angles θ_1, θ_2 and θ_3 to infer, we see which method is more accurate when θ_2 and θ_3 become closer and closer together, i.e. the second two sources are more and more correlated.

3.1.1 Performance under Varying SNR and SNR Ratios

We first compared the two methods under varying Signal-to-Noise Ratio (SNR) conditions. The following parameters were held constant throughout the experiments:

- **N**: Number of elements in the Uniform Linear Array (ULA)
- **d**: Element spacing (expressed as a fraction of the wavelength)
- θ: True directions of arrival of two signal sources
- **Number of snapshots**: Number of samples per simulation run

Only the **SNR difference** between the two sources was varied, and the performance of the algorithms was compared using the **squared error of their DoA predictions**.

1. Single Trial

We conducted a *single* trial simulation in which the SNR of one source was held constant while the SNR of the other was incrementally increased. For each SNR combination, we calculated the squared errors of the predicted angles. Our key observations are:

- For **small SNR differences** (0–7 dB), both MUSIC and ML-based approaches performed well, with MUSIC showing **slightly better squared error performance** on average.
- As the SNR difference increases (around 7–10 dB), the ML algorithm begins to match MUSIC. The **crossover point** is observed around 8 dB, beyond which ML consistently outperforms MUSIC.
- At **high SNR differences** (12–20 dB), MUSIC becomes highly sensitive and unstable, exhibiting a sharp increase in squared error, while ML-based methods remain robust and reliable as shown in Figure 3.

2. Multiple Trials
To gain a finer understanding, we also *averaged* over 100 noise realizations to find the mean squared error. We conducted simulations with one source's SNR fixed while the other increased incrementally by 0.3 dB. Our key observations are:

- The **Mean Squared Error (MSE)** of MUSIC's DoA prediction increases monotonically with growing SNR difference.
- This confirms that MUSIC's accuracy degrades with uneven signal strengths, while ML models maintain consistent performance as displayed in Figure 4.

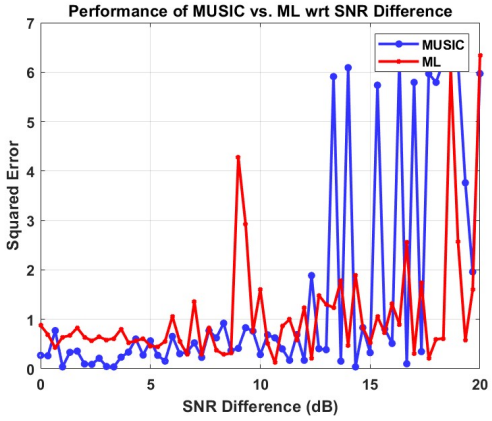

Figure 3: Squared Error of MUSIC and ML vs. SNR Differential

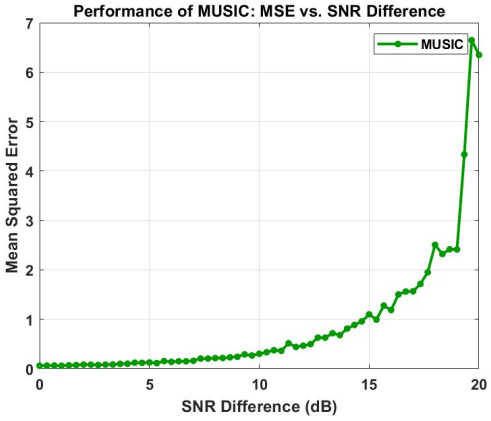

Figure 4: Mean Squared Error of MUSIC vs. SNR Differential

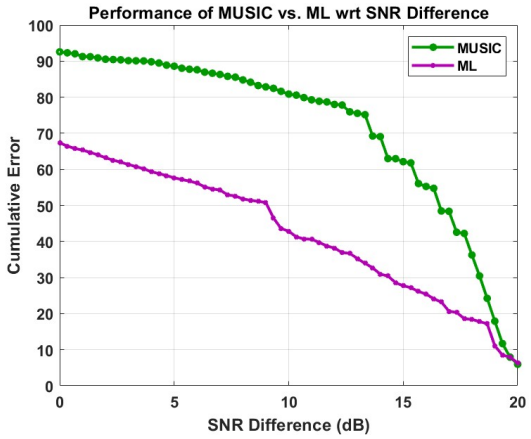

Figure 5: Cumulative Error of MUSIC and ML vs. SNR Differential

3. Cumulative Error Comparison

Finally, we used the **squared error of DoA predictions**, as defined in Section 1, to evaluate cumulative performance. The MUSIC-to-COSNET error ratio was found to be approximately **1.37**, indicating COSNET has lower prediction error across the SNR range. The **cumulative sum of squared errors** (which is a running sum of the squared errors for a single trial) shows COSNET consistently outperforms MUSIC, especially at higher SNR disparities. These results demonstrate that COSNET is a more stable and accurate alternative for DoA estimation under varying signal strength conditions as illustrated in Figure 5.

3.1.2 Changing Angle Separations

To compare the performance of the COSNet and MUSIC algorithms, we fix the source angles and perform 100 independent simulations for each method. Figures 6 and 7 present histograms of the angle estimates for MUSIC and COSNet, respectively, along with their corresponding mean squared errors (MSE). The results indicate that COSNet achieves significantly lower estimation error, demonstrating superior accuracy in recovering the true source angles compared to MUSIC in this case.

To further evaluate the sensitivity of both algorithms to the angular separation between sources, we fix two source angles and systematically vary the third, thereby altering the angular distance between the second and third sources. Figure 8 (left) presents the MSE of angle estimation as a function of

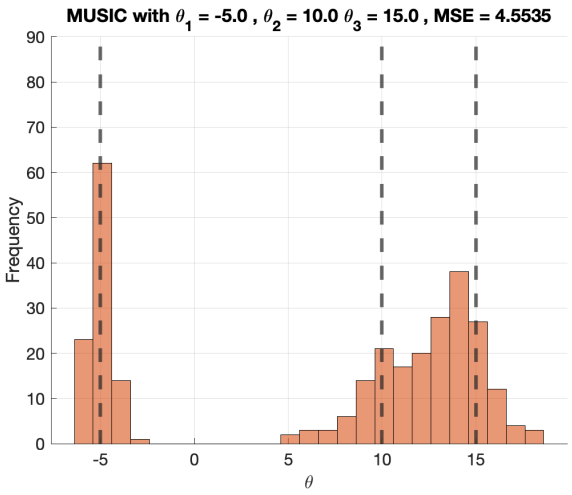

Figure 6: Histogram of angle estimates made by MUSIC using 100 independent simulations with $\theta_1 = -5°, \theta_2 = 10°$, and $\theta_3 = 15°$ (vertical dashed lines). The MSE was computed to be 4.5535.

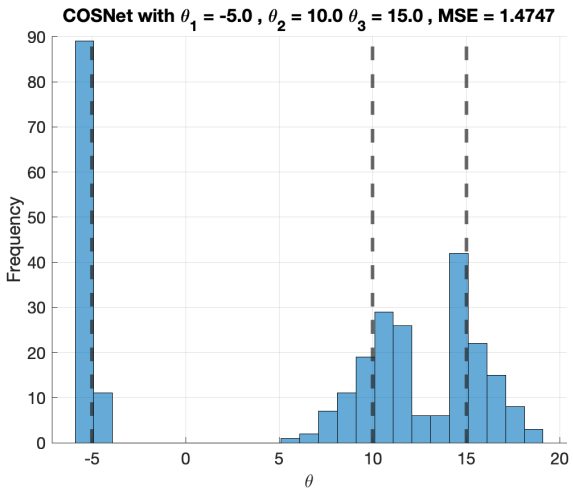

Figure 7: Histogram of angle estimates made by COSNet using 100 independent simulations with $\theta_1 = -5°, \theta_2 = 10°$, and $\theta_3 = 15°$ (vertical dashed lines). The MSE was computed to be 1.4747.

the third angle θ_3. The results indicate that MUSIC consistently yields higher MSE across the tested range, regardless of the angular separation. Figure 8 (right) illustrates the failure rate of each method in accurately detecting the correct number of peaks, given by the fraction of the 100 simulations that predicted fewer than 3 peaks. When the second and third sources are separated by 3° or less, MUSIC has a lower failure rate than COSNet. However, once the angle separation is 4° or greater, the failure rate for COSNet decreases significantly, demonstrating both improved estimation accuracy and robustness compared to MUSIC.

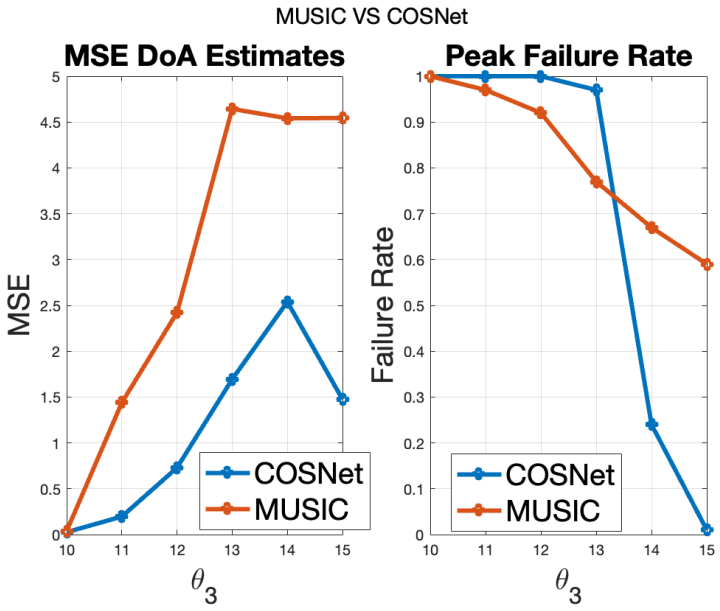

Figure 8: MSE (left) and peak identification failure rates (right) for DoA estimates made by MUSIC and COSNet when two sources are fixed at $\theta_1 = -5°$ and $\theta_2 = 10°$ and θ_3 approaches θ_2, beginning at 15°.

3.2 Metric for, and dependence on the antenna separation of, angular resolution

In this Section, we will first present a metric for the quality of the angular separation, which can be used to quantify the performance of either MUSIC or COSnet. Then, we will point out the effect of increasing the antenna separation on source resolution by MUSIC.

3.2.1 Distinguishability of Sources

In this section, the ability to distinguish sources when they are close to each other is discussed. Conclusions are drawn for both the MUSIC and ML algorithms. We already know that ML works better than MUSIC when the distance between the sources is small (less than 10 degrees). In Figure 9 we show the peaks of

$$\frac{1}{g_{MUSIC}(\theta)} = \frac{1}{a^H(\theta)V_N V_N^H a(\theta)}, \tag{8}$$

for two sources which become more and more correlated. The comparison is made for angle differences of 10°, 5°, 4°, 3°, 2° and 1°

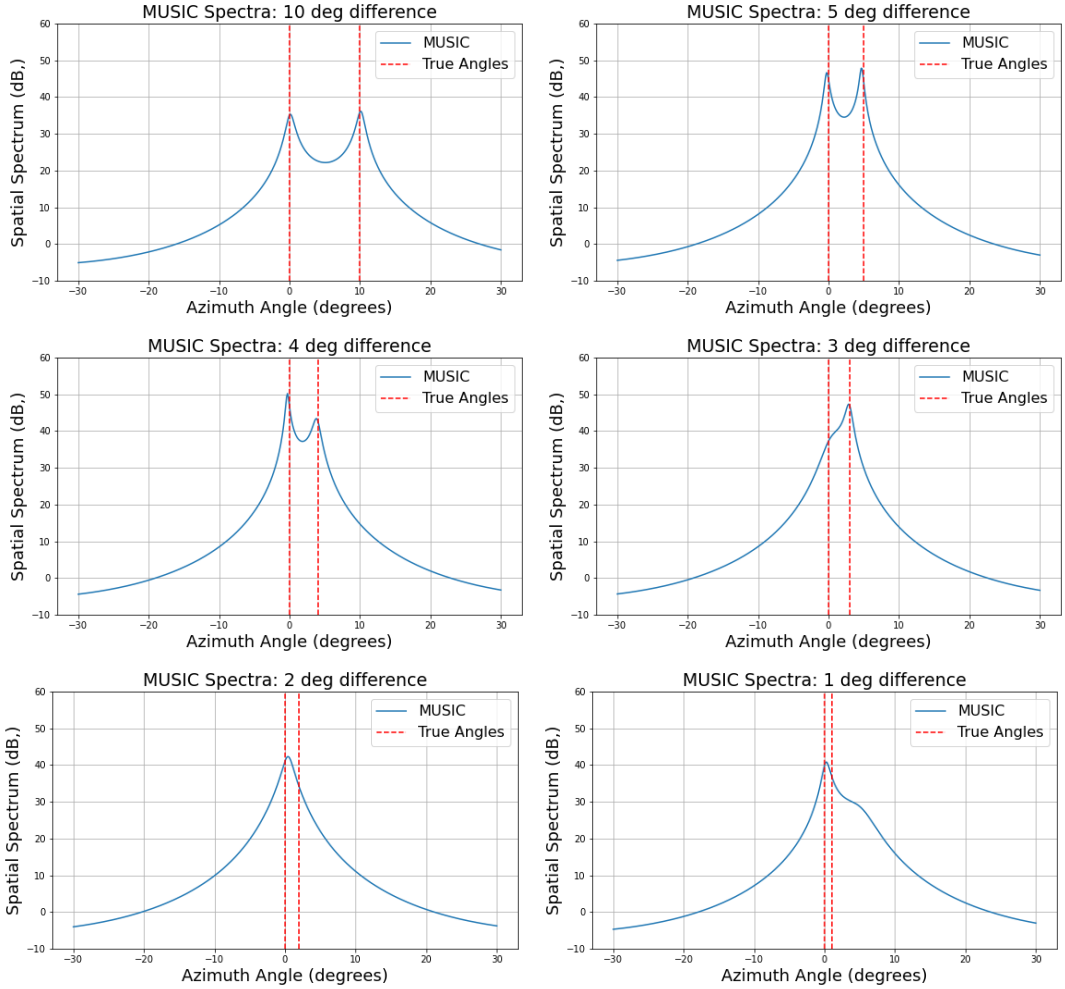

Figure 9: Comparison of the spatial spectrum for 10°, 5°, 4°, 3°, 2° and 1° difference between the sources.

The reason why the ML algorithm performs better compared to the MUSIC algorithm is unknown, although some observations to that effect are made in Section 3.8.1. For the purpose of better understanding, a metric is needed to measure the spectrum quality for both algorithms. According to the convention where quantity 8 is plotted on a logarithmic scale, the metric should also be shown on a

logarithmic scale. The metric consists of two main parts: Base size of the peaks and length of the peaks. Both of the characteristics should be big in order to have good defined peaks. A promising metric is:

$$Dist(P) = \sum_{i=1}^{Npeaks} 10 \log_{10}(\text{PeakHeight}_i * \text{PeakBaseSize}_i) \tag{9}$$

Here, PeakHeight_i is the height of the i^{th} peak and PeakBaseSize_i is the base size of the i^{th} peak. The height of the peak is the value of the corresponding maximum. The base size of the peak is:

$$size = \int_{-c}^{c} Dist(p)\, dp, \tag{10}$$

where c is a sufficiently big constant. In Figure 10 the result for Dist(P) is shown for $c = 13$. Python code for this subchapter is uploaded here.

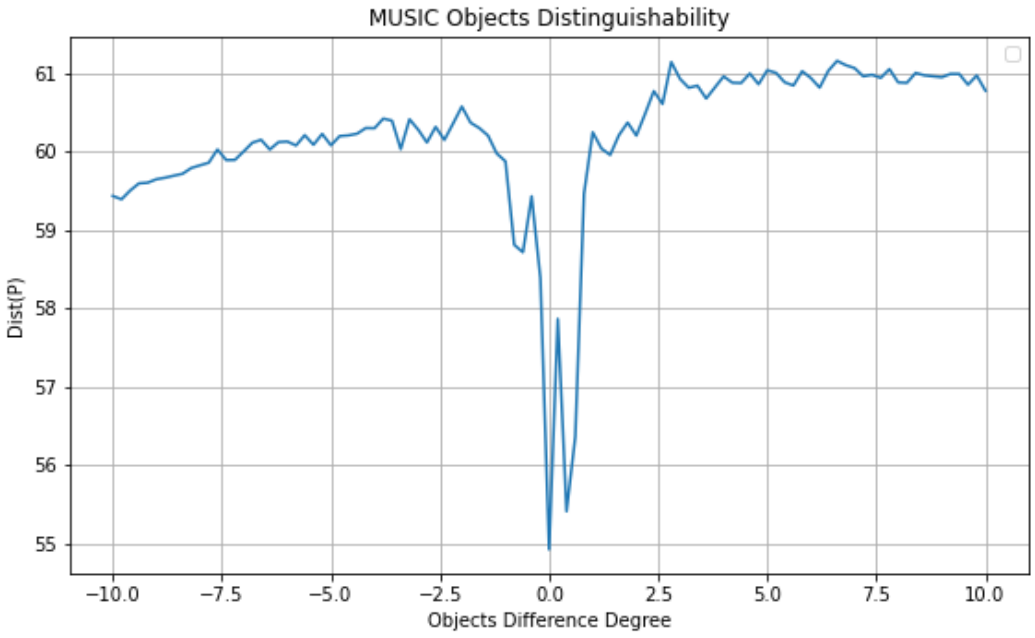

Figure 10: Distinguishability of Two Sources

3.2.2 Increasing the antenna separation improves angular resolution

In the rest of this report, the antenna separation in units of the wavelength (parameter d, the "Element Spacing (in terms of wavelength)" in code `MUSIC.m`), is set to its default value of 0.5. Here we report what happens if d is increased.

If one sets d to 1.5 instead of 0.5, one observes that the code `MUSIC.m` will consistently and accurately resolve peaks of the pseudospectrum separated by as little as 3°. Compare this statement with the result shown in the middle-right panel in Fig. 9. I.e., increasing d leads to enhanced resolution of the sources by the MUSIC algorithm. The price one pays for this enhanced resolution is the appearance of false peaks at about 45° away from the true peaks. Reducing d to, say, 1 will push those false peaks away from the true ones to about 75°. However, as long as one is interested in detecting sources in a narrow angular range, one can use a larger d to enhance the resolution.

3.3 VarPro-Style Minimization Method

We develop a method based on the VarPro methodology from exponential fitting [GP73]. From eq. (1), we are trying to solve

$$x \approx A(\vec{\theta})s, \tag{11}$$

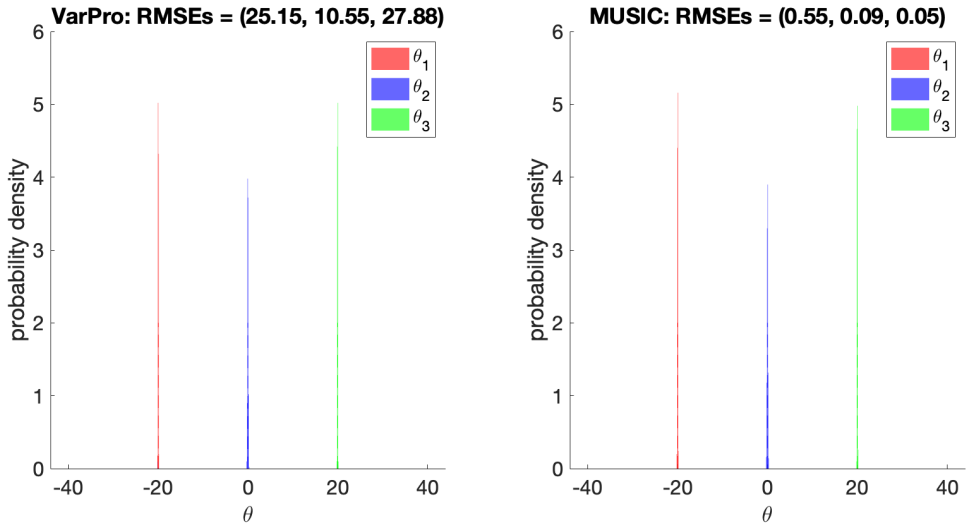

Figure 11: Comparison of VarPro and MUSIC algorithms in DoA estimation with $M = 3$ sources and $N = 4$ antennas at $-20°$, $0°$ and $20°$. For each method, we used $K = 100$ snapshots, and $d = 0.5$ with SNR = (40,40,40) for each source. RMSE triplets for $(\theta_1, \theta_2, \theta_3)$ are shown in each panel and were calculated from 500 trials.

in the least-squares sense by choosing the best $\vec{\theta}$, where $x \in \mathbb{R}^N$, $s \in \mathbb{R}^M$ and $A \in \mathbb{R}^{N\times M}$. Providing $M < N$ (i.e. the number of sources is less than the number of antennas), the least squares solution is given by the pseudo-inverse:

$$s = (A^H A)^{-1} A^H x. \tag{12}$$

Instead of minimizing $||x - A(\vec{\theta})s||$ which is a function of the unknowns θ and s, we can instead minimize

$$f(\vec{\theta}) = ||x - A(A^H A)^{-1} A^H x|| = ||(I - A(A^H A)^{-1} A^H)x||, \tag{13}$$

where $A = A(\vec{\theta})$. The objective function $f(\theta)$ can be quickly minimized using Matlab's `fmincon.m` for example. The `fmincon.m` algorithm is initiated with an initial guess. Because the final solution depends on the quality of the initial guess, 5 random initial guesses are used and the angles corresponding to the smallest value of the objective function (13) is used as the final result.

We found that MUSIC outperformed this VarPro minimization method. Although the histograms in Figure 11 look similar, the root mean-square error was larger for the VarPro method, with more outliers that were far from the ground truth. Furthermore, the MUSIC algorithm ran about 10 times more quickly, after optimizing the provided Matlab MUSIC code.

3.4 Integral Equation Approaches

In this section, we derive an integral equation that relates a continuous source distribution to an antenna density. This is a generalization of the original discrete DoA equations for discrete sources and antennas. The extra antennas could correspond to virtual sensors. For the ith antenna, the signal satisfies

$$x_i = \sum_{q=1}^{M} s_q a_i(\theta_q), \qquad i = 1, \ldots, N, \tag{14}$$

where $a_i(\theta) = e^{j2\pi d \cdot i \sin\theta}$. Now define new variables

$$\sigma(\theta_q)\Delta\theta = s_q, \tag{15}$$

$$y = i/N, \tag{16}$$

$$\chi(i/N) = x_i \tag{17}$$

where $\Delta\theta$ is the angular separation between the sources and we have normalized the antenna position so that $0 < y < 1$. One can think of $\chi(\cdot)$ as a smooth function that interpolates the values x_i. Eq. (14) becomes

$$\chi(i/N) = \sum_{q=1}^{M} \sigma(\theta_q) e^{j2\pi d \cdot i \sin\theta_q} \Delta\theta, \tag{18}$$

$$\chi(y) = \sum_{q=1}^{M} \sigma(\theta_q) e^{j2\pi \cdot yNd \sin\theta_q} \Delta\theta, \tag{19}$$

We now take the limit as $N \to \infty$, $M \to \infty$, $\Delta\theta \to 0$ and $d \to 0$ such that $Nd \to 1$. The Riemann sum becomes an integral and we obtain a Fredholm integral equation of the first kind

$$\chi(y) = \int_{-\pi}^{\pi} A(y,\theta)\sigma(\theta)d\theta, \qquad A(y,\theta) = e^{2\pi j \cdot y \sin\theta}. \tag{20}$$

We attempt to solve this integral equation numerically using two methods. First, we split it into its real and imaginary components and discretize the integrals using a trapezoidal rule. Then, we attempt to use a Beurling-LASSO method.

3.4.1 Separation into Real and Imaginary Components

Observe that χ, A, σ are complex-valued functions. We can write eq. (20) in their real and imaginary parts

$$\chi(y) = \chi_R(y) + j\chi_I(y), \tag{21}$$

$$A(y,\theta) = A_R(y,\theta) + jA_I(y,\theta), \tag{22}$$

$$\sigma(\theta) = \sigma_R(\theta) + j\sigma_I(\theta). \tag{23}$$

Equating real and imaginary parts:

$$\chi_R(y) = \int_{-\pi}^{\pi} \cos[2\pi y \sin\theta]\sigma_R(\theta)d\theta - \int_{-\pi}^{\pi} \sin[2\pi y \sin\theta]\sigma_I(\theta)d\theta, \tag{24}$$

$$\chi_I(y) = \int_{-\pi}^{\pi} \sin[2\pi y \sin\theta]\sigma_R(\theta)d\theta + \int_{-\pi}^{\pi} \cos[2\pi y \sin\theta]\sigma_I(\theta)d\theta. \tag{25}$$

This set of coupled real integral equations can be written in block form

$$\begin{pmatrix} \chi_R \\ \chi_I \end{pmatrix} = \begin{pmatrix} A_R & -A_I \\ A_I & A_R \end{pmatrix} \begin{pmatrix} \sigma_R \\ \sigma_I \end{pmatrix} \tag{26}$$

where A_R and A_I are integral operators. Note that the system has nonunique solutions. For example, when $\chi(y) = -2\pi J_1(2\pi y)$, where J_1 denotes a Bessel function of the first kind, both $\sigma(\theta) = e^{j\theta}$ and $\sigma(\theta) = j\sin\theta$ are solutions to 20. Thus, solving for the source distribution $\sigma(\theta)$ is not as simple as inverting the matrix in 26.

To obtain a numerical solution, we discretize the integral operators in 26 using the trapezoid rule and then use the Moore-Penrose pseudoinverse to solve for $\sigma(\theta)$. Symbolically, we have

$$\begin{pmatrix} \sigma_R \\ \sigma_I \end{pmatrix} = \begin{pmatrix} \cos(2\pi y \sin(\theta^T))W & -\sin(2\pi y \sin(\theta^T))W \\ \sin(2\pi y \sin(\theta^T))W & \cos(2\pi y \sin(\theta^T))W \end{pmatrix}^{+} \begin{pmatrix} \chi_R \\ \chi_I \end{pmatrix} \tag{27}$$

where $W = \Delta\theta \cdot \mathrm{diag}(1/2, 1, 1, ..., 1, 1/2)$, $^+$ denotes the pseudoinverse, and trigonometric functions are applied elementwise. To test the scheme, we prescribe $\sigma(\theta)$, obtain $\chi(y)$ using 26, and assess the scheme's ability to recover $\sigma(\theta)$ using 27. We prescribe a doubly-peaked distribution consisting of a superposition of two Gaussians of the form

$$\sigma_R(\theta) = \begin{cases} a_1 \exp\left(-\frac{(\theta-\theta_1^*)^2}{2c_1^2}\right) + a_2 \exp\left(-\frac{(\theta-\theta_2^*)^2}{2c_2^2}\right), -\pi/6 \le \theta \le \pi/6 \\ 0, \text{ otherwise} \end{cases} \tag{28}$$

$$\sigma_I(\theta) = \sin(\theta)\sigma_R(\theta). \tag{29}$$

When the prescribed source distribution has wide peaks and varies smoothly, the solution given by 27 reasonably reconstructs $\sigma(\theta)$ (see Figure 12). On the other hand, when the true source distribution has sharp peaks, approximating discrete point sources, the solution 27 has trouble reconstructing $\sigma(\theta)$, though it can correctly predict some of the peak locations (see Figure 13).

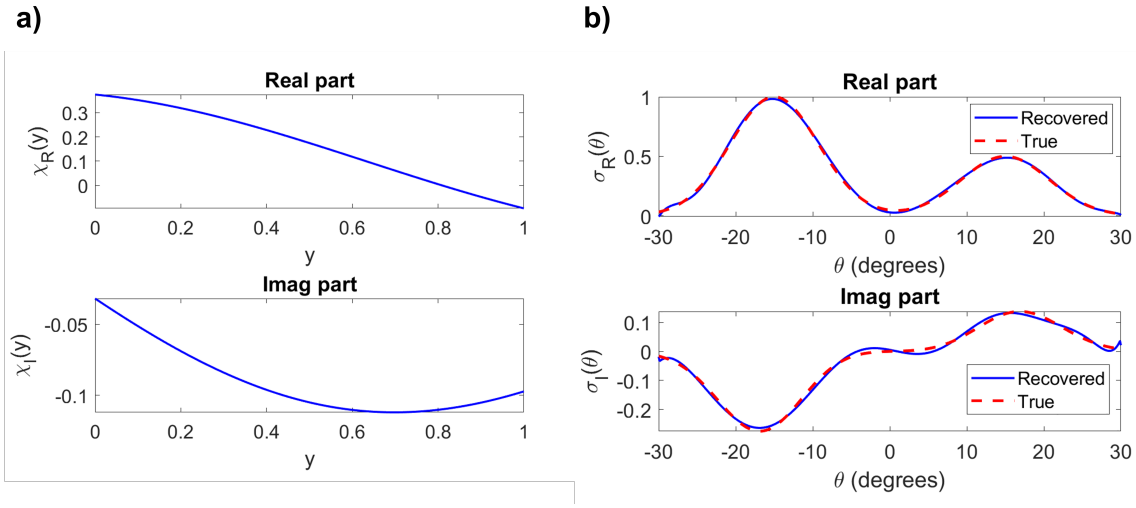

Figure 12: Reconstruction performance of the integral equation solution 27 for a smoothly varying source distribution ($\theta^*_{1,2} = \mp 15°, a_1 = 1, a_2 = 0.5, c_1 = c_2 = 0.1$). a) Real and imaginary parts of the measured signal $\chi(y)$. b) Real and imaginary parts of the true and recovered source distribution $\sigma(\theta)$.

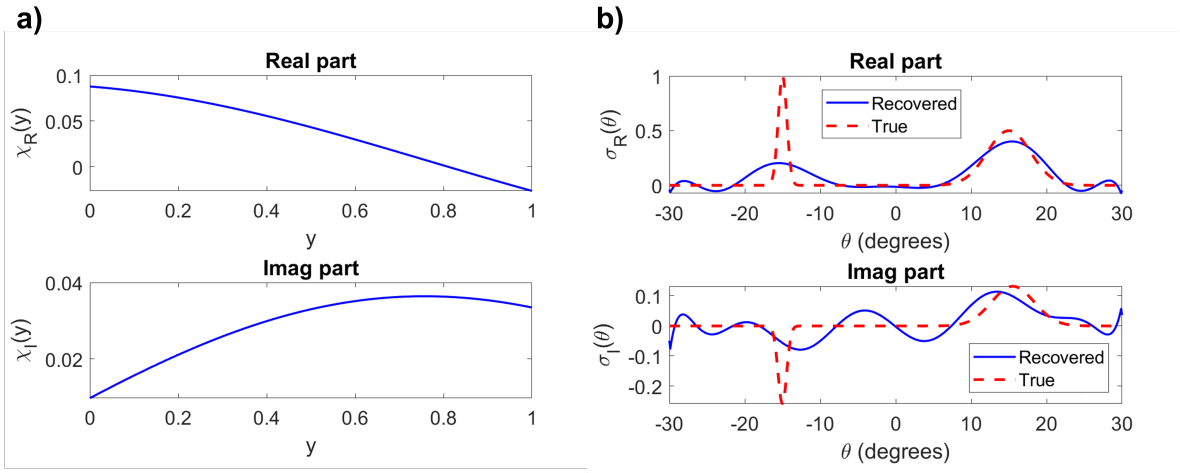

Figure 13: Reconstruction performance of the integral equation solution 27 for a sharply varying source distribution ($\theta^*_{1,2} = \mp 15°, a_1 = 1, a_2 = 0.5, c_1 = 0.01, c_2 = 0.05$). a) Real and imaginary parts of the measured signal $\chi(y)$. b) Real and imaginary parts of the true and recovered source distribution $\sigma(\theta)$.

When $\sigma(\theta) = e^{j\theta}$, $-\pi \leq \theta \leq \pi$ is used as the prescribed source distribution, $\sigma(\theta) = j \sin\theta$ is recovered instead due to the nonuniqueness of solutions to the system. This issue does not seem to arise when the sources are concentrated in a narrower range of angles, i.e. when $\sigma(\theta)$ is supported on a subinterval of $[-\pi, \pi]$.

The problem of tackling the ill-posedness of first kind Fredholm integral equations is well-studied. Several numerical methods are reviewed in [YZ19]. One popular choice is Tikhonov regularization; however, the regularization parameter α has to be chosen carefully depending on the measured signal $\chi(y)$ to ensure stability and accuracy [NP07].

3.4.2 Beurling-LASSO

The classic sparse spike recovery problem relies on ℓ^1 regularization, formerly known as LASSO in the statistical community and basis pursuit in the signal processing community. The ℓ^1 regularizer provides sparse solutions (that is, few non-zero entries) and can be computed efficiently through convex optimization. We show in a later section that the LASSO formulation is infeasible for the angle estimation problem and therefore seek to explore a new formulation based on the integral equation. In particular, we formulate the LASSO problem over a continuous domain (without resorting to discretizations over a grid) and instead use the total-variation norm, an optimization problem dubbed Beurling-LASSO or BLASSO. This formulation allows us to make exact statements about the location of the recovered spikes. For the remainder of this section, we follow (with modifcation) [DDPS18, SK22].

Definition 3.4.1 (Impulse Train)**.** *Let $\delta_{x_q} = \delta(x - x_q)$ denote the Dirac measure (delta function) centered at a point x_q. Let $\mathcal{A} \subset \mathbb{C}$. Given a grid $x_1, \dots, x_M \in \mathcal{A}$, we say that the measure $\sigma(x)$ is an **impulse train** on $\mathcal{A}$ if it is of the form*

$$\sigma(x) = \sum_{q=1}^{M} s_q \delta_{x_q} \tag{30}$$

for some complex weights $s_1, \dots, s_K \in \mathbb{C}$. We define the space of all impulse trains on $\mathcal{A}$ to be $\mathcal{M}(\mathcal{A})$.

It can be shown that for any set $\mathcal{A} \subset \mathbb{C}$ and function $f : \mathcal{A} \to \mathbb{C}$, the Dirac measure satisfies

$$\int_A f(x)\delta_z dx = f(z), \quad \forall\, z \in \mathcal{A}. \tag{31}$$

We can exploit this property directly in our continuous formulation. Suppose that our source is of the form

$$\sigma(\theta) = \sum_{q=1}^{M} s_q \delta_{\theta_q}$$

for angles $\theta_1, \dots, \theta_M \in [-\pi, \pi]$ and weights $s_1, \dots, s_M \in \mathbb{C}$. Then by the aforementioned property of Dirac measures,

$$\chi(y) = \int_{-\pi}^{\pi} A(y,\theta)\sigma(\theta)d\theta = \int_{-\pi}^{\pi} A(y,\theta) \sum_{q=1}^{M} s_q \delta_{\theta_q} d\theta = \sum_{q=1}^{M} s_q \int_{-\pi}^{\pi} A(y,\theta)\delta_{\theta_q} d\theta \tag{32}$$

$$= \sum_{q=1}^{M} s_q A(y, \theta_q). \tag{33}$$

Observe that under this assumption on the source, we nearly recover the original discrete formulation. If we discretize in the antenna position y, we obtain the discrete formulation exactly. Thus, this assumption allows us to formulate a continuous problem that produces an equivalent solution to the discrete problem. We now define the total variation norm to develop the optimization problem.

Definition 3.4.2. *For a bounded Radon measure σ, we define the total variation norm on $[-\pi, \pi]$*

$$\|\sigma\|_{TV}([-\pi,\pi]) = \sup_{\psi \in \mathcal{C}_0} \left\{ \int_{-\pi}^{\pi} \psi \, d\sigma \;\middle|\; \|\psi\|_{\infty,[-\pi,\pi]} \le 1 \right\}. \tag{34}$$

Here, $\mathcal{C}_0$ is the space of continuous functions that vanish at infinity and for $\psi \in \mathcal{C}_0$,

$$\|\psi\|_{\infty,[-\pi,\pi]} = \sup_{x \in [-\pi,\pi]} |\psi(x)|. \tag{35}$$

While this is a very theoretical definition, for $\sigma \in M([-\pi, \pi])$ an impulse train, we indeed have

$$\|\sigma\|_{TV}([-\pi,\pi]) = \|s\|_1 = \sum_{q=1}^{M} |s_q| \tag{36}$$

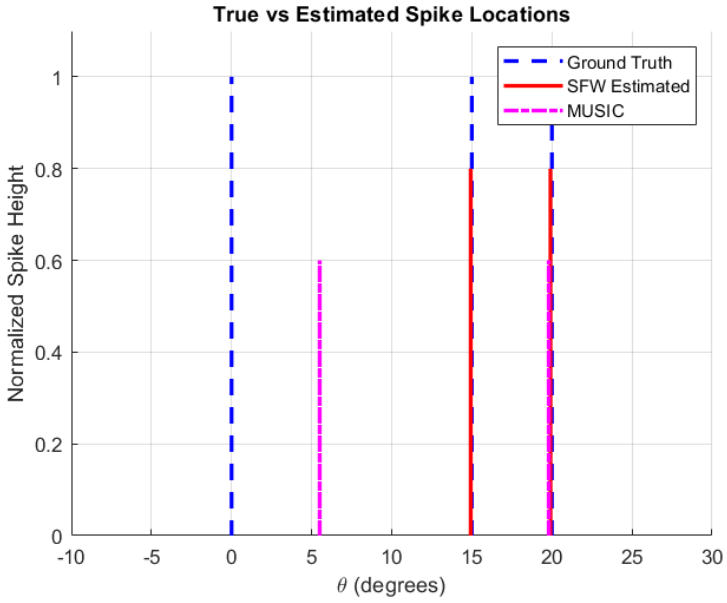

Figure 14: A comparison between the Sliding Frank-Wolfe (SFW) and MUSIC algorithms for the angle estimation problem. We choose three sources at angles $\theta = 0, 15, 20$ with SNR $= 10, 15, 20$, respectively. 4 sensors are used and 300 snapshots in time to generate the noisy signal. Shown here are the locations of the true sources (blue), the SFW estimated sources (red), and MUSIC estimated sources (pink). The corresponding amplitudes are shown for differentiating the spikes and do not have any physical relevance to the problem.

where $s = (s_1, \ldots, s_M)^T \in \mathbb{R}^M$ which shows that the total variation norm generalizes the ℓ^1 norm to the continuous setting of measures. For an impulse train source $\sigma(\theta)$, we denote it's integral transform under $A(y, \theta)$

$$(\Phi\sigma)(y) = \int_{-\pi}^{\pi} A(y, \theta)\sigma(\theta)d\theta \tag{37}$$

Given (noisy) measurements $\chi(y)$ of the above, we consider the problem

$$\min_{\sigma \in \mathcal{M}([-\pi,\pi])} \frac{1}{2}\|\Phi\sigma - \chi\|_2^2 + \lambda\|\sigma\|_{TV} \tag{38}$$

where $\|\cdot\|_2$ is the L^2 norm and $\lambda > 0$ is a regularization parameter. Note that this formulation is not restrictive to the kernel

$$A(y, \theta) = \exp(2\pi j y \sin(\theta)) \tag{39}$$

and can hold for arbitrary kernels $A(y, x)$. For instance, the Fourier transform kernel. In our problem, the spike locations (that is, the locations of the Dirac measures in the impulse train) precisely correspond to the angle location – provided we can obtain them accurately.

Following [DDPS18], we utilize the Sliding Frank-Wolfe algorithm (also called the Conditional Gradient Method) to solve this continuous formulation for the optimal spike locations and amplitudes. For a reference to the algorithm, see Algorithm 2 in [DDPS18] which successfully applies this approach to Laplace and correlation kernels and obtains accurate spike locations. Their code can be obtained from https://github.com/qdenoyelle. We attempt an implementation in Matlab, though the results are not stable (they are only successful about half of the time). However, we suspect that this is not an issue with the algorithm (and its theory) as it has been proven to work extremely well for other kernels (and should extend to ours the same), but rather an issue with either the implementation or the underdetermined nature of the system (we have a small number of sensors). To access our implementation, see https://github.com/AnshDesai1/BLASSO. Going forward, if enough time is dedicated to debugging and developing a BLASSO solver, we believe that it will be a very strong solution for the angle estimation problem.

3.5 Sparse Recovery of Sources

3.5.1 Regularized Least Squares

Suppose we are given N sensors and M sources as known quantities. We seek to recover the DoA angles of the sources. Recall that the received signals are generated by $x_k = As_k + n_k$ as defined in equation (1). Suppose we have a range of angles of interest, $[\theta_{\min}, \theta_{\max}]$. To generate a least-squares problem, we will discretize this interval: given a grid size parameter L, let $\Delta\theta = \frac{\theta_{\max} - \theta_{\min}}{L}$ and $\theta_l = \theta_{\min} + l\Delta\theta$ for $l = 0, 1, \ldots, L$. Next, we use these θ_l values to generate a matrix of candidate steering vectors

$$\hat{A} = \begin{bmatrix} | & | & & | \\ a(\theta_0) & a(\theta_1) & \ldots & a(\theta_L) \\ | & | & & | \end{bmatrix} \tag{40}$$

where $a(\theta)$ is defined in equation (3). One method of sparse recovery is then to solve the least squares problem

$$\min_{\mathbf{t}} \|\hat{A}\mathbf{t} - \mathbf{x}_k\|_2^2 + \lambda\|\mathbf{t}\|_1 \tag{41}$$

where λ is a regularization parameter. If all of the entries of these matrices/vectors were real-valued, this would be a classic LASSO problem, however, we must consider complex-valued matrices and solutions.

Sparse enumeration for least squares. While inefficient for large M and L, for toy versions of the problem it is possible to solve this via enumeration. Specifically, given M and L, there are $\binom{L}{M}$ possible sparsity patterns for the vector $\mathbf{t}$.

For simplicity of exposition, let us focus on the case that $M = 4$. In this case, each sparsity pattern corresponds to a tuple (i, j, k, l) indexing the nonzero entries of $\mathbf{t}$, which we label t_i, t_j, t_k, and t_l. For a fixed i, j, k, and l, our least squares problem becomes

$$\min \left\| \begin{bmatrix} | & | & | & | \\ \mathbf{a}_i & \mathbf{a}_j & \mathbf{a}_k & \mathbf{a}_l \\ | & | & | & | \end{bmatrix} \begin{bmatrix} t_i & t_j & t_k & t_l \end{bmatrix}^\top - \mathbf{x} \right\|_2^2 \tag{42}$$

If we let A_{ijkl} be the matrix with columns shown above, this is a standard (unregularized) least squares problem with solution $t_{ijkl} = (A_{ijkl}^H A_{ijkl})^{-1} A_{ijkl}^H \mathbf{x}$. We would like to understand the properties of this problem to determine whether the solution to the sparse reconstruction problem will do a good job identifying our θ values of interest.

Numerical Experiments. Our first question is whether this method will recover θ without any regularization ($\lambda = 0$)? Our preliminary experiments suggest that the answer is no. Below are two examples showing that the ground truth solution will not necessarily produce the best approximation of the vector $\mathbf{x}$. In these examples, we let $M = 4$ (4 sources), $\theta \in [-20, 20]$, and $L = 21$ (number of discrete θ values to try). We set the source angles to be $[-12, -4, 4, 12]$. Note that these angles align with our discretized θ values, ensuring that our steering vectors will contain the exact ground truth vectors. We use SNR [25, 25, 25, 25]. In this case, there are $\binom{21}{4} = 5985$ possible sparsity patterns with 4 nonzero entries.

In Figure 15, we compare the results for $N = 4$ and $N = 6$ sensors. When $N = M = 4$ all subsets of 4 steering vectors approximate the signal to within machine accuracy. This is what we would expect, since any selection of 4 steering vectors will span $\mathbb{R}^4$. Unfortunately, it means that in general we cannot characterize the direction of arrival angle by simple minimizing mean squared-error (even when constrained to only considering sparse solutions). When $N = 6$, there is more differentiation, since not every set of 4 steering vectors will contain the signal in its span. However, we can see that steering vectors associated with the ground truth angles of arrival do not minimize the mean squared error. Thus, even when restricted to sparse solutions with the correct number of nonzero entries, least-squares recovery of the directions of arrival is not well-defined.

Regularization. Even though we have already restricted our possible solutions to sparse vectors, we can explore whether regularization could improve the recovering of the correct θ values. One easy place to start is by considering l^1 regularization in addition to restricting the solution to vectors with

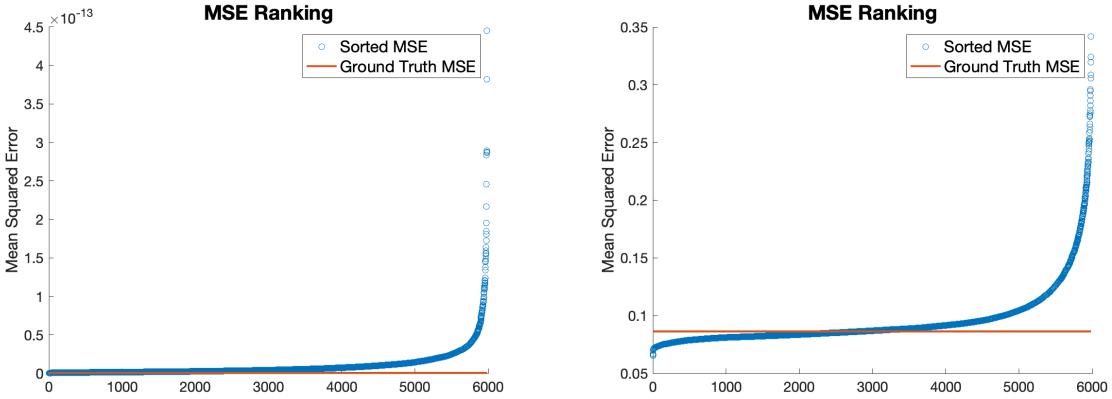

Figure 15: Mean squared error between reconstructed signal to the actual signal for $M = 4$ sources and $N = 4$ sensors (left) versus $N = 6$ sensors (right).

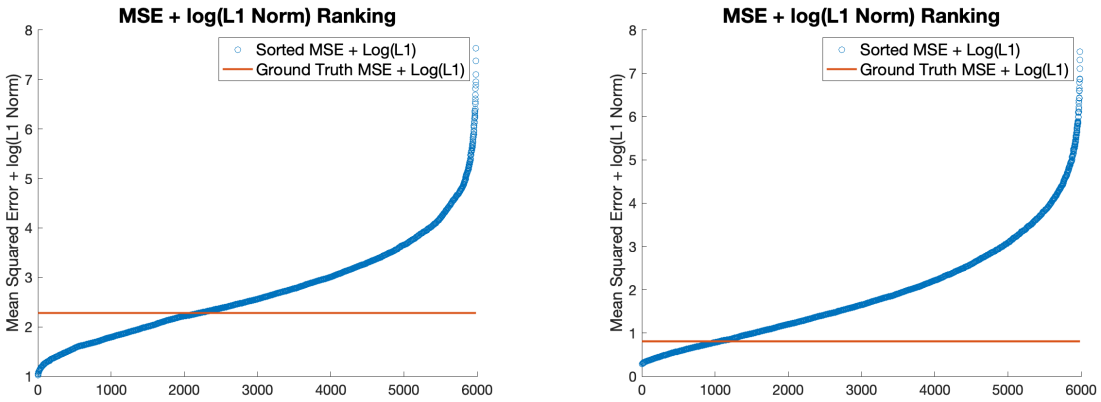

Figure 16: Sum of mean squared error and log of the l^1 norm of the solution for for $M = 4$ sources and $N = 4$ sensors (left) versus $N = 6$ sensors (right).

M nonzero entries. In figure 16, we show the results of looking at the combined MSE and l^1 norm of potential solutions. Even with the additional regularization, the ground truth set of steering vectors do not minimize the optimization objective (MSE + the log of the l^1 norm.

While this single example is not promising, there is some randomness involved in the generation of the signal, and thus one realization may not be representative. Additionally, this does not give use information about what steering vectors would be selecting if we minimize MSE or regularized MSE. To investigate both of these topics, we ran the above sparsely enumerated least squares for 50 random realizations of a generated signal and show the frequency with which each angle was identified across those realizations. Figures 17 and 18 summarize the results for $N = 4$ and $N = 6$ antennae respectively.

Results of preliminary regularized least squares. While there are reasons to expect that l^1 regularized least squares will not be able recover the correct angles of arrival, we still tested a simplified version of the algorithm. Here, to address the complex nature of our least squares problem, we define

$$\hat{A} = \hat{A}_1 + j\hat{A}_2, \qquad \mathbf{t} = \mathbf{t}_1 + j\mathbf{t}_2 \qquad \mathbf{x} = \mathbf{x}_1 + j\mathbf{x}_2, \tag{43}$$

which allows us to equation (41) as

$$\min \left\| \begin{bmatrix} A_1 & -A_2 \\ A_2 & A_1 \end{bmatrix} \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t_2} \end{bmatrix} - \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \right\|_2^2 + \lambda \left\| \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t_2} \end{bmatrix} \right\|_{1,1/2}. \tag{44}$$

Here we define $\|\cdot\|_{1,1/2}$ to be $\sum_i \sqrt{t_{1i}^2 + t_{2i}^2}$ where i indexes the elements of the vectors $\mathbf{t}_1$ and $\mathbf{t}_2$.

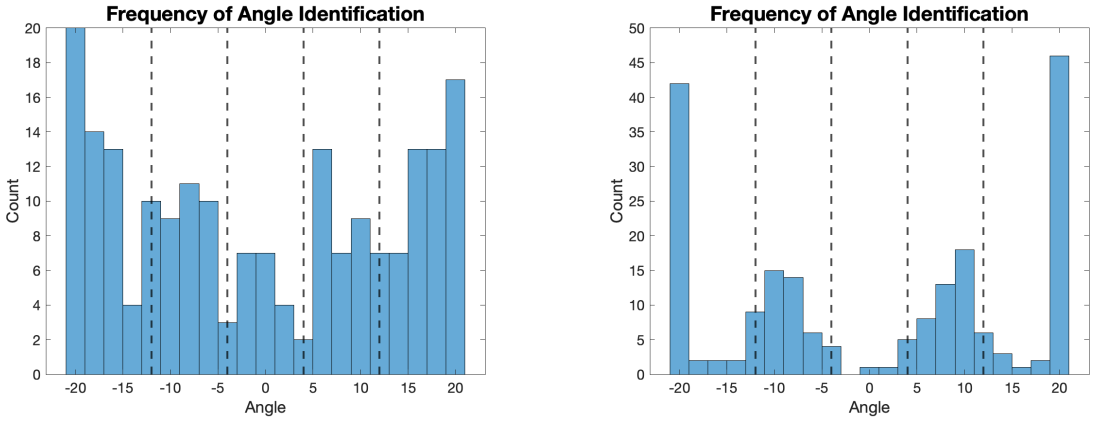

Figure 17: Frequency with which an angle was identified as being part of the set of steering vectors that minimized the least squares objective (left) or regularized objective (right) for $M = 4$ sources and $N = 4$ sensors. Black dotted lines show ground truth DoA angles. The sparsely enumerated least squares is not effective at identifying DoA angles.

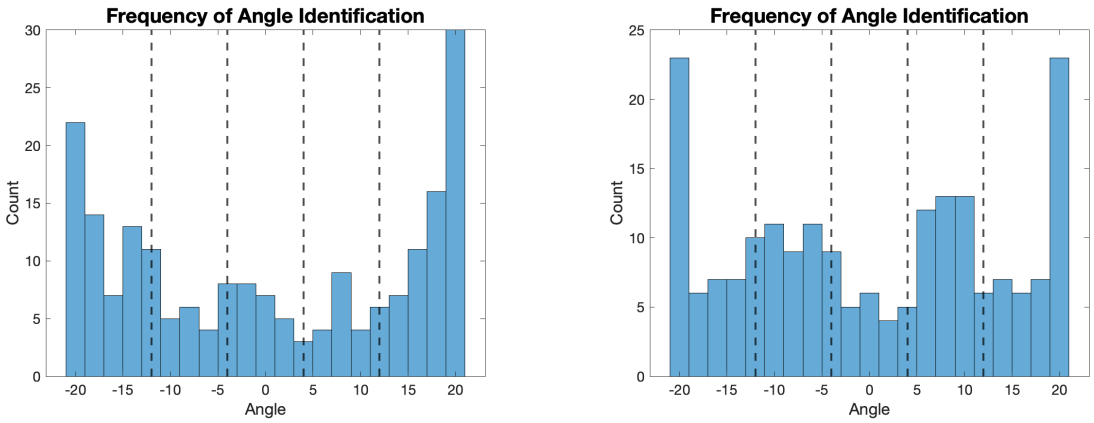

Figure 18: Frequency with which an angle was identified as being part of the set of steering vectors that minimized the least squares objective (left) or regularized objective (right) for $M = 4$ sources and $N = 6$ sensors. Black dotted lines show ground truth DoA angles. The sparsely enumerated least squares is not effective at identifying DoA angles.

While not identical, for preliminary results, we solve the related problem

$$\min \left\| \begin{bmatrix} A_1 & -A_2 \\ A_2 & A_1 \end{bmatrix} \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t_2} \end{bmatrix} - \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \right\|_2^2 + \lambda \left\| \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t_2} \end{bmatrix} \right\|_1 . \tag{45}$$

which is a standard l^1 regularized problems. Across 5 initial realizations, the number of nonzero entries in the recovered vector $\mathbf{t}$ ranged from 4 to 8 and most commonly contained or was close to 2 or 3 of the ground truth angles but rarely identified all four and often contained false positives. Further investigation could look at alternative regularization strategies or implementing the minimization in equation (44) as written.

3.6 Upper Limits of Source Detection

We have observed that MUSIC can detect up to $M = N - 1$ sources whereas COSnet can be trained to detect $> N - 1$ sources. In this section, we attempt to explain this improved behavior from the neural network.

3.6.1 Paths in the Complex Plane

An in-depth comparison of Raytheon's MATLAB code to the algorithms provided in their problem proposal slides was required to root cause the reason COSNet is able to find more solutions than the MUSIC algorithm's theoretical maximum. A brief review of the MUSIC algorithm, outlined in Figure 1, starts with the reception of a data vector by n receivers. This data vector becomes a complex data matrix $X = X_{data}$ of size n by m with m being the number of samples taken in time.

$$X_{data} = X = A(\theta)s + n_{noise}$$

This data matrix is then right multiplied by it's Hermitian and divided by the total number of samples to form the Estimated Correlation Matrix R which contains all real eigenvalues. A singular value decomposition (SVD) of the Correlation Matrix R provides the Eigenvector Matrix U which corresponds to the signal space and the noise space of R.

$$SVD(XX^H) = SVD(\mathbf{R}) = U\Lambda U^H$$

The intensity of an incoming signal is significantly greater than that of noise which will cause their corresponding eigenvalues to be of similar difference in magnitude. Therefore, one can rethink the signal space as the solution space and the noise space as the null space of a general linear algebra problem.

$$U = \begin{bmatrix} U_s U_n \end{bmatrix}$$

It is not possible to reconstruct the exact location of an incoming signal using the solution space alone so the MUSIC algorithm capitalizes on the orthonormal characteristics of the null component of the eigen vector matrix U_n by right multiplying this sub matrix with its Hermitian to produce the X_{MU} matrix. When this matrix is right multiplied by the steering vector $a(\theta)$ and left multiplied by $a(\theta)^H$, the result is the function $\tilde{p}(\theta)$ (6) that will go to zero as θ approaches a value in the solution space and go to ∞ otherwise. It therefore follows that the inverse of $\tilde{p}(\theta)$ produces the function $P_1(\theta)$ that will go to ∞ as θ approaches a value in the solution space and to zero otherwise.

$$P_1(\theta) = \frac{1}{a(\theta)^H (U_n U_n^H) a(\theta)} = \frac{1}{a(\theta)^H X_{MU} a(\theta)} = \frac{1}{\tilde{p}(\theta)}$$

This $P_1(\theta)$ solution directly aligns to the equation provided on slides 15 and 16 of Raytheon's problem proposal presentation. However, in Raytheon's LoadAndTestSerrogateMatrix.m script the application of this equation diverges with an absolute value of $\tilde{p}(\theta)$ resulting in the following applied equation for the MUSIC algorithm.

```
RTX MATLAB CODE: plot(az_angs, 10*log10(abs(Pmus)))
```

$$\boxed{P_{MU}(\theta) = \frac{1}{|\tilde{p}(\theta)|}}$$

Although this small change does not impact the performance MUSIC algorithm, it does have a significant impact when the COSNet's solution matrices X_{ML} are used in place of MUSIC's X_{MU}. The Estimated Correlation Matrix R was used to train the COSNet Neural Network using the MUSIC algorithm's $P_{MU}(\theta)$ for the cost function resulting in the X_{ML} which has complex eigenvalues. The original equation defined by Raytheon's problem proposal slides directly replaces X_{MU} with X_{ML} without an absolute value.

$$P_2(\theta) = \frac{1}{a(\theta)^H X_{ml} a(\theta)} = \frac{1}{g(\theta)}$$

However, after further review of Raytheon's LoadMatrixAndTestSurrogateMatrix.m script, a similar divergence from theory was discovered. Taking the absolute value of $P_2(\theta)$ transfers COSNet's solution from the complex plane to the Cartesian plane.

```
RTX CODE: plot(az_angs, 10*log10(abs(Psurr)))
```

$$P_{ML}(\theta) = \frac{1}{|a(\theta)^H X_{ml} a(\theta)|} = \frac{1}{\sqrt{(a^H X_{ml}^H a)(a^H X_{ml} a)}} = \boxed{\frac{1}{\sqrt{g^*(\theta) g(\theta)}}}$$

Focusing on the complex function $g(\theta)$ from COSNet's X_{ML}, and comparing against the real function $\tilde{p}(\theta)$ from MUSIC's X_{MU} matrix, provides the first indication of a divergence in solution methodology between the two algorithms. The second indication is illustrated with a direct comparison between the eigenvalues of MUSIC's Estimated Correlation Matrix R and the absolute values of the eigenvalues of COSNet's X_{ML}. This is achieved visually in Figure 19 via condition number of the respective matrices.

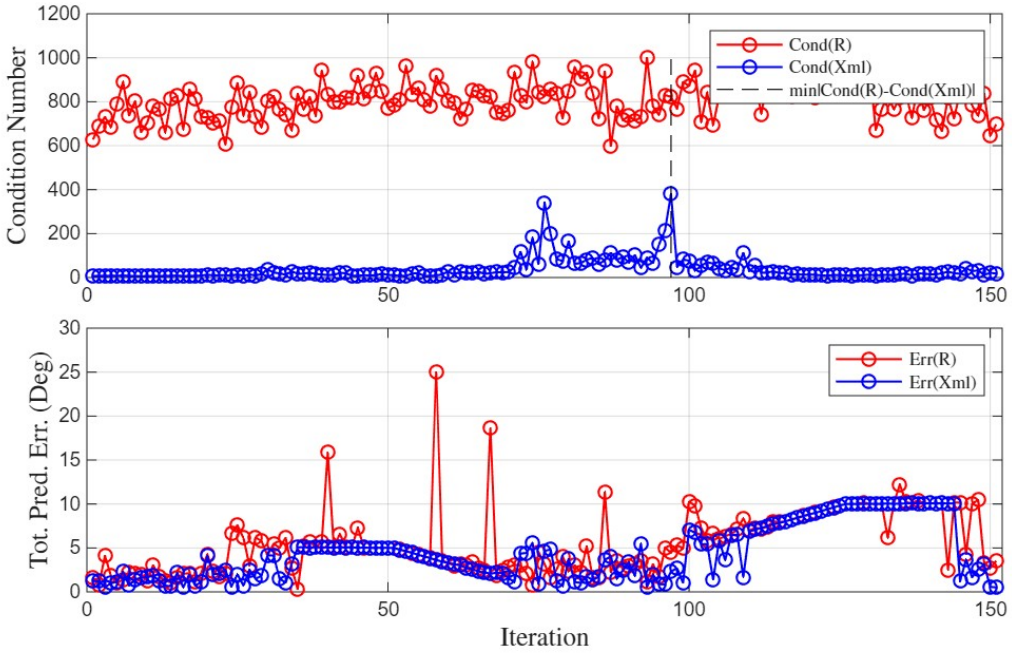

Figure 19: Condition Number and Total Error

As expected, the MUSIC algorithm had large condition numbers ranging between 600 and 1000 while most of the condition numbers for COSNet were less than 10 with a few spikes up to 400 between iterations 75 and 100. This shows that the MUSIC algorithm requires large condition numbers to work properly while COSNet does not show a correlation between accuracy and condition number magnitude. This provides the final evidence to support the hypothesis that COSNet is achieving solutions using poles on the Complex plane while MUSIC solves solutions via zeros on the Cartesian plane.

As shown in Figure 20, COSNet achieves a closed curve (-90 to 90 deg) in the Complex plane (Left) where solutions are identified in the Cartesian plane (Right) as the Complex function passes by the origin. While MUSIC is constrained to a maximum of $n-1$ solutions, n being the number of receivers on the array, it is unknown how many solutions are possible with COSNet. While Figure 20 highlights a three angled solution at the iteration corresponding to the maximum COSNet condition number, Figure 19, a fourth solution provided by COSNet is manifested outside Raytheon's area of interest.

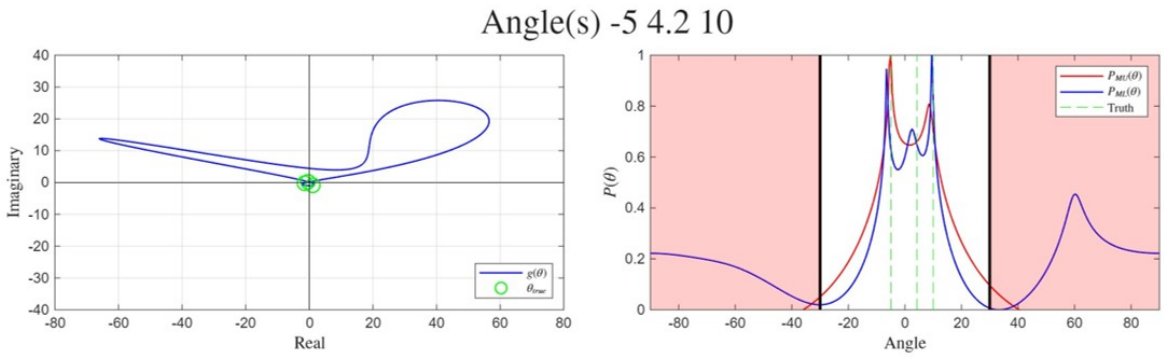

Figure 20: Iteration 97, Complex (Left) vs. Cartesian (Right)

As shown in Figure 21, when the solution set is pushed to four for the COSNet algorithm, an additional two poles are achieved outside of the area of interest indicating the possiblity that more than four solutions are possible with COSNet.

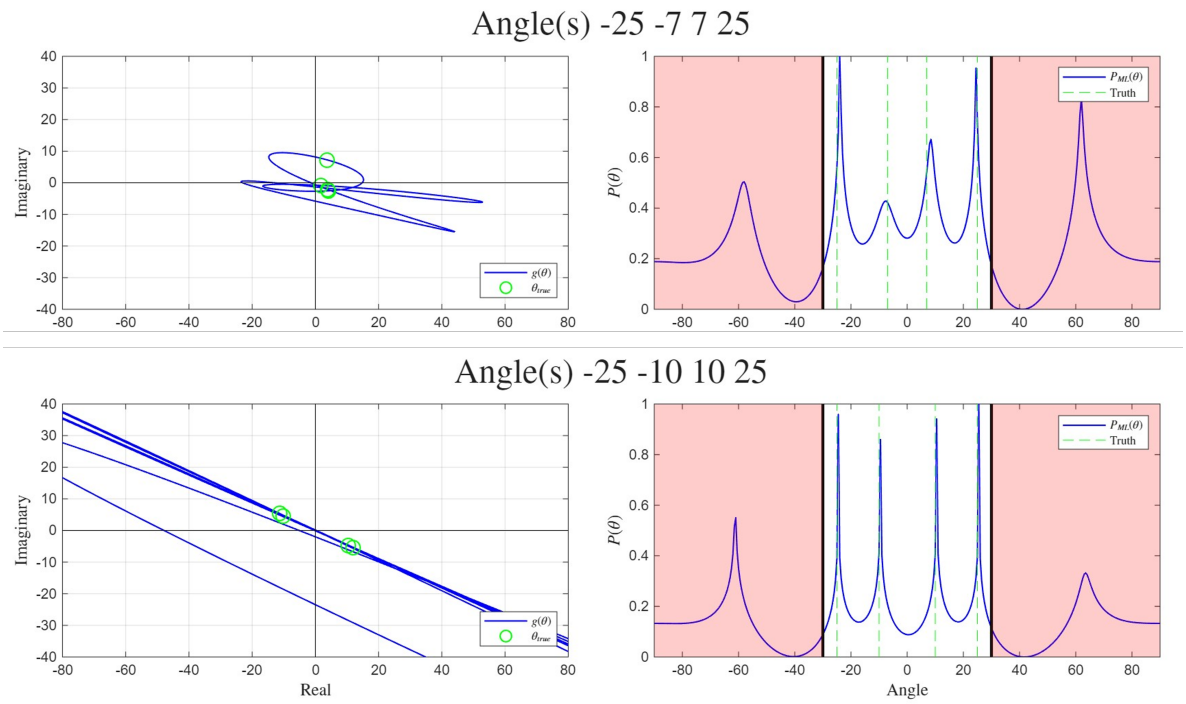

Figure 21: Four Solutions

To test if COSNet can find more than four sources, it is advisable that Raytheon constructs a data matrix, X_{data}, that contains more than four sources. This can be achieved using Raytheon's current four source X_{data} generation algorithm twice, each with their own set of four angles. The two X_{data} matrices are then concatonated to achieve a large X_{data} matrix, containing eight different sources, prior to right multiplying with the large X_{data}'s hermitian to achieve the estimated correlation matrix R.

Overall, COSNet achieved less total error 69.54 percent of the time indicating a near 20 percent improvement during this simulation. The large increases in error in Figure 19 indicate the inability of either algorithm to find all three solutions.

3.6.2 Functions in the Complex Plane

When determining directions of arrival, we are attempting to find the angle θ that minimizes $|g_{MUSIC}|$ and $|g_{ML}|$ where these functions are given by

$$g_{MUSIC}(\theta) = a^H(\theta) X_{MUSIC} a(\theta), \qquad X_{MUSIC} = V_N V_N^H, \tag{46}$$

$$g_{ML}(\theta) = a^H(\theta) X_{ML} a(\theta) \tag{47}$$

In Figure 22, we see that g_{MUSIC} has zero imaginary part and has *minima* near the incident angles. It is clear that $g_{MUSIC}(\theta)$ must be real when $\theta \in [-\pi, \pi]$ since $g_{MUSIC} = ||V_N^H a||^2$ In contrast, g_{ML} has a non-zero imaginary part and *simple zeros* near the incident angles. It is tempting to think that $Re[g_{ML}]$ is the derivative of $Re[g_{MUSIC}]$ but there is no simple relationship between the two functions.

The steering vectors are defined as

$$a = (1, z, z^2, \ldots, z^{N-1}), \qquad z = e^{j2\pi d \sin\theta}, \tag{48}$$

where N is the number of antennas. We now think of z as a complex number with modulus 1 and angle $\alpha(\theta) = 2\pi d \sin\theta$. If $d < 1/2$, z does not traverse the entire circle $|z| = 1$. For $X \in \mathbb{C}^{N \times N}$, we

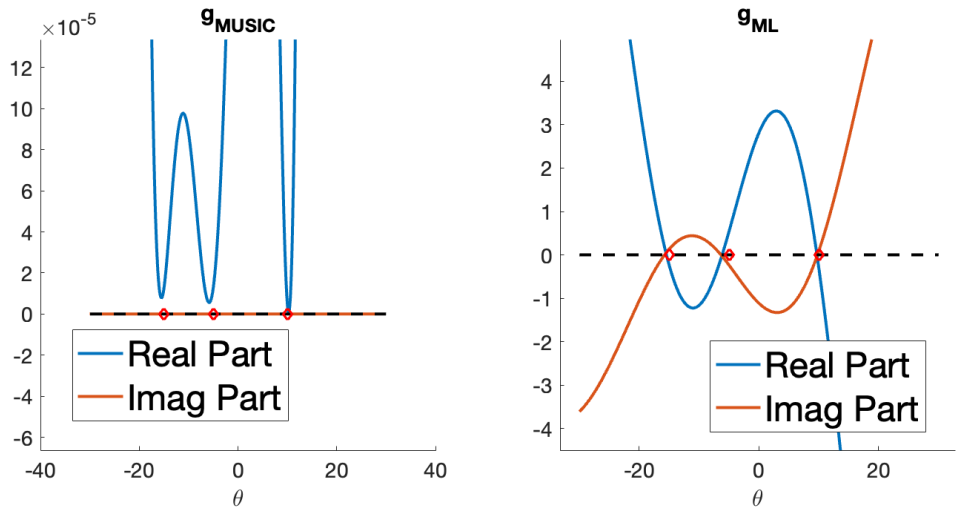

Figure 22: Real and Imaginary parts of the $g_{MUSIC}(\theta)$ (left) and $g_{ML}(\theta)$ (right).

define a complex function

$$g(z, X) = (1, z^{-1}, z^{-2}, \ldots, z^{-(N-1)}) X \begin{pmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^{N-1} \end{pmatrix}. \tag{49}$$

so that $g_{MUSIC}(z) = g(z, X_{MUSIC})$ and $g_{ML}(z) = g(z, X_{ML})$. These functions are now considered to be functions on the complex plane, analytic on $\mathbb{C}\backslash\{0\}$:

$$g_{MUSIC}(z) = \sum_{l=-(N-1)}^{N-1} b_l z^l, \tag{50}$$

$$g_{ML}(z) = \sum_{l=-(N-1)}^{N-1} c_l z^l, \tag{51}$$

and the coefficients b_l and c_l are related to the entries in X_{MUSIC} and X_{ML}.

While the functions (50) and (51) both have $2N-2$ roots, only M of these are "physical" and correspond to angles of arrival. In fact, both MUSIC and COSnet are attempting to construct complex functions with zeros at

$$z_k^* = \exp(j2\pi d \sin\theta_k), \qquad k = 1, \ldots, M. \tag{52}$$

These roots correspond to actual incident angles, always lie on the circle $|z| = 1$ and have phase $\alpha(\theta_k)$. In general, $g_{ML}(z)$ will have $2N-2$ distinct roots. However, X_{MUSIC} is Hermitian with the result that $b_l = \bar{b}_{-l}$, where the bar denotes complex conjugate. Therefore, the roots of g_{MUSIC} always occur as *reciprocal conjugates*: z is a root $\Rightarrow 1/\bar{z}$ is a root.

The loss function in COSnet evaluates a cross-entropy over a given range of θ values. Since COSnet is, in effect, trying to determine the coefficients of the function $g_{ML}(z)$ in the complex plane, we recommend that the neural network be trained on the range $-90° \leq \theta \leq 90°$ so that $\sin\theta \in [-1, 1]$ in eq. (48). If a longer arc of $|z| = 1$ is provided to COSnet, it is provided with more information, and should be able to learn $g_{ML}(z)$ more effectively.

In Figure 23, we show the complex functions (50) and (51) with $N = 4$ antennas and $M = 3$ sources. In Figure 23 (left), we see that $g_{MUSIC}(z)$ appears to only have 3 roots. Because the roots exist as reciprocal conjugates, these roots are either double roots, or are very close to (and on either side of)

the circle $|z| = 1$. Either way, they manifest numerically as 3 peaks in the function $1/|g_{MUSIC}(\theta)|$. Note that all three double roots of g_{MUSIC} correspond to three actual angles of arrival.

In the example of Figure 23 (right), $g_{ML}(z)$ clearly has 6 roots. This has several important consequences. First, COSnet must position the 3 "extra" roots somewhere in the complex plane. If these roots are close enough to $|z| = 1$, they may give rise to false directions of arrival. Second, with $N = 4$ antennas, the theoretical maximum number of incident angles detectable is $2N-2 = 6$. COSnet could be trained to detect up to 6 distinct angles and position all 6 roots near or on $|z| = 1$.

The matrix X_{MUSIC} is Hermitian while X_{ML} is generally not Hermitian. We were asked to explore the roots of $X_{ML}X_{ML}^H$ and $X_{ML} + X_{ML}^H$ in an attempt to "Hermitianize" the matrix outputted by COSnet. The results are shown in Figure 24. In Fig. 24 (left), we see that the roots of $X_{ML}X_{ML}^H$ are not related in any obvious way to X_{ML} or X_{MUSIC}. A simple 2×2 example illustrates that the complex functions associated with X_{ML} and $X_{ML}X_{ML}^H$ are very different:

If $X_{ML} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ with $a, b, c, d \in \mathbb{C}$, then $X_{ML}X_{ML}^H = \begin{pmatrix} |a|^2 + |b|^2 & a\bar{c} + b\bar{d} \\ \bar{a}c + \bar{b}d & |c|^2 + |d|^2 \end{pmatrix}$ and

$$g(z, X_{ML}) = \frac{c}{z} + (a + d) + bz, \tag{53}$$

$$g(z, X_{ML}X_{ML}^H) = \frac{\bar{a}c + \bar{b}d}{z} + |a|^2 + |b|^2 + |c|^2 + |d|^2 + (a\bar{c} + b\bar{d})z. \tag{54}$$

It is clear that $g(z, X_{ML})$ and $g(z, X_{ML}X_{ML}^H)$ are very different functions with different root structures. In Fig. 24 (right), we see that the root structure of $X_{ML} + X_{ML}^H$ appears to be very similar to that of X_{ML}. The positions of the unphysical zeros have not changed and are still close to $|z| = 1$. Therefore, using $X_{ML}X_{ML}^H$ or $X_{ML} + X_{ML}^H$ as alternatives to X_{ML} is not recommended.

We believe that the non-Hermitian nature of X_{ML} is actually a *strength*, not a weakness, for the COSnet algorithm. X_{ML} has several advantages over X_{MUSIC}:

- The extra roots of $g_{ML}(z)$ can be used to detect more sources. COSnet can theoretically be trained to detect up to $2N - 2$ sources, *double* the number that MUSIC is capable of. Although $g_{MUSIC}(z)$ also has $2N - 2$ roots, they appear either as double roots or reciprocal conjugates close to $|z| = 1$. Numerically, in the line search over θ, these $2N - 2$ roots appear as $N - 1$ DoA angles. The Hermitian nature of X_{MUSIC} restricts the number of angles that MUSIC can detect.

- Because the zeros of $g_{ML}(z)$ are simple, the peaks of $1/|g_{ML}(\theta)|$ are generally sharper than those of $1/|g_{MUSIC}(\theta)|$, allowing more certainty in the angle estimates. This is consistent with the numerical evidence from Figures 6, 7, and 8, indicating that COSnet is able to better distinguish directions of arrival that have small angular separation (spatially correlated signals). However, these properties are contingent on the positions of the unphysical zeros (see below).

Controlling the placement of unphysical roots

One issue with COSnet is that the remaining "unphysical" roots associated with $g_{ML}(z)$ can lie near $|z| = 1$, resulting in the contamination of the signal for the true angles of arrival, or even predicting false directions of arrival. This can be remedied by imposing constraints on the X_{ML} matrix via $g(z, X_{ML})$. In the case with $N = 4$ antennas and $M = 3$ sources, we see in Figure 23 (right) that we have 3 unphysical zeros. COSnet can be trained to position these zeros far from $|z| = 1$ by imposing the linear constraints

$$g(w_i, X_{ML}) = 0, \qquad i = 1, 2, 3, \tag{55}$$

where w_1, w_2, w_3 are complex numbers far from $|z| = 1$. For example, one could take $w_1 = 10$, $w_2 = -10$, $w_3 = 10j$. Eqs. (55) manifest as linear constraints on the entries of X_{ML}:

$$(1, w_i^{-1}, w_i^{-2}, \ldots, w_i^{-(N-1)}) X_{ML} \begin{pmatrix} 1 \\ w_i \\ w_i^2 \\ \vdots \\ w_i^{N-1} \end{pmatrix} = 0. \tag{56}$$

If these constraints can be imposed on COSnet while it is being trained, the unphysical roots of $g_{ML}(z)$ would be "safely" placed far from $|z| = 1$ so that the remaining physical zeros can be easily

and accurately detected. Of course, if $2N-2$ sources are being detected, no constraints are needed since all roots correspond to directions of arrival.

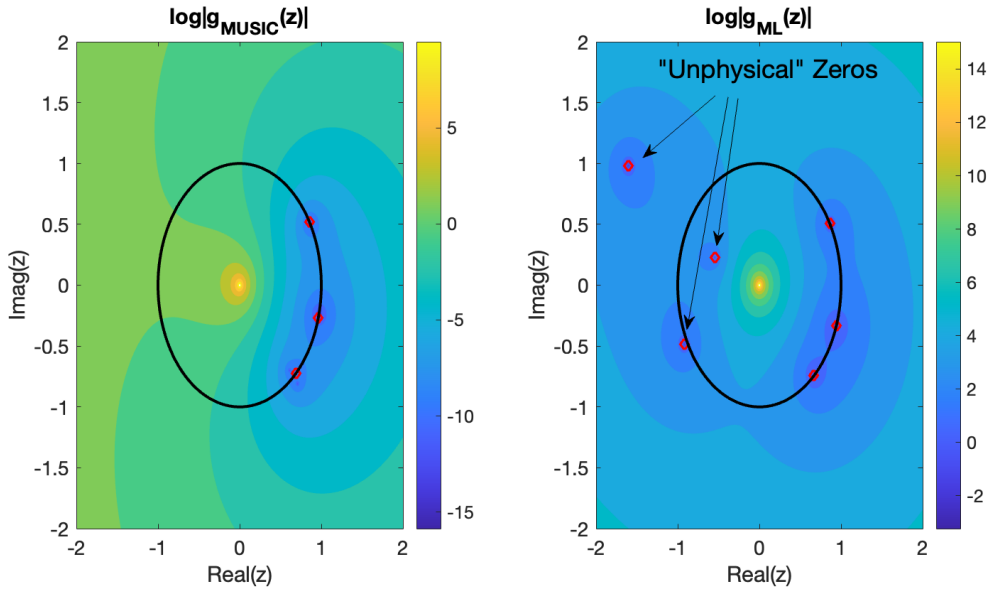

Figure 23: The logarithms of $|g_{MUSIC}(z)|$ and $|g_{ML}(z)|$, showing zeros of $g_{MUSIC}(z)$ and $g_{ML}(z)$ in the complex plane. Note also the singularity at $z=0$ as well as the presence of three roots in $g_{ML}(z)$ that have no physical meaning. For both MUSIC and COSnet, the true angles are at $(\theta_1,\theta_2,\theta_3)=(-15°,-5°,10°)$, SNR $=(25,\ 25,\ 25)$, $M=3$ sources, $N=4$ antennas and $K=200$ snapshots.

3.7 Spectral Analysis of $\mathbf{X}_{ML}$ and $\mathbf{X}_{MUSIC}$

Spectral analysis was pursued as a means of detecting characteristic differences between the matrix produced by the MUSIC algorithm, X_{MUSIC}, and the surrogate matrix produced by COSnet (machine learning), X_{ML}.

3.7.1 Eigenvalue Comparison

Eigendecomposition was first attempted, where the eigenvalues of each matrix were plotted in dB to compare the ability to detect sources versus noise. Most notably, X_{ML} has complex eigenvalues—a result of it not being Hermitian. The magnitudes of X_{ML}'s eigenvalues were plotted and scaled by the maximum value to better qualitatively compare the sets. A representative case is shown below.

Figure 25 shows that X_{ML} does a better job at detecting sources, which is exemplified by having two markers at relatively large values. X_{MUSIC}, however, does a better job at detecting noise, which is exemplified by having two markers at relatively small values. There is a relatively large gap between X_{MUSIC}'s eigenvalues that should be detecting sources strongly rather than noise, while there is a relatively small gap between X_{ML}'s eigenvalues that should be detecting noise strongly rather than sources. The gap between the first and second dominant eigenvalues of X_{MUSIC} is explained later in Sec. 3.9.2.

It is important to note that in cases where sources are placed at the same azimuth, X_{ML} errantly distinguishes between the two sources, resulting in two large eigenvalues rather than just one; X_{MUSIC} correctly detects one source in cases like this shown in Figure 26.

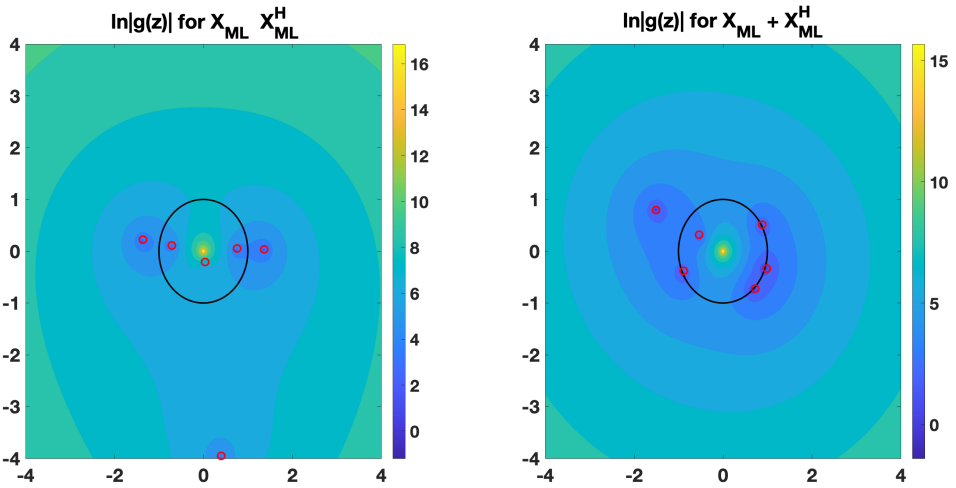

Figure 24: Complex functions $g(z, X_{ML}X_{ML}^H)$ and $g(z, X_{ML} + X_{ML}^H)$ corresponding to $X_{ML}X_{ML}^H$ (left) and $X_{ML} + X_{ML}^H$ (right): see eq. (49). Roots are indicated by red circles. For both MUSIC and COSnet, the true angles are at $(\theta_1, \theta_2, \theta_3) = (-15°, -5°, 10°)$, SNR = (25, 25, 25), $M = 3$ sources, $N = 4$ antennas and $K = 200$ snapshots.

3.7.2 Pseudospectra and Zeros of $g(\theta)$

In this section, our setup involves setting two fixed angles (at $-5°$ and $10°$) and sweeping a third angle across the domain from $-15°$ to $5°$. The following plots are selected snapshots of these angle sweeps. We compared plots of g_{ML}^{-1} and g_{MUSIC}^{-1} in what we denote as pseudospectra plot. These plots are visuals for the DoA estimation in each respective method (i.e. the MUSIC algorithm and the COSNET result). We also looked at corresponding plots of g_{ML} and g_{MUSIC} which we call zeros of $g(\theta)$. This is the same as inverting the pseudospectra plot. Because X_{MUSIC} is constructed from noise information and is consequently Hermitian, it was proposed to extract a Hermitian matrix from X_{ML} that can be used for analysis instead. In what is referred to as "Hermitian decomposition," X_{ML} was written as the sum of a Hermitian matrix and a skew-Hermitian matrix.

The X_{ML} matrix can be written as a sum of its Hermitian, X_H, and skew-Hermitian, X_{SH}, parts:

$$X_{ML} = X_H + X_{SH}, \qquad X_H = \frac{1}{2}\left(X_{ML} + X_{ML}^H\right), \quad X_{SH} = \frac{1}{2}\left(X_{ML} - X_{ML}^H\right). \tag{57}$$

Matrices X_H and X_{SH} satisfy the relations

$$X_H^H = X_H, \qquad X_{SH}^H = -X_{SH}, \tag{58}$$

which are found by taking the Hermitian conjugate of the last two expressions in (57).

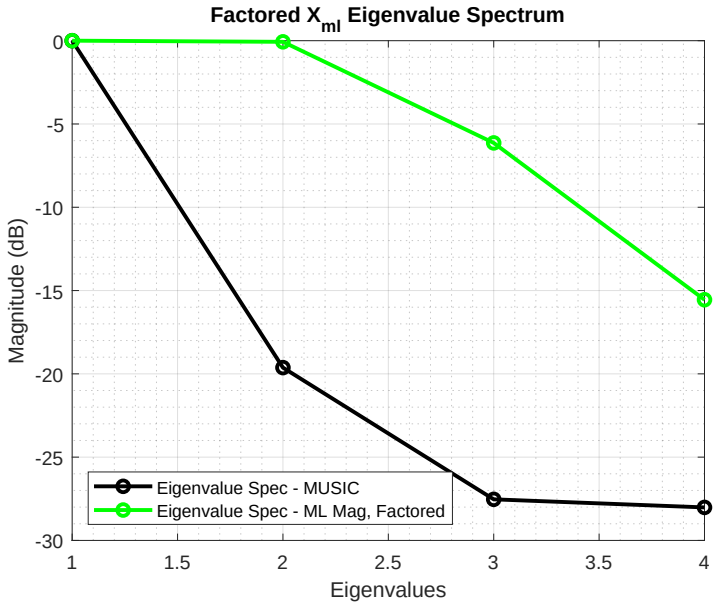

Figure 25: Eigenvalue Comparison of MUSIC and COSnet algorithms in DoA estimation with $M = 2$ sources at $0°$ and $3°$ and $N = 4$ antennas. For each method, we used $K = 100$ snapshots, and $d = 0.5$ with SNR $= 25$ for each source. The x-axis is an enumeration of the eigenvalues from largest to smallest.

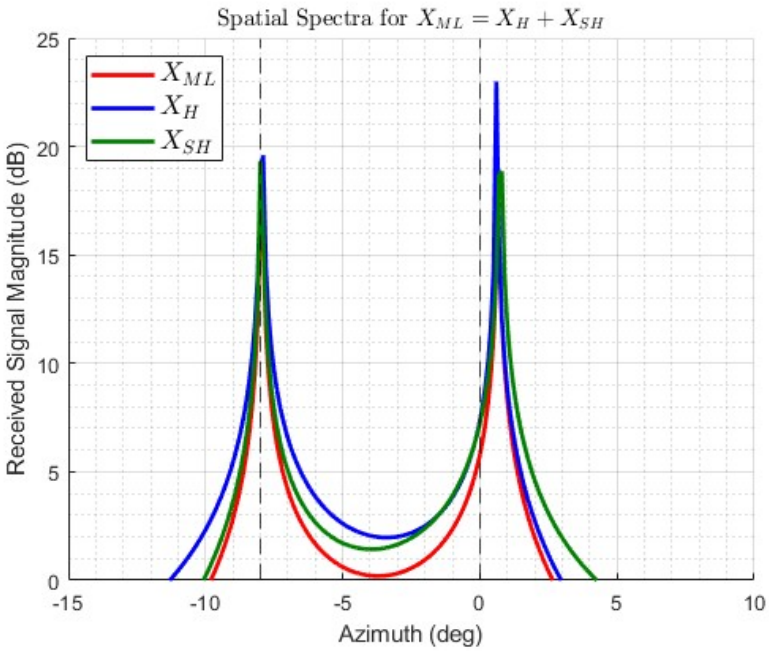

Figure 27: Pseudospectra plot with signal sources at $\theta = -8°$ and $\theta = 0°$ and equal signal-to-noise ratios.

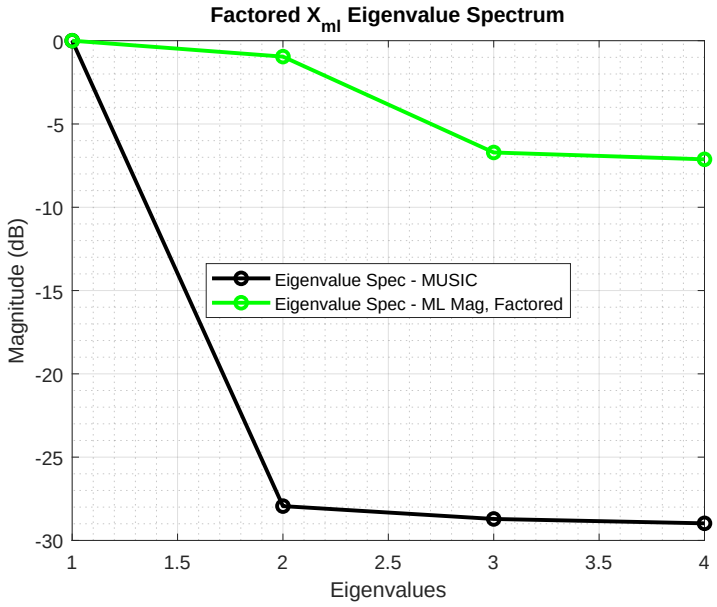

Figure 26: Overlapping Sources: Eigenvalue Comparison of MUSIC and COSnet algorithms in DoA estimation with $M = 2$ sources at $0°$ and $0°$ and $N = 4$ antennas. For each method, we used $K = 100$ snapshots, and $d = 0.5$ with SNR $= 25$ for each source.

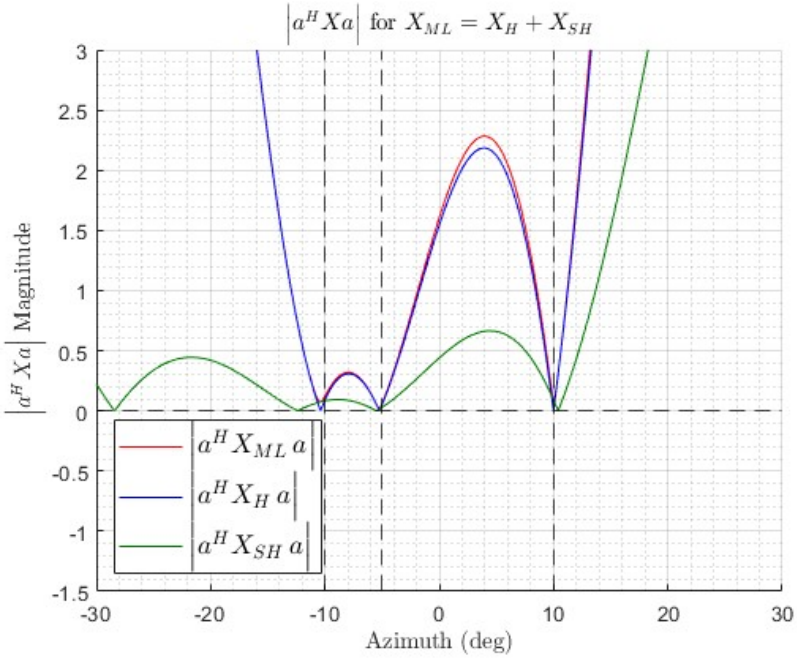

Figure 28: Plot of zeros of g_{ML} and g_{MUSIC} (Inverse plot of Pseudospectra) with signal sources at $\theta = -10°, -5°$, and $10°$

A major observation from the pseudospectra in Figures 27 and 28 is that the peaks formed using both the Hermitian and skew-Hermitian parts of $\mathbf{X}_{ML}$ are sharper than the peaks from $\mathbf{X}_{ML}$ itself. From our experiments, the Hermitian part does good job of recognizing the number of signals in general whereas the skew-Hermitian part sometimes recognizes more sources than there actually are (see Figure 28). However, we note that the skew-Hermitian part does a better job than the Hermitian part when identifying overlapping signals (see Figure 27.

3.7.3 Suggestions on Future Directions

After performing spectral analysis on the $\mathbf{X}_{ML}$ and $\mathbf{X}_{MUSIC}$ in varying perspectives, we conclude with a few insights on suggestions for future work. The plots in Section 3.7.2 indicate that the Hermitian or skew-Hermitian parts of the $\mathbf{X}_{ML}$ matrix individually do the heavy-lifting in determining the angles of the signals. This leads us to believe that imposing a Hermitian or skew-Hermitian condition on resulting matrices from COSNET may improve DoA estimation via sharper, more precise peaks and more closely mimic the $\mathbf{X}_{MUSIC}$ matrix. Alternatively, one may consider applying weighting to the Hermitian and skew-Hermitian parts of $\mathbf{X}_{ML}$, depending on which feature of using each part discussed in Section 3.7.2 is desired.

3.8 Fast computation of the X_{ML} matrix

The results of Sec. 3.7 suggest that either X_H alone or X_{SH} alone can be used in place of the whole X_{ML} and often performs just as well as the latter. Motivated by this observation, below we will first show *theoretically* that, indeed, a certain Hermitian or skew-Hermitian matrix X alone can produce pseudospectra

$$P(\theta) = 10 \log_{10} \frac{1}{|a(\theta)^H \, X \, a(\theta)|} \tag{59}$$

with peaks at the correct angles of the sources. In (59), $a(\theta)$ is the steering vector defined in (3) and X denotes the aforementioned counterpart of X_{ML}. Second, we will exhibit a problem that such a matrix has, which is producing peaks at wrong (i.e., unrelated to the locations of the sources) angles. Finally, we will demonstrate how this problem can be overcome. All of these steps have been implemented in Matlab codes `compute_Xml.m` and `construct_Xml_improved.m`, which take just a couple of seconds to run.

3.8.1 Algorithm for the computation of X

We begin by arguing that a Hermitian and a skew-Hermitian matrix of the same dimension have the same number of independent entries. For concreteness, below we will use 4×4 matrices to illustrate all our statements. The general forms of a Hermitian and an skew-Hermitian matrix are:

$$X_H = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{12}^* & x_{22} & x_{23} & x_{24} \\ x_{13}^* & x_{23}^* & x_{33} & x_{34} \\ x_{14}^* & x_{24}^* & x_{34}^* & x_{44} \end{pmatrix}, \qquad X_{SH} = \begin{pmatrix} j\,y_{11} & y_{12} & y_{13} & y_{14} \\ -y_{12}^* & j\,y_{22} & y_{23} & y_{24} \\ -y_{13}^* & -y_{23}^* & j\,y_{33} & y_{34} \\ -y_{14}^* & -y_{24}^* & -y_{34}^* & j\,y_{44} \end{pmatrix}, \tag{60}$$

with all x_{ll} and y_{ll} being real. Multiplying X_H by j, one observes that jX_H and X_{SH} have exactly the same form. Hence a Hermitian and a skew-Hermitian matrix of the same dimension have the same number of independent entries.

Our next step is to take a look at the expression in the denominator of (59), i.e.,

$$g(\theta) \equiv a(\theta)^H (X_H + X_{SH})\, a(\theta) = a(\theta)^H X_H \, a(\theta) + a(\theta)^H X_{SH} \, a(\theta) \equiv g_H(\theta) + g_{SH}(\theta), \tag{61}$$

defined similarly to g_{ML} in (47). We will show that the task of enforcing the roots of $g(\theta)$ to be at specified angles θ_m, $m = 1, \ldots, M$ can be accomplished with *either* X_H or X_{SH}. Moreover, employing both of these matrices for this task *together* may, in general, degrade the result.

We note that

$$g_H(\theta) \text{ is purely real} \qquad \text{and} \qquad g_{SH}(\theta) \text{ is purely imaginary.} \tag{62}$$

These relations follow from the general identity, valid for any vectors $\vec{a}$, $\vec{c}$ and matrix B (of appropriate dimensions):

$$\left(\vec{a}^{\,H} B \, \vec{c}\right)^* = \left(\vec{c}^{\,T} B^T \, \vec{a}^{\,*}\right)^* = \vec{c}^{\,H} B^H \, \vec{a}, \tag{63}$$

and letting $\vec{c} = \vec{a}$ and using relations (58). Then from (61), (62) one has

$$|g(\theta)| = \sqrt{|g_H(\theta)|^2 + |g_{SH}(\theta)|^2}. \tag{64}$$

This equation illustrates two points:
(i) If X_H is chosen so as to have $g_H(\theta_m) = 0$, for some θ_m, the latter condition cannot be improved by choosing X_{SH} in any way.
(ii) Moreover, it can be *degraded* if X_{SH} is not chosen carefully enough so as to make $g_{SH} = 0$ at *exactly* $\theta = \theta_m$.
Thus, to enforce $g(\theta)$ to have zeros at the specified angles θ_m, it *suffices* to consider either $X = X_H$ or $X = X_{SH}$. Choosing one or the other does not affect the number of independent entries of the matrix (see the sentence after (60)), and therefore we use $X = X_H$ until the end of this subsection.

We will now derive the equations satisfied by the entries x_{kl} in (60). Denote components of vectors $a(\theta_m)$, where $m = 1, \ldots, M$ (M is the number of sources), by a_{nm}, where $n = 1, \ldots, N$ (N is the number of antennas). From (3):

$$a_{nm} = e^{j2\pi\, n\, d\, \sin\theta_m}. \tag{65}$$

Then, using $X = X_H$ from (60), one has:

$$\begin{aligned} a(\theta_m)^H X a(\theta_m) \;=\; & |a_{1m}|^2(x_{11} + x_{22} + x_{33} + x_{44}) + \\ & 2\mathrm{Re}\big[a_{1m}^* a_{2m}\big]\,(x_{12,R} + x_{23,R} + x_{34,R}) - 2\mathrm{Im}\big[a_{1m}^* a_{2m}\big]\,(x_{12,I} + x_{23,I} + x_{34,I}) + \\ & 2\mathrm{Re}\big[a_{1m}^* a_{3m}\big]\,(x_{13,R} + x_{24,R}) - 2\mathrm{Im}\big[a_{1m}^* a_{3m}\big]\,(x_{13,I} + x_{24,I}) + \\ & 2\mathrm{Re}\big[a_{1m}^* a_{4m}\big]\,x_{14,R} - 2\mathrm{Im}\big[a_{1m}^* a_{4m}\big]\,x_{14,I}, \end{aligned} \tag{66}$$

where $x_{kl,R} = \mathrm{Re}[x_{kl}]$, $x_{kl,I} = \mathrm{Im}[x_{kl}]$. In obtaining (66) we have used the following corollaries of (65) valid for all $n = 1, \ldots, N$, $k = 1, \ldots,$ [as appropriate, see below], and $m = 1, \ldots, M$:

$$|a_{nm}|^2 = 1, \qquad a_{1m}^* a_{(1+k)m} = a_{2m}^* a_{(2+k)m} = \cdots = a_{(N-k)m}^* a_{Nm}. \tag{67}$$

The expressions in (66) represent the left-hand side (l.h.s.) of the system of linear equations which we will discuss below. To specify the right-hand side (r.h.s.), we will now count the number of independent variables in that linear system. These independent variables will be the following combinations of x_{kl}:

$$\begin{aligned} E_1 &= x_{11} + x_{22} + x_{33} + x_{44}, \\ E_2 &= 2(x_{12,R} + x_{23,R} + x_{34,R}), \qquad E_3 = -2(x_{12,I} + x_{23,I} + x_{34,I}), \\ E_4 &= 2(x_{13,R} + x_{24,R}), \qquad E_5 = -2(x_{13,I} + x_{24,I}), \\ E_6 &= 2x_{14,R}, \qquad E_7 = -2x_{14,I}. \end{aligned} \tag{68}$$

(Cosmetic note: the notation E_p, $p = 1, \ldots, 7$ stands for a combination of x_{kl} which plays the role of an Effective variable.) Thus, there are $7 = 1 + 2 \cdot (4 - 1)$ effective variables that determine the entries of X in the $N = 4$ example considered above. For the general N, there are $2N - 1$ effective variables made of the N real components x_{kk} and $N(N-1)/2$ complex components x_{kl} for $k \neq l$.

Note that the individual components, say, x_{11}, x_{22}, etc. are not defined by (68). Therefore, there is a freedom of setting them equal; similarly, one can set, without loss of generality, $x_{12} = x_{23} = x_{34}$, $x_{13} = x_{24}$, thereby making X a Toeplitz matrix. Alternatively, one can set only the first row and column of X to contain nonzero entries, with the rest of the matrix populated by zeros. Either choice will eventually result in the same function $g(\theta)$.

We now discuss how the r.h.s. of the linear system can be set. Let us begin by considering the case where the number of antennas N equals the number of sources M; so, $\underline{N = M}$. Clearly, there must be M conditions requiring

$$g(\theta_m) \equiv a(\theta_m)^H X a(\theta_m) = 0, \qquad m = 1, \ldots, M \tag{69}$$

at all source angles θ_m. We then have $(2N - 1) - M = M - 1$ free variables and thus can prescribe that many more equations. We want $g(\theta)$ to *not* vanish at angles other than θ_m, and we can prescribe this condition at some $M - 1$ *extra* angles θ_e. We can set $g(\theta_e)$ to any nonzero numbers, and we have arbitrarily set this number to 1:

$$g(\theta_e) \equiv a(\theta_e)^H X a(\theta_e) = 1, \qquad e = 1, \ldots, M - 1 \tag{70}$$

Locations θ_e can also be set arbitrarily; for example, one can set them to be the midpoints between consecutive θ_m and θ_{m+1}.

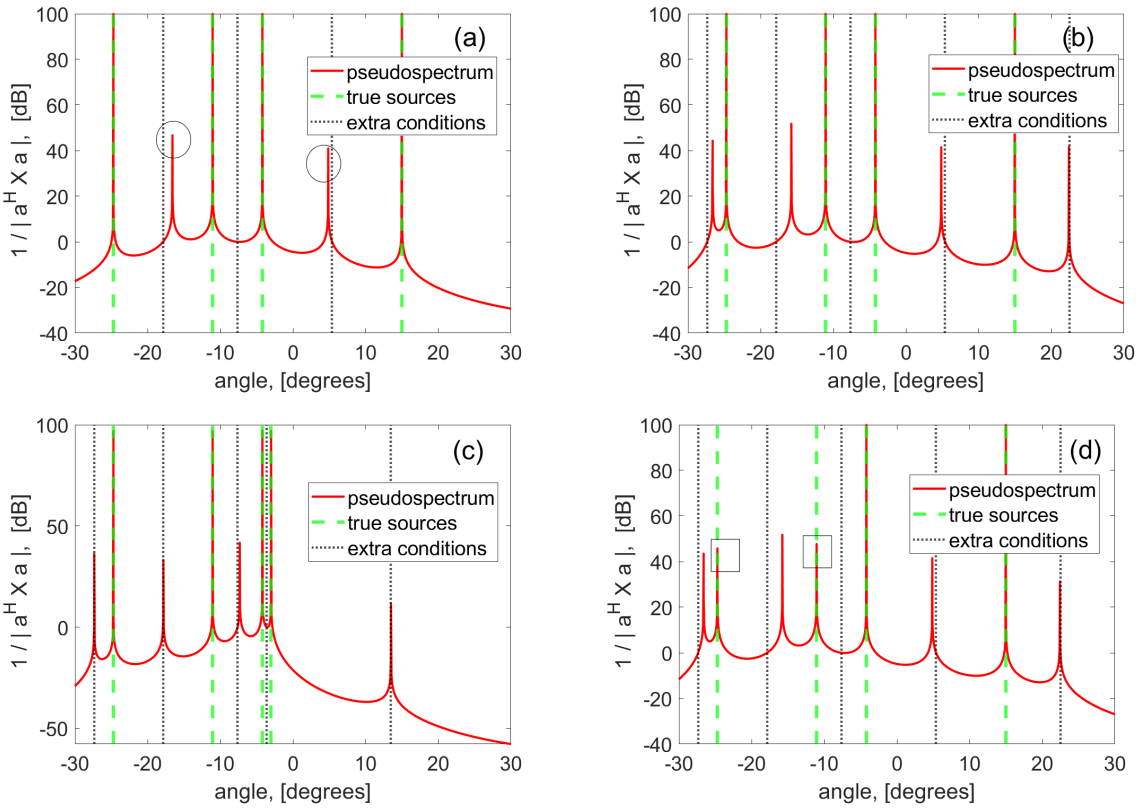

Figure 29: Pseudospectra computed with matrix X calculated by the algorithm of Sec. 3.8.1. The 'extra conditions' mentioned in the legend are the θ_e where condition (70) holds. Sources are located at $\theta_m = \{-24.7, -11.05, -4.2, 15\}^\circ$ in panels (a), (b), (d), with the last of these angles being replaced by -3° in panel (c). The number of antennas is $N = 4$ in panel (a) and $N = 5$ in panels (b)–(d). The grid for the plots is $\theta = -30 : 0.00010 : 30$ degrees in panels (a)–(c) and $\theta = -30 : 0.00015 : 30$ degrees in panel (d). See main text for a discussion.

When the number of antennas N exceeds the number of sources M by 1, i.e., $N = M + 1$, the M equations (69) remain the same, but one can now set 2 more conditions where $g(\theta_e) = 1$ in addition to (70) (here the $2 = 2(M+1) - 1 - M - (M-1)$). For example, one can choose one of these additional angles to be the midpoint between the left end point of the range of the angles for which the graph of the pseudospectrum $P(\theta)$ is plotted and the left-most source angle; and similarly for the remaining one θ_e at the right end of the range.

When the number of antennas N exceeds M by more than 1, i.e., $N > M + 1$, one has yet $2(N-M)$ additional angles, whose locations appear to be chosen arbitrarily, where one can set $g(\theta_e) = 1$.

Pseudospectra (59) obtained with the matrix X calculated by the above algorithm are shown in Fig. 29. Since at the exact source angles, where condition (69) holds, the logarithm is undefined, we added 10^{-10} to the denominator in (59); this explains the height of the peaks corresponding to the sources being 100 dB. We note a remarkably fine resolution of the sources even when they are quite close, such as in panel (c). Yet, these pseudospectra highlight two problems of the above approach.

1. There are multiple false peaks, not corresponding to the true sources, but where condition (69) is (nearly) met. Two such false peaks are labeled with circles in panel (a) (and not labeled in the remaining panels).

2. The fineness of the resolution is less of a virtue than it is a liability. Namely, if the grid on which the pseudospectrum is plotted is shifted even slightly so that (some of) the source angles no longer "land" on it, the height of the corresponding "true" peaks can be greatly reduced and may become comparable with the height of the false peaks. This is labeled with squares for the two left-most true peaks in panel (d), which is to be compared with panel (b), where the true

peaks land precisely on the grid points.

Yet another problem with the fineness of the peaks is described in the next subsection, where we also present a remedy to both of the aforementioned drawbacks.

Let us now discuss what would happen if we had allowed matrix X to be arbitrary, as in (57), rather than Hermitian. Such an arbitrary matrix is produced by COSnet, as discussed in Sec. 3.7 and earlier. If the source angles corresponding to X_H and to X_{SH} differ even so slightly (which COSnet has no way of preventing), the corresponding function $|g(\theta)|$ in (64) would not be exactly zero. This will "blunt" the peaks of the pseudospectrum.[1] It is this reason which lies behind the observation, made in Sec. 3.7, that matrices X_H and X_{SH} extracted from X_{ML} often produce finer peaks in the pseudospectrum if used separately.

The above observation, in fact, motivates a remedy for the two aforementioned drawbacks of the current method, which (the remedy) will be described in the next subsection.

We conclude this subsection with two side notes.

Side Note 1:

The presence of false peaks in the pseudospectrum agrees with the observation made in Sec. 3.6.2 that X_{ML} (or its Hermitian part, which we have essentially considered in this section) can detect up to $2N-2$ sources. In Sec. 3.6.2 that was concluded based on counting the number of zeros of $g_{ML}(z)$ in the complex plane of variable z defined there. Here, we can arrive at the same conclusion by slightly modifying conditions (69) and (70) as follows. Since we have $2N-1$ effective parameters E_p, $p=1,\ldots,2N-1$ defined as in (68) in matrix X, we can set $2N-2$ conditions of the form (69) and just one *inhomogeneous* condition of the form (70). We need at least one inhomogeneous condition since otherwise all E_p will equal zero. Thus, we can find X for which $g(\theta)$ will vanish at $2N-2$ source angles θ_m, $m=1,\ldots,2N-2$.

The interested reader can slightly modify, as explained in the previous paragraph, the code `construct_Xml.m` and explore the corresponding pseudospectra.

Side Note 2:

The explicit form of matrix X allows one to examine its eigenvalues. For the cases presented in panels (a), (b,d), and (c) of Fig. 29, they are, respectively: $\{-4.0,-0.89,0.17,0.19\}\cdot 10^3$, $\{-0.33,-0.26,-0.13,2.1,8.0\}\cdot 10^3$, and $\{-0.096,-0.075,0.0059,0.73,2.3\}\cdot 10^6$. Note that X can have eigenvalues of both signs, just as X_{ML} in Sec. 3.7. This is in contrast to X_{MUSIC}, which can only have non-negative eigenvalues and hence has fewer "degrees of freedom" than X_{ML} or the X matrix considered in this section.

3.8.2 Improving the X of Sec. 3.8.1

We will first present the idea/algorithm behind this improvement and then will explain why it works.

Instead of considering a single matrix X, as in the previous subsection, consider *multiple* matrices X_q, $q=1,\ldots,Q$ (for some arbitrary Q) satisfying the following properties.

1. Each X_q satisfies condition (69) at all of the angles $\theta'_m(q)$, where

$$\theta_m-\sigma\le\theta'_m(q)\le\theta_m+\sigma,\qquad m=1,\ldots,M, \tag{71}$$

   with the "source jitter" parameter σ being discussed later in the text.

2. Each X_q satisfies condition (70) at angles $\theta_e(q)$ that *differ substantially* with q. For example, one can choose

$$\theta_e(q)=\big(q\,\theta_m+(Q+1-q)\,\theta_{m+1}\big)/(Q+1),\quad m=1,\ldots,M-1 \tag{72}$$

   for the first $(M-1)$ "extra" angles. If $N=M+1$, one can similarly set the other two $\theta_e(q)$ to be

$$\theta_{e,\,\text{left}}(q)=\big(q\,\min(\theta)+(Q+1-q)\,\theta_1\big)/(Q+1),\quad \theta_{e,\,\text{right}}(q)=\big(q\,\theta_M+(Q+1-q)\,\max(\theta)\big)/(Q+1). \tag{73}$$

   If $N>M+1$, one can set the remaining $\theta_e(q)$ to be either at the left or right edge of the scanned range, i.e., to $\min(\theta)$ or $\max(\theta)$.

[1] This has already been mentioned after (64).

Finally, define function $g(\theta) \geq 0$ to be:

$$g(\theta) = \sqrt{g_1(\theta)^2 + \cdots + g_Q(\theta)^2}, \qquad \text{where} \quad g_q(\theta) \equiv a^H(\theta) X_q a(\theta), \quad q = 1, \ldots, Q. \tag{74}$$

By Property 1 above,

$$g(\theta) \approx 0 \qquad \text{for} \quad \theta_m - \sigma \leq \theta \leq \theta_m + \sigma, \qquad m = 1, \ldots, M. \tag{75}$$

However, since individual $\theta'_m(q)$ in (71) are all different (unless $\sigma = 0$), then different g_q vanish at slightly different angles, and the resulting $g(\theta)$ in (75) is small but nonzero. This blunts the peaks of the pseudospectrum. One reason why this may be desirable was illustrated by comparison of panels (d) and (b) of Fig. 29. Another reason will be discussed later in this section.

Furthermore, recall from Fig. 29 that the X computed in Sec. 3.8.1 gave rise to a $g(\theta)$ which had "false zeros" (i.e., false peaks of $1/g(\theta)$) at some angles that did *not* correspond to source angles. Now, *individual* $g_q(\theta)$ in (75) can also have such false zeros, but the probability that the combined $g(\theta)$ would vanish is, practically, nil. Indeed, for $g(\theta)$ to have a zero away from the intervals (71) is practically impossible because it would require that the false zeros of *all* $g_q(\theta)$ would coincide. Thus, Property 2 ensures that the pseudospectrum computed from (75) does not have false peaks.

The pseudospectra based on Eqs. (71)–(74) are shown in Fig. 30. In all panels, $N = 5$ and $M = 4$, as in Fig. 29(b)–(d). All panels of Fig. 30 demonstrate absence of false peaks, in accordance with the explanation above. The fact that all peaks are of different height is not at all related to different powers of the sources and is, rather, a mathematical artifact of how closely $g(\theta)$ based on (74) approaches 0 (see (75)). Panel (b) demonstrates that the resolution provided by (74) is low when the source jitter parameter σ becomes comparable to the inter-source angular separation (see the two right-most peaks). This resolution can be improved by reducing σ, as demonstrated by panel (c). Panel (d), which should be compared to panel (c), demonstrates that the new $1/g(\theta)$ becomes much less sensitive to its (approximate) zeros not landing on the grid points than the $1/g(\theta)$ of Sec. 3.8.1. This is contrasted to the high sensitivity of the latter $1/g(\theta)$ shown by Fig. 29(b,d). Also, we have verified that using $N = 4$ antennas instead of 5 for the same sources does not change the resolution of the close-by peaks 3 and 4, although (surprisingly) it slightly improves the resolution of the already well-resolved peaks 1 and 2.

To conclude this section, we present a proposal of how fast computation of the counterpart of X_{ML} can be used in signal detection. Note that our X was *not* computed based on an actual signal $\vec{x}$, defined in (1).

1. Assume a value for the number of sources M. Below, let us suppose that $M = 5$. Pre-compute a "bank" of $\{X_q\}_{q=1}^Q$'s for all combinations of M source angles θ_m on a relatively coarse grid, say, $[-30 : 0.5 : 30]$ degrees. There will be $121!/(116! \cdot 5!) \approx 120^5/120 \approx 2 \cdot 10^8$ of such $\{X_q\}_{q=1}^Q$'s. Here the '121' is the number of grid points and the '5' is the number of sources. For $M = 4$ sources, the number of $\{X_q\}_{q=1}^Q$'s will be more than an order of magnitude smaller.

   Even the above $O(10^8)$ number of $\{X_q\}_{q=1}^Q$'s is manageable to scan (see next item) in real time. This number would have been much higher, and hence not manageable, if one could expect well-resolved peaks of $1/g(\theta)$ only on a fine grid. This is why Property 1 at the beginning of this subsection was needed to blunt the peaks of $1/g(\theta)$ and thereby make them persist on a coarse grid.

   If the MUSIC algorithm, based on the actual received signal $\vec{x}$, can narrow down the possible range of the locations of the sources, the above number of $\{X_q\}_{q=1}^Q$'s can be significantly reduced.

   Furthermore, if the task is to track specific sources (as opposed to discover new ones), the range of angle values to be scanned (same as the grid above) can (possibly) be reduced further, as for each new incoming sample of $\vec{x}$ the locations of the source angles are expected to be changing gradually.

   In either case, when the scanned range of θ's can be narrowed, one can afford to increase its resolution; in the above example, it would correspond to reducing the 0.5° to, say, 0.25°.

2. For each of the $\{X_q\}_{q=1}^Q$'s, compute

$$g[\vec{x}] \equiv \sqrt{\sum_{q=1}^{Q} \left(\vec{x}^H X_q \vec{x}\right)^2}, \tag{76}$$

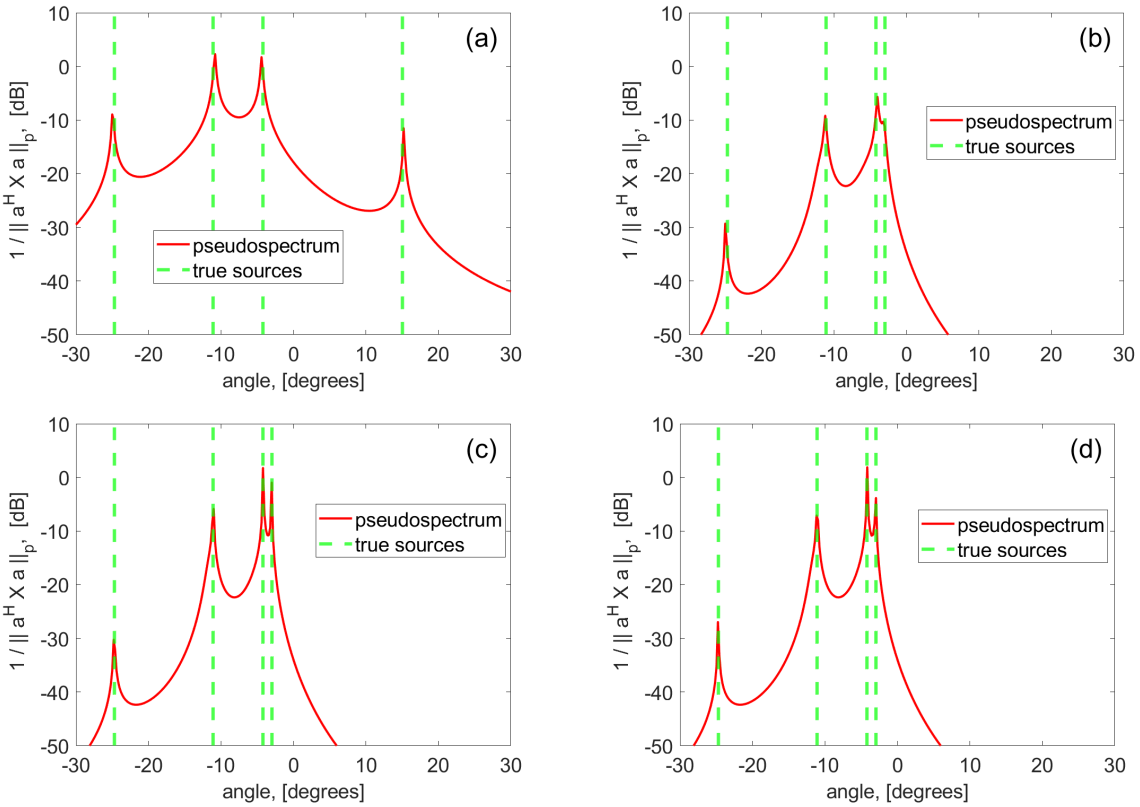

Figure 30: Pseudospectra computed with Eqs. (71)–(74) of Sec. 3.8.2. In all panels, we chose to display results for $Q = 25$, although the resolution of the peaks improves only slightly as Q exceeds 3 (this is not related to M and N in any way). Sources are located at $\theta_m = \{-24.7, -11.05, -4.2, 15\}^\circ$ in panel (a) (as in Fig. 30(b,d)) and at $\theta_m = \{-24.7, -11.05, -4.2, -3\}^\circ$ in panels (b)–(d) (as in Fig. 30(c)). The source jitter parameter $\sigma = 0.25^\circ$ in panels (a,b) and $\sigma = 0.025^\circ$ in panels (c,d). The grid for the plots is $\theta = -30 : 0.2 : 30$ degrees in panels (a)–(c) and that shifted by 0.04° (an arbitrary small number) in panel (d). See main text for a discussion.

similarly to (74). One can expect (*but this assumption must be tested*) that

$$g\big[\vec{x}\big] = 0 \quad \text{or, especially in the presence of noise, minimum} \tag{77}$$

for the $\{X_q\}_{q=1}^{Q}$ corresponding to the actual configuration of sources. This will indicate what this actual set of source angles is.

We emphasize that Step 2 of the above proposal contains an assumption and hence remains to be tested.

3.9 Fourth-Order statistics

3.9.1 On modeling the signal vector $\vec{x}$

Approaches to the Direction of Arrival estimation problem based on fourth-order statistics use fourth-order cumulants, instead of the covariance matrix, of the incoming signal $\vec{x}$ (defined in (1)). Therefore, the first issue to address is how to properly model this signal and, more specifically, the source vector $\vec{s}$. Currently, this is done in the code `signal_gen.m` by assuming that components s_k of $\vec{s}$ are zero-mean, normally or uniformly distributed random variables (using respective Matlab's commands `randn` and `(rand-0.5)`). In the case of the covariance matrix, there should be little difference between these two choices. In the case of cumulants, the difference is dramatic, because they vanish for the normally distributed s_k but do not vanish for uniformly distributed ones. It is not clear, however, that using uniformly distributed s_k is the only remaining, or correct, choice, either. Indeed, there is a continuum of other distributions, all of which would yield different cumulants compared to the choice of the uniformly distributed s_k.

What seems reasonable instead is to not only model s_k by the distribution that arises in actual applications (for example: do s_k really have zero mean values?), but also to take into account how $\vec{x}$ is detected before being processed. For example, and hypothetically, if only the intensity, not phase, of each component of $\vec{x}$ is detected by the antennas, then the transformation of the electromagnetic field into the electric current (which retains the information about the field's intensity only) at the antennas, is needed.

3.9.2 Angle resolution is degraded by cumulants compared to covariances

References [PF91] and [CF99] appear to suggest that using cumulants of $\vec{x}$ is equivalent to using the so-called Kronecker product of $\vec{x}$:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} \otimes \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}^{*} = \begin{pmatrix} x_1 x_1^* \\ x_2 x_1^* \\ \vdots \\ x_N x_1^* \\ x_1 x_2^* \\ \vdots \end{pmatrix} \tag{78}$$

See Appendix in [PF91] and the text before Eq. (3) in [CF99]. Mathematically, replacing signal $\vec{x}$ with $\vec{x} \otimes \vec{x}^*$ is equivalent to replacing the matrix A of steering vectors in Eq. (2) of this report with the Kronecker product

$$A_{Kron} \equiv A \otimes A^*; \tag{79}$$

see Eq. (3.4) in [PF91]. Below we show what implications this replacement has for the angle resolution by an antenna array.

By experimenting with the code `MUSIC.m`, we have found that MUSIC is able to resolve the source angles up to their separation of about 3°. For example, with the default parameters of MUSIC:

```
N=4;                % Number of ULA Array Elements
d=0.5;              % Element Spacing (in terms of wavelength)
K=2e+2;             % Number of Snapshots
az_angs=-30:.1:30; % Span of angles that are assessed
SNR=[25 25]+0;      % Signal to Noise Ratio of Signals, vector dim(1,M)
```

and the source angles

```
theta=[0 3];        % DoA of Sources, vector dim(1,M)
```

the peaks of the pseudospectrum produced by `MUSIC.m` are not consistently resolved from one run of the code to another (i.e., are dependent on the initial state of the random number generator in `signal_gen.m`). When the angular separation is increased from 3° to 6°, the peaks become being resolved consistently (in the above sense).

We look for a clue to this behavior in the eigenvalues of the signal covariance matrix, defined by `R=x*x'/K` in the code. They are shown in Table 1 as the output of the code `MUSICplus.m`, which is a slight modification of `MUSIC.m` that also computes the covariance matrix of the sources, `Rsource = s*s'/K` and its eigenvalues, as well as finds singular values of the matrices of steering vectors A and of A_{Kron}, defined in (79). One notices that the two dominant eigenvalues of `R`, corre-

```
Signal Subspace Implemented = 2, Number of Sources =  2, Number of Elements = 4
Num.Samples = 200, snr = 25, 25,
Source Angles = 0.0, 3.0,
***** ***** *****
Eigenvalues of R (dB) = 27.63,  7.78,  0.22, -0.69,
Eigenvalues of R-SOURCE (dB) = 24.16, 25.51,
Singular values of A (dB) = 1.49, -8.87,
Singular values of kron(A,A*) (dB) = 2.97, -7.38, -7.38, -17.74,
```

Table 1: Eigenvalues and singular values for sources separated by 3°.

sponding to the two sources, differ by about 20 dB (see also Fig. 25 in Sec. 3.7). Recall that

$$R = A^H R_{source} A. \tag{80}$$

Since the eigenvalues of `Rsource` differ by only ~ 1 dB, the rest of the difference, i.e. around 19 dB, must be contributed by the difference in the singular values of $A^H A$. This is qualitatively born out by `MUSICplus.m`'s output displayed in Table 1: the singular values of A differ by ~ 10.5 dB, whence those of $A^H A$ differ by twice as much (in dB).

For the source angle separation of 6°, the corresponding results are shown in Table 2. There one sees that the dominant eigenvalues of `R`, corresponding to the sources, differ by only 14 dB (as opposed to 20 dB in Table 1), and this difference is explained by (twice, in dB) the difference of the singular values of A.

Now, if one uses cumulants instead of covariances in MUSIC, which is tantamount to using A_{Kron} instead of A, all the previous estimates will need to be made with the singular values of A_{Kron}. There is, probably, a theoretical result relating singular values of A_{Kron} (denoted by $\sigma_{A_{Kron}}$) to those of A,

```
Signal Subspace Implemented=  2, Number of Sources =  2, Number of Elements = 4
Num.Samples = 200, snr = 25, 25,
Source Angles = 0.0, 6.0,
***** ***** *****
Eigenvalues of R (dB) = 27.93, 13.43,  0.04, -0.77,
Eigenvalues of R-SOURCE (dB) = 24.82, 25.11,
Singular values of A (dB) = 1.43, -5.90,
Singular values of kron(A,A*) (dB) = 2.86, -4.46, -4.46, -11.79,
```

Table 2: Eigenvalues and singular values for sources separated by 6°.

but it may be more illuminating to illustrate it with numeric examples. From the last two lines of Tables 1 and 2, one sees that

$$\max(\sigma_{A_{Kron}}) - \min(\sigma_{A_{Kron}}) \;\approx\; 2\,(\max(\sigma_A) - \min(\sigma_A)\,) \quad \text{in dB.} \tag{81}$$

In particular, for the source angle separation of 6°,

$$\max(\sigma_{A_{Kron}}) - \min(\sigma_{A_{Kron}}) \approx 14.5 \text{ dB}.$$

This is more (i.e., worse) than $\max(\sigma_A) - \min(\sigma_A)$ for the angular separation of 3°, and therefore one expects that sources that are as far as 6° apart cannot be consistently resolved by the MUSIC algorithm based on the fourth-order statistics.

Below we present an idea of how one can attempt to reduce $\max(\sigma_{A_{Kron}}) - \min(\sigma_{A_{Kron}})$. We emphasize that implementation of this idea will require a substantial amount of work. For example, it can be considered at a future MPI workshop.
Step 1: "Regularize" (a.k.a. well-condition) matrix $A_{Kron}^H A_{Kron}$ in a standard way:

$$\left(A_{Kron}^H A_{Kron}\right)_{Reg} \;\equiv\; A_{Kron}^H A_{Kron} + \tau I, \tag{82}$$

where $\tau > 0$ is a (small) regularization parameter and I is the identity matrix of appropriate dimensions. Choose a value of τ to make the condition number of $\left(A_{Kron}^H A_{Kron}\right)_{Reg}$, i.e.,

$$\max\left(\sigma_{\left(A_{Kron}^H A_{Kron}\right)_{Reg}}\right) - \min\left(\sigma_{\left(A_{Kron}^H A_{Kron}\right)_{Reg}}\right), \tag{83}$$

"acceptable" (which is something to be found by experimentation).
Step 2: Find a regularized $A_{Kron,Reg}$ by using a low-rank decomposition of $\left(A_{Kron}^H A_{Kron}\right)_{Reg}$:

$$\left(A_{Kron}^H A_{Kron}\right)_{Reg} \;\approx\; A_{Kron,Reg}^H A_{Kron,Reg}. \tag{84}$$

While low-rank decomposition must be known to experts in Linear Algebra, the challenge in (84) is to keep the phases of the entries of $A_{Kron,Reg}$ (i.e., the virtual antennas) minimally distorted compared to the phases of A_{Kron}.

3.10 Alternative Methods for Subspace Identification

3.10.1 Paths to Subspace Identification in Physical Space

The ultimate goal is to identify the signal subspace $\text{span}(\mathbf{V}_S)$ (or its complement, the noise subspace $\text{span}(\mathbf{V}_N)$) from the data. When the number of sources M is less than the number of sensors N, a noise subspace with non-zero dimension exists in the physical $\mathbb{C}^N$ space. The following methods operate within this physical space.

3.10.2 The Core Principle: A Chain of Equivalences

All subspace-based algorithms working in the physical space are built upon a chain of logical equivalences that connects the physical reality of a signal's direction to a computable mathematical condition:

$$\mathbf{a}(\theta) \in \text{span}(\mathbf{A}) \quad \Leftrightarrow \quad \mathbf{a}(\theta) \in \text{span}(\mathbf{V}_S) \quad \Leftrightarrow \quad \mathbf{a}(\theta) \perp \text{span}(\mathbf{V}_N)$$

The first equivalence holds because the steering vectors for distinct angles are linearly independent. The second equivalence is the fundamental theorem of subspace methods. The final equivalence follows from the orthogonality of the signal and noise subspaces.

3.10.3 Path 1: Eigendecomposition of the Covariance Matrix (MUSIC)

The MUSIC algorithm is the most direct implementation of this principle. It involves constructing the sample covariance matrix $\hat{\mathbf{R}}$ (a second-order statistic) from the data and then finding its eigenspace

through analytical decomposition to obtain a basis for the noise subspace, $\hat{\mathbf{V}}_N$. The DOA is then found by searching for peaks in the pseudospectrum:

$$P_{MUSIC}(\theta) = \frac{1}{\mathbf{a}(\theta)^H \hat{\mathbf{V}}_N \hat{\mathbf{V}}_N^H \mathbf{a}(\theta)} = \frac{1}{||\hat{\mathbf{V}}_N^H \mathbf{a}(\theta)||^2}$$

This path is fundamentally limited to cases where $M < N$, as otherwise the physical noise subspace does not exist.

3.10.4 Path 2: Learning the Noise Subspace Directly (AI-based)

An alternative path in the physical space is to use a Neural Network (NN) to learn the noise subspace directly from the data, bypassing the explicit calculation and decomposition of the covariance matrix.

The mapping can be represented as:

$$\text{NN}(\mathbf{x}(t_1), \ldots, \mathbf{x}(t_K)) \to \mathbf{V}_{N,ML}$$

Here, the NN takes raw snapshots as input and its learning objective is to output an orthonormal basis $\mathbf{V}_{N,ML}$ for the physical noise subspace. This learned basis can then be used in the standard MUSIC pseudospectrum formula.

The advantage of this approach is its robustness. By training on vast amounts of data under various conditions (low SNR, few snapshots), the NN can learn a more robust mapping from imperfect data to the true noise subspace than the simple sample-and-decompose procedure of MUSIC. In essence, this NN acts as a powerful "denoiser" or "corrector" for the subspace estimation process, potentially yielding a more accurate covariance matrix estimate implicitly.

However, this architecture has a critical weakness. Since the NN's learning target is still the noise subspace $\mathbf{V}_N$ within the physical $\mathbb{C}^N$ space, and it does not alter the steering vector $\mathbf{a}(\theta)$, it is fundamentally just a different, more powerful path to the same objective as MUSIC. Therefore, it is still bound by the same mathematical constraints. **When the number of sources is equal to or greater than the number of sensors ($M \geq N$), the physical noise subspace dimension is zero or less. The NN's target ceases to exist, and this method, like MUSIC, will fail.** Its improvement over MUSIC is primarily in estimation accuracy and robustness under challenging but "well-posed" ($M < N$) conditions, not in overcoming this fundamental limitation.

3.10.5 Paths to Subspace Identification in Virtual Space

To overcome the fundamental limitation of $M < N$, we must move beyond the physical space. The core idea is to find a non-linear mapping, Φ, that "lifts" the problem from the N-dimensional physical space into a much higher N_{virt}-dimensional "virtual" or "feature" space.

$$\Phi : \mathbf{a} \in \mathbb{C}^N \to \mathbf{a}' = \Phi(\mathbf{a}) \in \mathbb{C}^{N_{virt}}$$

The key is to find this mapping Φ such that a consistent transformation for the data, $\mathbf{x} \to \mathbf{x}'$, also exists, resulting in a new linear model in the virtual space:

$$\mathbf{x}' = \mathbf{A}'\mathbf{s}' + \mathbf{n}'$$

Once this new linear model is established in the N_{virt}-dimensional space, where hopefully $M < N_{virt}$, we can simply re-apply the entire MUSIC algorithm pipeline: construct a new covariance matrix, find its high-dimensional noise subspace $\mathbf{V}'_N$, and test for orthogonality with the virtual steering vectors $\mathbf{a}'(\theta)$.

3.10.6 Path 1: Explicit Transformation via Higher-Order Statistics (HOS/C4)

This is the classic, physically-motivated approach to creating a virtual space. It explicitly defines the mapping Φ and the corresponding data transformation. Assuming the sources are uncorrelated, the source covariance matrix $\mathbf{R}_s$ is a diagonal matrix, $\mathbf{R}_s = \text{diag}(P_1, \ldots, P_M)$, where $P_m = E[|s_m(t)|^2]$ is the power of the m-th source. The transformation is derived by vectorizing the covariance matrix. Let's define our new data vector $\mathbf{y}$ as:

$$\mathbf{y} = \text{vec}(\mathbf{R}_x)$$

where $\mathbf{R}_x = \sum_{m=1}^{M} P_m \mathbf{a}_m \mathbf{a}_m^H + \sigma^2 \mathbf{I}$. Using the linear property of the 'vec' operator and the Kronecker product identity $\text{vec}(\mathbf{u}\mathbf{v}^H) = \mathbf{v}^* \otimes \mathbf{u}$, we derive:

$$\begin{aligned}\mathbf{y} &= \text{vec}\left(\sum_{m=1}^{M} P_m \mathbf{a}_m \mathbf{a}_m^H + \sigma^2 \mathbf{I}\right) \\ &= \sum_{m=1}^{M} P_m \cdot \text{vec}(\mathbf{a}_m \mathbf{a}_m^H) + \sigma^2 \cdot \text{vec}(\mathbf{I}) \\ &= \sum_{m=1}^{M} P_m (\mathbf{a}_m^* \otimes \mathbf{a}_m) + \sigma^2 \text{vec}(\mathbf{I})\end{aligned}$$

By defining the virtual steering matrix $\mathbf{A}' = [\mathbf{a}_1^* \otimes \mathbf{a}_1, \ldots, \mathbf{a}_M^* \otimes \mathbf{a}_M]$ and the source power vector $\mathbf{p} = [P_1, \ldots, P_M]^T$, we obtain a new linear model in an N^2-dimensional space:

$$\mathbf{y} = \mathbf{A}'\mathbf{p} + \sigma^2 \text{vec}(\mathbf{I})$$

This proves that the mapping $\Phi(\mathbf{a}) = \mathbf{a} \otimes \mathbf{a}^*$ consistently induces a new linear model. We can now apply subspace methods to this model, which is valid as long as $M < N^2$.

3.10.7 Path 2: Implicit Transformation via Kernel Methods

The HOS/C4 method is powerful but represents just one specific, "heuristic" way of expanding dimensionality. A more general and often more powerful approach comes from machine learning, known as the Kernel Method. This method is superior for several reasons:

1. **Generality**: It provides a framework for creating a vast family of transformations Φ, not just the fixed Kronecker product. Any Φ is valid as long as it satisfies key properties (non-linearity, dimensionality expansion, injectivity, and inducement of a linear model). This allows us to choose transformations better suited to the problem.

2. **Bypassing $\mathbf{V}_N$ Construction**: We observed that the noise subspace $\mathbf{V}_N$ is almost exclusively used via its inner product with a test vector $\mathbf{a}(\theta)$ (in the term $||\mathbf{V}_N^H \mathbf{a}(\theta)||^2$). The kernel method cleverly recognizes that if we can compute this final "orthogonality score" directly, we don't need to find $\mathbf{V}_N$ as an intermediate step.

3. **Bypassing Φ Construction**: Most remarkably, a non-linear kernel function $\mathcal{K}$ implicitly defines a mapping Φ to a very **high-dimensional** (or even infinite-dimensional) feature space. The existence of this Φ is not assumed, but mathematically guaranteed by Mercer's Theorem for any valid kernel. This allows us to reap the benefits of the high-dimensional space without ever explicitly constructing it.

The power of the kernel trick is that it allows us to bypass the explicit construction of the high-dimensional subspace basis $\mathbf{V}_\Phi$ and directly compute the required geometric quantity—the inner product between a mapped test vector $\Phi(\mathbf{a}(\theta))$ and a feature-space basis vector $\mathbf{V}_{\Phi,k}$—using only the kernel function. This key relationship is:

$$\langle \Phi(\mathbf{a}(\theta)), \mathbf{V}_{\Phi,k} \rangle = \frac{1}{\sqrt{\lambda_k}} \sum_{i=1}^{K} (\boldsymbol{\alpha}_k)_i \mathcal{K}(\mathbf{a}(\theta), \mathbf{x}_i)$$

where:

- $\mathcal{K}(\cdot, \cdot)$ is the chosen kernel function (e.g., Polynomial or Gaussian RBF).
- $\mathbf{a}(\theta)$ is the physical steering vector for a test angle θ.
- $\{\mathbf{x}_i\}_{i=1}^K$ are the K received signal snapshots.
- λ_k and $\boldsymbol{\alpha}_k \in \mathbb{C}^K$ are the k-th eigenvalue and corresponding eigenvector obtained from the eigendecomposition of the $K \times K$ Gram matrix $\mathbf{G}$, whose elements are defined as $\mathbf{G}_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$.

3.10.8 Algorithm Workflow

The practical implementation of Kernel MUSIC or Kernel PCA for DOA estimation follows these steps:

1. **Choose a Kernel Function**: Select a kernel $\mathcal{K}(\mathbf{u}, \mathbf{v})$ that is appropriate for the problem.
2. **Form the Gram Matrix**: Using K snapshots of the received signal $\{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$, construct the $K \times K$ Gram matrix $\mathbf{G}$, where its elements are given by:

$$\mathbf{G}_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$$

3. **Decompose G to Find Coefficients**: Solve the eigenvalue problem for the Gram matrix, $\mathbf{G}\boldsymbol{\alpha}_k = \lambda_k \boldsymbol{\alpha}_k$. Keep the M largest eigenvalues $\{\lambda_k\}_{k=1}^{M}$ and their corresponding eigenvectors $\{\boldsymbol{\alpha}_k\}_{k=1}^{M}$. These are the coefficients that define the signal subspace in the feature space.
4. **Calculate the Pseudospectrum**: For each test angle θ, calculate its projection onto the feature space signal subspace. The pseudospectrum is the sum of the squared magnitudes of the projections onto each basis vector:

$$P_{\mathcal{K}}(\theta) = \sum_{k=1}^{M} |\langle \Phi(\mathbf{a}(\theta)), \mathbf{V}_{\Phi,k} \rangle|^2 = \sum_{k=1}^{M} \frac{1}{\lambda_k} \left| \sum_{i=1}^{K} (\boldsymbol{\alpha}_k)_i \mathcal{K}(\mathbf{a}(\theta), \mathbf{x}_i) \right|^2$$

   We search for the peaks of $P_{\mathcal{K}}(\theta)$ over θ to find the DOAs. This final formula remarkably contains no reference to Φ or $\mathbf{V}_\Phi$, yet perfectly performs the analysis in the high-dimensional feature space.

3.10.9 A Concluding Outlook: Learning the Optimal Kernel via Constrained Neural Networks

This entire discussion leads to a fascinating and ultimate prospect: if the choice of a kernel function dictates the structure of the virtual space and the algorithm's performance, then the core problem of DOA estimation could be redefined as the search for an optimal, data-driven kernel. A natural, forward-looking question then arises: can this search process itself be accomplished by a Neural Network?

Allowing an NN complete freedom to generate any arbitrary kernel function, however, presents significant hurdles. The training process would be a complex nested optimization, and more importantly, the resulting kernel would be a complete "black box," making it difficult to understand its physical meaning or guarantee its mathematical validity.

Therefore, a more promising and interpretable path is to constrain the neural network's task. Instead of learning a kernel from scratch, the NN could be designed as a sophisticated "architect." In this paradigm, we would pre-define a library of simple, well-understood kernels (e.g., various polynomial, Gaussian, or linear kernels), each acting as a fundamental "building block." The neural network's role would then be to learn the optimal way to **compose and combine** these known kernels—for instance, by finding the best weighted sum or product of them ($\mathcal{K}_{optimal} = \sum w_i \mathcal{K}_i + \ldots$)—and to determine the ideal coefficients for the given data.

3.11 Formal Description of Solving DoA by Machine Learning

It is desirable to have a good understanding of why the ML (Neural Network in particular) approach works well in cases where the MUSIC Algorithm is not working. Based on the investigation of the problem and the some familiarity with the Neural Network approach, it can be a good idea to illustrate a simple, end-to-end example of a Neural Network (NN) performing optimization to obtain Direction of Arrival estimates.

1. Problem Setup

Consider the following scenario:

- **Array:** A Uniform Linear Array (ULA) with $N = 2$ sensors.

- **Sources:** $M = 1$.
- **Noise:** Additive White Gaussian Noise (AWGN).
- **Goal:** Estimate the single DOA, $\theta \in [-90°, 90°]$.

2. Signal Model

The received signal vector $\mathbf{x}(t)$ at time t for N sensors and M sources is generally given by:

$$\mathbf{x}(t) = \mathbf{A}(\boldsymbol{\theta})\mathbf{s}(t) + \mathbf{n}(t)$$

For our specific case ($N = 2$, $M = 1$):

$$\mathbf{x}(t) = \mathbf{a}(\theta)s(t) + \mathbf{n}(t)$$

where:

- $\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \in \mathbb{C}^{2\times 1}$ is the received signal vector.
- $s(t) \in \mathbb{C}$ is the complex amplitude of the single source.
- $\mathbf{n}(t) = \begin{bmatrix} n_1(t) \\ n_2(t) \end{bmatrix} \in \mathbb{C}^{2\times 1}$ is the noise vector.
- $\mathbf{a}(\theta) = \begin{bmatrix} 1 \\ e^{-j2\pi(d/\lambda)\sin\theta} \end{bmatrix} \in \mathbb{C}^{2\times 1}$ is the steering vector for a ULA, where d is the inter-element spacing and λ is the wavelength. We commonly set $d/\lambda = 0.5$.

3. Input Feature for the Neural Network: Covariance Matrix

For DOA estimation, the Spatial Covariance Matrix (SCM) is a standard input feature.

The covariance matrix $\mathbf{R}_x = E[\mathbf{x}(t)\mathbf{x}^H(t)]$ for $M = 1$ source and white noise is:

$$\mathbf{R}_x = \sigma_s^2 \mathbf{a}(\theta)\mathbf{a}^H(\theta) + \sigma_n^2 \mathbf{I}$$

where σ_s^2 is the source power and σ_n^2 is the noise power, and $\mathbf{I}$ is the identity matrix. For simplicity, $\mathbf{R}_x$ can be calculated as follows:

$$\hat{\mathbf{R}}_x = \frac{1}{K}\sum_{n=1}^{K} \mathbf{x}(t_n)\mathbf{x}^H(t_n)$$

For $N = 2$, $\hat{\mathbf{R}}_x$ is a 2×2 complex Hermitian matrix:

$$\hat{\mathbf{R}}_x = \begin{pmatrix} \hat{r}_{11} & \hat{r}_{12} \\ \hat{r}_{12}^* & \hat{r}_{22} \end{pmatrix}$$

Due to the Hermitian property ($\hat{r}_{ii}$ are real, and $\hat{r}_{12}^*$ determines $\hat{r}_{21}$), we only have four independent components. Let $\hat{r}_{12} = a_{12} + jb_{12}$. We flatten these independent components into our input vector $\mathbf{X}_{\text{input}}$:

$$\mathbf{X}_{\text{input}} = \begin{bmatrix} \hat{r}_{11} \\ \hat{r}_{22} \\ a_{12} \\ b_{12} \end{bmatrix} \in \mathbb{R}^4$$

Thus, our input dimension for this NN is $D_{\text{in}} = 4$.

4. Neural Network Architecture

For simplicity, we assume with one hidden layer of NN for this example.

- **Input Layer:** $D_{\text{in}} = 4$ neurons (for $\mathbf{X}_{\text{input}}$).
- **Hidden Layer:** $D_{h1} = H$ neurons (e.g., $H = 10$). We will use the Rectified Linear Unit (ReLU) as the activation function.
- **Output Layer:** $D_{\text{out}} = 1$ neuron (for the estimated angle $\hat{\theta}$). This layer will use a linear activation.
- $\mathbf{W}^{(1)} \in \mathbb{R}^{H \times D_{\text{in}}}$: Weight matrix for the input-to-hidden layer.
- $\mathbf{b}^{(1)} \in \mathbb{R}^{H \times 1}$: Bias vector for the hidden layer.
- $\mathbf{W}^{(2)} \in \mathbb{R}^{1 \times H}$: Weight matrix for the hidden-to-output layer.
- $b^{(2)} \in \mathbb{R}$: Bias scalar for the output layer.
- $f(\cdot)$: ReLU activation function, $f(z) = \max(0, z)$.

5. Forward Propagation

Given an input $\mathbf{X}_{\text{input}}$:

1. **Calculate weighted sum for hidden layer:**

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{X}_{\text{input}} + \mathbf{b}^{(1)}$$

   where $\mathbf{z}^{(1)} \in \mathbb{R}^{H \times 1}$ is the pre-activation output of the hidden layer.

2. **Apply activation function to hidden layer:**

$$\mathbf{h}^{(1)} = f(\mathbf{z}^{(1)}) = \max(0, \mathbf{z}^{(1)})$$

   where $\mathbf{h}^{(1)} \in \mathbb{R}^{H \times 1}$ represents the input in the higher-dimensional feature space learned by the hidden layer.

3. **Calculate weighted sum for output layer:**

$$z^{(2)} = \mathbf{W}^{(2)}\mathbf{h}^{(1)} + b^{(2)}$$

   where $z^{(2)} \in \mathbb{R}$ is the pre-activation output of the output layer.

4. **Final output (estimated DOA):**

$$\hat{\theta} = z^{(2)}.$$

6. Loss Function

We use the Mean Squared Error (MSE) to quantify the difference between the network's prediction $\hat{\theta}$ and the true DOA θ_{true} for a single training example:

$$L(\hat{\theta}, \theta_{\text{true}}) = \frac{1}{2}(\hat{\theta} - \theta_{\text{true}})^2$$

For a batch of N_{batch} examples, the loss is averaged over the batch:

$$L_{\text{batch}} = \frac{1}{N_{\text{batch}}} \sum_{n=1}^{N_{\text{batch}}} \frac{1}{2}(\hat{\theta}_n - \theta_{\text{true},n})^2$$

The primary goal of the optimization process is to minimize this loss function. Note that a different loss function can be used here. For example a loss function designed for a classification Algorithm can be used. Nevertheless, the objective of all such loss function remains the same. That is, minimizing the difference between $\hat{\theta}$ and θ_{true}.

7. Backward Propagation (Backpropagation) and Optimization (Adam)

Backpropagation is the algorithm used to compute the gradients of the loss function with respect to each weight and bias in the network. These gradients indicate the direction and magnitude of change needed for each parameter to reduce the loss. The Adam optimizer then uses these gradients to update the parameters iteratively.

A. Gradients of the Loss Function (Backpropagation)

We need to compute $\frac{\partial L}{\partial \mathbf{W}^{(2)}}, \frac{\partial L}{\partial b^{(2)}}, \frac{\partial L}{\partial \mathbf{W}^{(1)}}, \frac{\partial L}{\partial \mathbf{b}^{(1)}}$.

1. **Output Layer Gradients:**

   - **Loss w.r.t. $\hat{\theta}$ (network output):**

   $$\frac{\partial L}{\partial \hat{\theta}} = (\hat{\theta} - \theta_{\text{true}})$$

   - **Loss w.r.t. $z^{(2)}$ (pre-activation output of output layer):** Since $\hat{\theta} = z^{(2)}$ (linear activation, so $\frac{\partial \hat{\theta}}{\partial z^{(2)}} = 1$):

   $$\frac{\partial L}{\partial z^{(2)}} = \frac{\partial L}{\partial \hat{\theta}} \frac{\partial \hat{\theta}}{\partial z^{(2)}} = (\hat{\theta} - \theta_{\text{true}}) \cdot 1 = (\hat{\theta} - \theta_{\text{true}})$$

   - **Loss w.r.t. $\mathbf{W}^{(2)}$ (weights of output layer):** Recall $z^{(2)} = \sum_{j=1}^{H} W_{1j}^{(2)} h_j^{(1)} + b^{(2)}$. The partial derivative with respect to an element $W_{1j}^{(2)}$ is:

   $$\frac{\partial L}{\partial W_{1j}^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial W_{1j}^{(2)}} = (\hat{\theta} - \theta_{\text{true}}) h_j^{(1)}$$

   In vector/matrix form:

   $$\frac{\partial L}{\partial \mathbf{W}^{(2)}} = (\hat{\theta} - \theta_{\text{true}})(\mathbf{h}^{(1)})^T$$

   - **Loss w.r.t. $b^{(2)}$ (bias of output layer):**

   $$\frac{\partial L}{\partial b^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial b^{(2)}} = (\hat{\theta} - \theta_{\text{true}}) \cdot 1 = (\hat{\theta} - \theta_{\text{true}})$$

2. **Hidden Layer Gradients:** First, we need the gradient of the loss w.r.t. the hidden layer's activations $\mathbf{h}^{(1)}$:

   $$\frac{\partial L}{\partial h_j^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial h_j^{(1)}} = (\hat{\theta} - \theta_{\text{true}}) W_{1j}^{(2)}$$

   In vector form, the error signal propagating back to the hidden layer's activations is:

   $$\frac{\partial L}{\partial \mathbf{h}^{(1)}} = (\mathbf{W}^{(2)})^T (\hat{\theta} - \theta_{\text{true}})$$

   Next, we find the gradient w.r.t. the pre-activation $z_j^{(1)}$ for each hidden neuron:

   $$\frac{\partial L}{\partial z_j^{(1)}} = \frac{\partial L}{\partial h_j^{(1)}} \frac{\partial h_j^{(1)}}{\partial z_j^{(1)}} = \left((\hat{\theta} - \theta_{\text{true}}) W_{1j}^{(2)} \right) \cdot f'(z_j^{(1)})$$

   For the ReLU activation function, the derivative $f'(z_j^{(1)})$ is:

   $$f'(z_j^{(1)}) = \begin{cases} 1 & \text{if } z_j^{(1)} > 0 \\ 0 & \text{if } z_j^{(1)} \leq 0 \end{cases}$$

   In vector form, let $\boldsymbol{\delta}^{(1)}$ be the error signal for the hidden layer's pre-activations:

   $$\boldsymbol{\delta}^{(1)} = \left((\mathbf{W}^{(2)})^T (\hat{\theta} - \theta_{\text{true}}) \right) \odot f'(\mathbf{z}^{(1)}),$$

   where $\odot$ denotes the element-wise (Hadamard) product.

- **Loss w.r.t. $\mathbf{W}^{(1)}$ (weights of hidden layer):**

$$\frac{\partial L}{\partial W_{jk}^{(1)}} = \frac{\partial L}{\partial z_j^{(1)}} \frac{\partial z_j^{(1)}}{\partial X_{\text{input},k}} = \delta_j^{(1)} X_{\text{input},k}.$$

  In matrix form:

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \boldsymbol{\delta}^{(1)} (\mathbf{X}_{\text{input}})^T.$$

- **Loss w.r.t. $\mathbf{b}^{(1)}$ (biases of hidden layer):**

$$\frac{\partial L}{\partial \mathbf{b}^{(1)}} = \boldsymbol{\delta}^{(1)}.$$

B. Adam Optimization Update Rules

The Adam (Adaptive Moment Estimation) optimizer uses these computed gradients to update the network's parameters iteratively. For each parameter $P \in \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, b^{(2)}\}$, and its corresponding gradient $g_t = \frac{\partial L}{\partial P}$ at time step t (after processing a mini-batch):

1. **Initialization (for each parameter):**
   - First moment vector: $m_0 = \mathbf{0}$
   - Second moment vector: $v_0 = \mathbf{0}$
2. **Hyperparameters:**
   - Learning rate: α (e.g., 0.001)
   - Exponential decay rates for moment estimates: β_1 (e.g., 0.9), β_2 (e.g., 0.999)
   - Small constant for numerical stability: ϵ (e.g., 10^{-8})
3. **Update Rule (for parameter P at iteration t):**
   - Compute gradient $g_t = \frac{\partial L}{\partial P}$ (averaged over the current mini-batch).
   - Update biased first moment estimate:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

   - Update biased second moment estimate:

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \quad \text{(element-wise square)}$$

   - Compute bias-corrected first moment estimate:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

   - Compute bias-corrected second moment estimate:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

   - Update parameter:

$$P_{t+1} = P_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

This iterative process, involving forward pass, loss calculation, backward pass for gradients, and parameter updates via Adam, is repeated for many epochs until the network's predictions are sufficiently accurate (i.e., the loss function converges).

8. End-to-End Process Flow

1. **Data Generation (Offline Phase):**
   - For each training example:
     - Randomly choose a true DOA θ_{true} (e.g., uniformly from $[-90°, 90°]$).
     - Simulate a source signal $s(t)$ (e.g., complex Gaussian noise) and $\mathbf{n}(t)$.
     - Generate K received signal snapshots $\mathbf{x}(t_n)$.
     - Compute the sample covariance matrix $\hat{\mathbf{R}}_x = \frac{1}{N_{snapshots}} \sum \mathbf{x}(t_n)\mathbf{x}^H(t_n)$.
     - Extract the input vector $\mathbf{X}_{\text{input}} = \begin{bmatrix} \hat{r}_{11} & \hat{r}_{22} & \text{Re}(\hat{r}_{12}) & \text{Im}(\hat{r}_{12}) \end{bmatrix}^T$.
   - Store these $(\mathbf{X}_{\text{input}}, \theta_{\text{true}})$ pairs as your training dataset.
2. **Network Initialization (Offline Phase):**
   - Initialize all network weights $(\mathbf{W}^{(1)}, \mathbf{W}^{(2)})$ and biases $(\mathbf{b}^{(1)}, b^{(2)})$ with small random values.
   - Initialize Adam's first and second moment vectors (m and v) to zero for all parameters.
3. **Training Loop (Offline Phase):**
   - For a predefined number of *epochs*:
     - Divide the dataset into smaller *mini-batches*.
     - For each mini-batch:
       - **Forward Pass:** For each example in the mini-batch, compute the predicted DOA $\hat{\theta}$ using the current network parameters (as described in Section 5).
       - **Loss Calculation:** Compute the average batch loss L_{batch}.
       - **Backward Pass:** Calculate the gradients $\frac{\partial L_{\text{batch}}}{\partial P}$ for all network parameters P using backpropagation.
       - **Parameter Update:** Use the Adam optimizer to update all weights and biases based on the calculated gradients.
   - Check the loss function (e.g., on a separate validation set) for convergence and prevent overfitting.
4. **Inference (Online/Deployment Phase):**
   - Obtain new, unseen received array data (e.g., from an actual antenna array).
   - Compute the sample covariance matrix $\hat{\mathbf{R}}_x$ from this data.
   - Form the $\mathbf{X}_{\text{input}}$ vector from $\hat{\mathbf{R}}_x$.
   - Perform a single **forward pass** through the *trained* neural network using this $\mathbf{X}_{\text{input}}$.
   - The network's output $\hat{\theta}$ is the estimated DOA for the new signal.

Comments and Observations:

- The process described above shows that through iterative gradient-based optimization, the NN learns the complex nonlinear mapping from the input covariance matrix to the estimated DOA.
- Referring to the matrix X_{ML}, which is analogous to the MUSIC matrix X_{MUSIC}, it can be constructed from the updated parameters set $\boldsymbol{P} := \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, b^{(2)}\}$ of the NN.
- In fact, the NN can be also be designed such that X_{ML} is Hermitian, which makes it easier for comparison with X_{MUSIC}. That is, by having $a^H(\theta)X_{MUSIC}a(\theta) = 0$ and $a^H(\theta)X_{ML}a(\theta) = 0$, we get $a^H(\theta)(X_{MUSIC} - X_{ML})a(\theta) = 0$ if X_{ML} is Hermitian, but is not generally true if X_{ML} is not Hermitian.

- The MUSIC Algorithm starts by describing the system as

$$x_k = As_k + n_k,$$

with,

$$\begin{aligned} A &= [a(\theta_1)\ a(\theta_2)\ \dots\ a(\theta_M)], \\ a(\theta) &= (1, e^{j2\pi d \sin\theta}, e^{j2\pi 2d \sin\theta}, e^{j2\pi 3d \sin\theta}, \dots, e^{j2\pi(N-1)d \sin\theta})^T. \end{aligned}$$

That is, a system that is affine in the signals s_k. It is possible that the Lie-bracket iterations of the terms $a(\theta)$ can provide the "missing" information about the signals. This resembles the topic of controllability via Lie-bracket iterations of the control fields in the Geometric control theory.

- As described, the NN approach is able to obtain the missing information by optimization in a higher-dimensional space. This indicates that using virtual sensors, which effectively increasing N, can operate in a similar manner. Hence, making virtual sensors a very interesting topic to explore.

4 Conclusions

In this report, we have studied the Direction of Arrival problem as solved by the MUSIC algorithm and COSnet. Our approaches were varied. We compared the two methods when sources had different SNRs and when they had similar incident angles. We found that MUSIC performed better when SNR difference was low, but COSnet performed better when SNR difference was high. COSnet was better at distinguishing correlated signals. This was shown by comparing predicted angles as well as analyzing the spectrum of the key matrices from both methods, X_{ML} and X_{MUSIC}.

We also provided several explanations for why COSnet is able to infer a greater number of sources than MUSIC. Our analyses show that the COSnet matrix X_{ML} has more degrees of freedom which can be used to infer more sources. Separately, we investigated some numerical methods for source detection based on least-squares: Beurling-LASSO, regular LASSO, and an unregularized sparse-recovery method.

References

[CF99] Pascal Chevalier and Anne Férréol. On the virtual array concept for the fourth-order direction finding problem. *IEEE Trans. Signal Proc.*, 47(9):2592–2595, 1999.

[DDPS18] Quentin Denoyelle, Vincent Duval, Gabriel Peyré, and Emmanuel Soubies. The sliding frank-wolfe algorithm and its application to super-resolution microscopy, 2018.

[DTVB09] Lars KS Daldorff, Deepak S Turaga, Olivier Verscheure, and Alain Biem. Direction of arrival estimation using single tripole radio antenna. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2149–2152. IEEE, 2009.

[GP73] Gene H Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on numerical analysis*, 10(2):413–432, 1973.

[GRY+07] Marshall Grice, Jeff Rodenkirch, Anatoly Yakovlev, HK Hwang, Zekeriya Aliyazicioglu, and Anne Lee. Direction of arrival estimation using advanced signal processing. In *2007 3rd international conference on recent advances in space technologies*, pages 515–522. IEEE, 2007.

[HS02] Yingbo Hua and Tapan K Sarkar. Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(5):814–824, 2002.

[NP07] M Thamban Nair and Sergei V Pereverzev. Regularized collocation method for fredholm integral equations of the first kind. *Journal of COMPLEXITY*, 23(4-6):454–467, 2007.

[PF91] Boaz Porat and Benjamin Friedlander. Direction finding algorithms based on higher-order statistics. *IEEE Trans. Signal Proc.*, 39(9):2016–2024, 1991.

[RK89] Richard Roy and Thomas Kailath. Esprit-estimation of signal parameters via rotational invariance techniques. *IEEE Transactions on acoustics, speech, and signal processing*, 37(7):984–995, 1989.

[Sch82] Ralph Otto Schmidt. *A signal subspace approach to multiple emitter location and spectral estimation*. Stanford University, 1982.

[SK22] Sören Schulze and Emily J. King. Formulating beurling lasso for source separation via proximal gradient iteration, 2022.

[SP95] Tapan K Sarkar and Odilon Pereira. Using the matrix pencil method to estimate the parameters of a sum of complex exponentials. *IEEE Antennas and propagation Magazine*, 37(1):48–55, 1995.

[SS02] Petre Stoica and Kenneth C Sharman. Maximum likelihood methods for direction-of-arrival estimation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(7):1132–1143, 2002.

[YZ19] Di Yuan and Xinming Zhang. An overview of numerical methods for the first kind fredholm integral equation. *SN Applied Sciences*, 1:1–12, 2019.