OFDM Channel Estimation

By Daniel Paz

Table of Contents

OFDM Signal

The ODFM signal is an IFFT over the data symbols in frequency domain

$$s(t)=\sum_{k=0}^{N-1} d_k e^{j\frac{2\pi}{T}kt} 1_{t\in[-T_g,T]}$$

The received signal is:

$$y(t)=\sum_{p=0}^{P-1} h_p s(t-\tau_p)+n(t)=\sum_{p=0}^{P-1} h_p \sum_{k=0}^{N-1} d_k e^{j\frac{2\pi}{T}k(t-\tau_p)} 1_{t\in[\tau_p-T_g,\tau_p+T]}+n(t)=$$
$$\sum_{k=0}^{N-1} d_k e^{j\frac{2\pi}{T}kt} \sum_{p=0}^{P-1} h_p e^{-j\frac{2\pi}{T}k\tau_p} 1_{t\in[\tau_p-T_g,\tau_p+T]}+n(t)$$

If we activate an FFT back to the frequency domain:

(1): No ISI: $\tau_p < T_g$

$$\hat{H}_k=\frac{1}{T}\int_0^T y(t)\cdot e^{-j\frac{2\pi}{T}kt}dt=$$
$$\frac{1}{T}\int_0^T\left(\sum_{k'=0}^{N-1} d_k e^{j\frac{2\pi}{T}k't}\sum_{p=0}^{P-1} h_p e^{-j2\pi f(k')\tau_p} 1_{t\in[\tau_p-T_g,\tau_p+T]}+n(t)\right)\cdot e^{-j\frac{2\pi}{T}kt}dt=$$
$$\sum_{k'=0}^{N-1} d_k\cdot\overbrace{\sum_{p=0}^{P-1} h_p e^{-j2\pi f(k')\tau_p}}^{H(k')}\cdot\overbrace{\frac{1}{T}\int_0^T e^{-j\frac{2\pi}{T}(k-k')t}dt}^{g_p[k-k']}+\overbrace{\frac{1}{T}\int_0^T n(t)\cdot e^{-j\frac{2\pi}{T}kt}dt}^{\tilde{n}(t)}=$$

$$\boxed{\hat{H}_k=\sum_{k'=0}^{N-1} d_{k'}\cdot H[k']\cdot g_p[k-k']+\tilde{n}[k]}$$

$$g_p(k-k')=\delta[k-k']$$

$$\boxed{y_k=\hat{H}_k=d_k\cdot H[k]+\tilde{n}[k]}\overbrace{=}^{f(k')=f_0+k'\cdot\Delta f} d_k\cdot\sum_{p=0}^{P-1} h_p e^{-j2\pi(f_0+k\cdot\Delta f)\tau_p}=$$

$$d_k\cdot\sum_{p=0}^{P-1}\overbrace{h_p e^{-j2\pi f_0\tau_p}}^{c_p} e^{-j\overbrace{2\pi k\cdot\Delta f\tau_p}^{\phi_p}}=d_k\cdot\sum_{p=0}^{P-1} c_p\cdot e^{-j\phi_p}$$

(2): ISI: $\tau_p \geq T_g$

$$\hat{H}_k = \overbrace{\frac{1}{T}\int_0^{\tau_p - T_g} \tilde{y}(t)\cdot e^{-j\frac{2\pi}{T}kt}dt}^{\tilde{J}_k} + \frac{1}{T}\int_{\tau_p - T_g}^{T} y(t)\cdot e^{-j\frac{2\pi}{T}kt}dt =$$

$$\tilde{J}_k + \frac{1}{T}\int_0^T \left(\sum_{k'=0}^{N-1} d_k e^{j\frac{2\pi}{T}k't}\sum_{p=0}^{P-1} h_p e^{-j2\pi f(k')\tau_p} 1_{t\in[\tau_p - T_g,\tau_p + T]} + n(t)\right)\cdot e^{-j\frac{2\pi}{T}kt}dt =$$

$$\tilde{J}_k + \sum_{k'=0}^{N-1} d_k \cdot \overbrace{\sum_{p=0}^{P-1} h_p e^{-j2\pi f(k')\tau_p}}^{H(k')} \cdot \overbrace{\frac{1}{T}\int_{\tau_p - T_g}^{T} e^{-j\frac{2\pi}{T}(k-k')t}dt}^{g_p[k-k']} + \overbrace{\frac{1}{T}\int_{\tau_p - T_g}^{T} n(t)\cdot e^{-j\frac{2\pi}{T}kt}dt}^{\tilde{n}(t)} =$$

$$\boxed{\hat{H}_k = \tilde{J}_k + \sum_{k'=0}^{N-1} d_{k'}\cdot H[k']\cdot g_p[k-k'] + \tilde{n}[k]}$$

$$g_p(k-k') = e^{-j\pi\left(1+\frac{\tau_p - T_g}{T}\right)}\cdot\left(1-\frac{\tau_p - T_g}{T}\right)\cdot \overbrace{sinc\left((k-k')\cdot\overbrace{\left(1-\frac{\tau_p - T_g}{T}\right)}^{<1}\right)}^{\text{Broader than } sinc(k\text{-}k')}$$

We can see that if the longest multipath is longer that the guard interval $\tau_p \geq T_g$ we get ISI.

We are now dealing with **No ISI** case.

The matrix representation of a base-band discrete model:

$$H_k = \sum_{l=0}^{P-1} b_l e^{-j2\pi\cdot k\Delta f\cdot l\Delta\tau} \quad \overset{\Delta\tau=\frac{T}{N},T=\frac{1}{\Delta f}}{\Rightarrow} \quad \underline{H} = \underline{\underline{F}}\cdot\underline{b}$$

If the $\Delta\tau$ is not on the grid of the FFT than we get a leakage of energy to other time bins:

$$b_l = \frac{1}{N}\sum_{k=0}^{N-1} H_k e^{j\frac{2\pi}{N}k\ell} = \sum_{p=0}^{P-1} c_p \overbrace{\frac{1}{N}\sum_{k=0}^{N-1} e^{-j2\pi\cdot k\Delta f\cdot\tau_p}\cdot e^{j\frac{2\pi}{N}k\ell}}^{D\left(\frac{\tau_p}{T}-\frac{\ell}{N},N\right)} = \sum_{p=0}^{P-1} c_p D\left(\frac{\ell}{N}-\frac{\tau_p}{T},N\right)$$

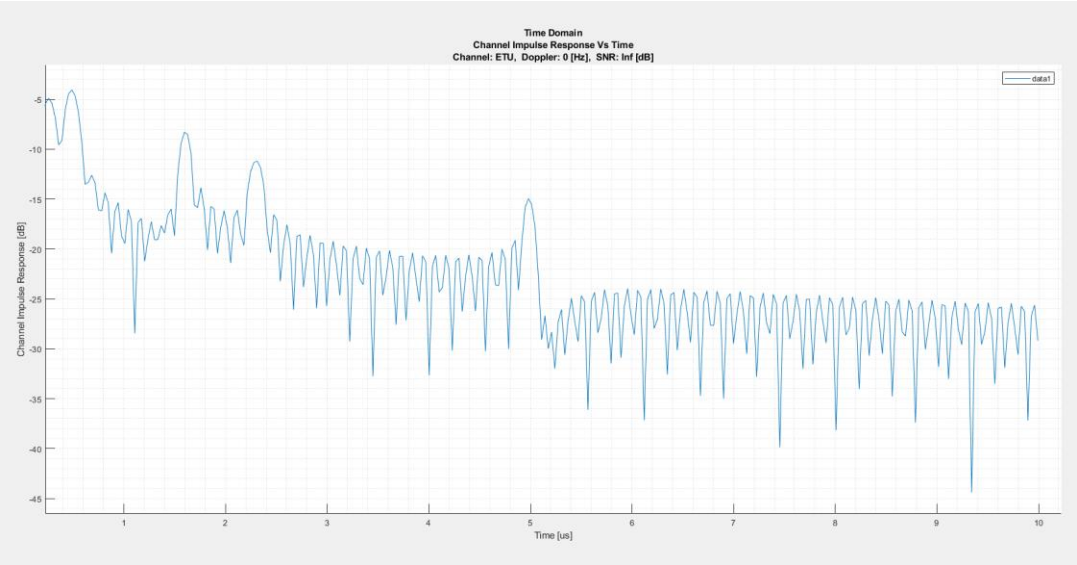
$$\boxed{b_l = \sum_{p=0}^{P-1} c_p D\left(\frac{\ell}{N}-\frac{\tau_p}{T},N\right)}$$

Where:

$$c_p = h_p e^{-j2\pi f_0\tau_p}$$

$$D(\theta,N) = e^{-j\pi\theta(N-1)}\cdot\frac{\sin(\pi\theta N)}{N\sin(\pi\theta)}$$

We have implemented a simulation that generates the Rayleigh channel in frequency and time domains:



Here we can see an example of a channel ETU in the time domain. We can see that the channel has time delay taps up to 5[us]. These tap delay times are the τ_p

So, in matrix representation we get:

$$\underline{y} = \underline{\underline{D}}\underline{x} + \underline{z} = \underline{\underline{D}}\underline{\underline{F}}\underline{b} + \underline{z}$$

Back Rotation of the Data

If we back rotate the input signal with the data D (we assume we know the data in the pilot REs):

$$\underline{x} = \underline{\underline{D}}^{-1}\underline{y} = \underline{\underline{F}}\underline{b} + \underbrace{\underline{\underline{D}}^{-1}\underline{z}}_{\tilde{\underline{z}}} = \underline{\underline{F}}\underline{b} + \tilde{\underline{z}}$$

The Matrix $\underline{\underline{D}}$ holds the transmitted data. Usually, we know it in the pilots' REs.

Filters Bank

In this work we deal with the channel estimation based on the pilots, and when we estimate the delay and doppler spread we can use filters bank to estimate the channel on all the other resource elements without the pilots using least squares solution.

We generate N realizations of the channel (AWGN, EPA, ETU) over different SNRs and doppler spreads.

The desired generated channel frequency response on the resource elements is:

$$\underline{\underline{H}}_{N\times K}$$

The back-rotated input noisy signal is:

$$\underline{\underline{X}}_{N\times M}$$

The coefficients matrix that estimates the desired channel frequency response in the desired REs based on the noisy signal is:

$$\underline{\underline{W}}_{M\times K}$$

N : #Realizations of the channel

K : #Desired REs

M : #Pilots (Known REs)

In other word we wish to find a coefficients matrix:

$$\underline{\underline{H}}_{K\times 1} = \underline{\underline{W}}^{T}_{K\times M} \cdot \underline{\underline{X}}_{M\times 1}$$

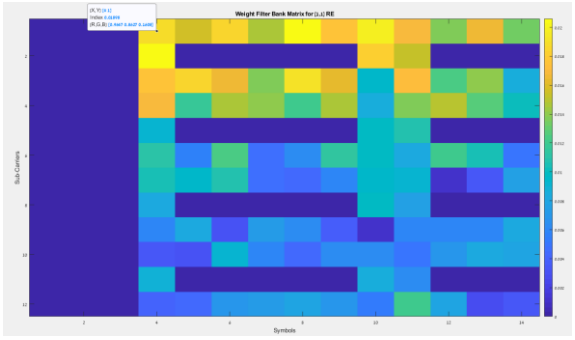
When we generated N realizations we have the following cost function:

$$\left\| \underline{\underline{H}}_{N\times K} - \underline{\underline{X}}_{N\times M} \cdot \underline{\underline{W}}_{M\times K} \right\|_2^2 \to \min$$

We solve the least squares solution:

$$\boxed{\underline{\underline{W}}^{LS}_{M\times K} = \left(\underline{\underline{X}}^{H}_{M\times N} \underline{\underline{X}}_{N\times M} \right)^{-1} \underline{\underline{X}}^{H}_{M\times N} \underline{\underline{H}}_{N\times K}}$$

For the filters bank to be accurate we need to estimate the delay and doppler spread and generate channel realizations of these estimations.



Basic Estimation Algorithms

The first algorithms we used are the: LS and LMMSE:

$$\underline{\hat{b}}_{LMMSE} = \underline{\underline{R}}_{bx}\underline{\underline{R}}_{xx}^{-1}\underline{x} = \underline{\underline{R}}_{bb}\underline{\underline{F}}^{H}\left(\underline{\underline{F}}\underline{\underline{R}}_{bb}\underline{\underline{F}}^{H} + \underline{\underline{R}}_{\tilde{z}\tilde{z}}\right)\underline{x}$$

$$\underline{\underline{R}}_{bx} = E\underline{b}\cdot\underline{x}^{H} = E\underline{b}\cdot\left(\underline{\underline{F}}\underline{b} + \tilde{\underline{z}}\right)^{H} \overset{E\underline{b}\tilde{\underline{z}}=0}{=} \underline{\underline{R}}_{bb}\underline{\underline{F}}^{H}$$

$$\underline{\underline{R}}_{xx} = E\left(\underline{\underline{F}}\underline{b} + \tilde{\underline{z}}\right)\cdot\left(\underline{\underline{F}}\underline{b} + \tilde{\underline{z}}\right)^{H} = \underline{\underline{F}}\underline{\underline{R}}_{bb}\underline{\underline{F}}^{H} + \underline{\underline{R}}_{\tilde{z}\tilde{z}}$$

$$\underline{\underline{R}}_{\tilde{z}\tilde{z}} = \sigma_{\tilde{z}}^{2} I_{N\times N}$$

$$\underline{\hat{b}}_{LS} = \left(\underline{\underline{F}}\underline{\underline{F}}^{H}\right)^{-1}\underline{\underline{F}}^{H}\underline{x}$$

The trade-off between Least Squares (LS) and Minimum Mean Square Error (MMSE) estimators primarily revolves around complexity versus performance under noise. LS is simpler and computationally less demanding but may perform worse in noisy conditions as it doesn't explicitly account for noise characteristics. MMSE, on the other hand, incorporates noise statistics for estimation, potentially offering better accuracy at the cost of increased computational complexity and the need for prior knowledge about noise characteristics.

To estimate the channel taps covariance matrix $\underline{\underline{R}}_{bb}$ we need to generate about $N_{stats} = 50\cdot 10^{3}$ channel realizations:

$$\underline{\underline{\hat{R}}}_{bb} = \frac{1}{N_{stats}}\sum_{n=0}^{N_{stats}-1}\underline{b}_{n}\underline{b}_{n}^{H}$$

We can optimize the performance by taking only the first L columns of $\underline{\underline{F}}$ since the multipath is limited to T_{MP}. So, we take the first:

$$L = \left\lceil \frac{T_{MP}}{\frac{T}{N}} \right\rceil = \left\lceil \frac{N\cdot T_{MP}}{T} \right\rceil$$

$$\underline{\underline{\tilde{F}}} = \underline{\underline{F}}_{N\times L}$$

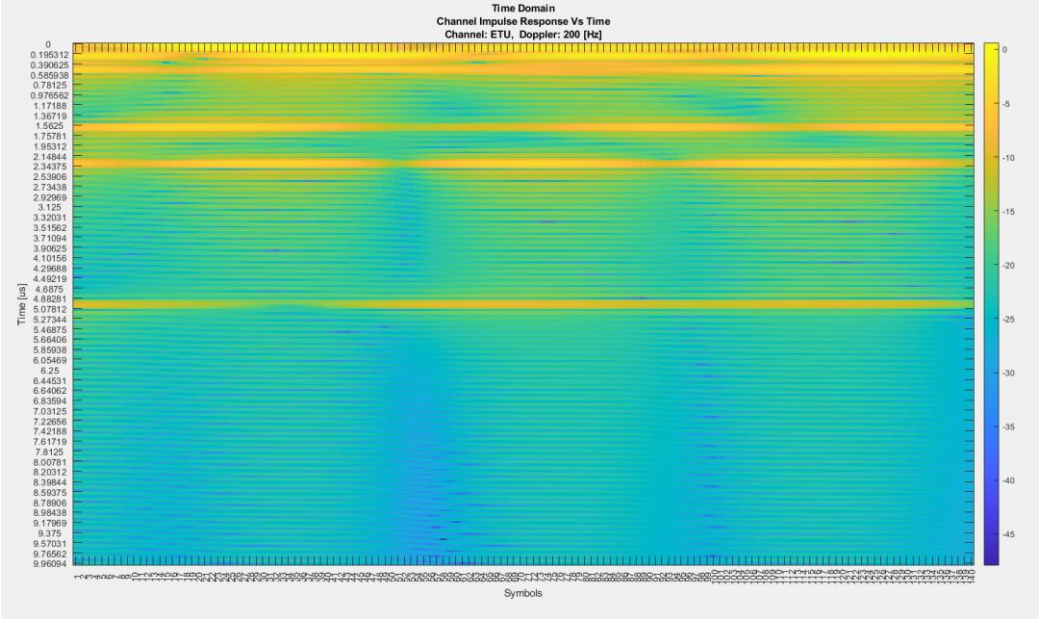
$$\underline{\hat{b}}_{LMMSE} = \underline{\underline{R}}_{bx}\underline{\underline{R}}_{xx}^{-1}\underline{x} = \underline{\underline{R}}_{bb}\underline{\underline{\tilde{F}}}^{H}\left(\underline{\underline{\tilde{F}}}\underline{\underline{R}}_{bb}\underline{\underline{\tilde{F}}}^{H} + \underline{\underline{R}}_{\tilde{z}\tilde{z}}\right)\underline{x}$$

$$\underline{\hat{b}}_{LS} = \left(\underline{\underline{\tilde{F}}}\underline{\underline{\tilde{F}}}^{H}\right)^{-1}\underline{\underline{\tilde{F}}}^{H}\underline{x}$$

The performance measurement we choose is:

$$NMSE = \frac{1}{N}\left\| \underline{\underline{H}} - \underline{\underline{\hat{H}}} \right\|_2^2$$

Now we deal with a different family of algorithms, they all use the sub-space information of the received signal $\underline{\underline{X}}_{N_{sc} \times N_{Symb}}$



The signal $\underline{\underline{X}}_{N_{sc} \times N_{Symb}}$ is drawn above. We generated an ETU channel with doppler 200[Hz]

This is the time response of the channel after FFT in the sub-carrier axes.

Most of the algorithms we use are of the super resolution type, mostly used for array signal processing for beam steering and angle of arrival estimation. The analogy is natural because the sub-carries of the OFDM are functioning exactly like ULA (Uniform Linear Array) of antenna receivers.

We will use the eigen decomposition of the OFDM symbols $\underline{\underline{X}}_{N_{sc} \times N_{Symb}}$ and use the signal sub-space, therefore we will need to perform a **spatial smoothing** of the covariance matrix.

Common Processes

The following processes are common for almost all the suggested algorithms.

Spatial Smoothing

Spatial smoothing is a technique used in array signal processing to enhance the performance of algorithms like MUSIC and ESPRIT in resolving closely spaced sources or signals in environments with correlated or coherent interference. It works by averaging across subarrays, effectively increasing the rank of the signal covariance matrix in scenarios with limited snapshots or coherent signals, thereby improving the estimator's ability to distinguish between sources. This technique can significantly enhance resolution and detection capabilities in challenging signal environments.

This process reduces the rank deficiency caused by coherent signals, as **each subarray sees a slightly different mix of the signals**. By averaging, the method exploits **spatial diversity**, **reducing the coherence between signals** and enhancing the signal matrix's rank, allowing better estimation of signal parameters like directions of arrival (DOAs).

The Eigen decomposition:

$$\underline{\underline{R}}_{xx} = E\underline{\underline{X}}\underline{\underline{X}}^H = EDE^H = \sum_i \lambda_i \underline{\phi}_i \underline{\phi}_i^H$$

We assume that the number of signal driven eigen values are less than half of the total number of the eigen values, hence we use moving average on $\underline{\underline{R}}_{xx}$ of the size:

$$N_{block} = \frac{N_{sc}}{2} = N_b$$

The number of sub-blocks we average is:

$$N = N_{sc} - N_b + 1$$

The forward smoothed covariance matrix is:

$$\underline{\underline{R}}_{xx}^f = \frac{1}{N} \sum_{k=0}^{N-1} \underline{\underline{X}}_{b,k} \cdot \underline{\underline{X}}_{b,k}^H$$

Now we also use backward spatial smoothing.

The motivation for using backward spatial smoothing, in addition to forward spatial smoothing, lies in its ability to further decorrelate coherent signals by exploiting the structure of the received signal matrix. By averaging both forward and backward processed data, it enhances the estimator's performance in identifying closely spaced sources and improves the robustness of DOA estimation algorithms, especially in environments with multipath propagation or in scenarios where signal reflections create coherent signal arrivals.

We define the flipped eye matrix:

$$\underline{\underline{J}} = flipud\left(\underline{\underline{I}}_{N_b \times N_b}\right)$$

$$\underline{\underline{R}}_{xx}^{b} = \underline{\underline{J}} \cdot \underline{\underline{R}}_{xx}^{f} \cdot \underline{\underline{J}}$$

And the forward backward spatial smoothed covariance matrix is:

$$\underline{\underline{R}}_{xx}^{fb} = \frac{1}{2}\left(\underline{\underline{R}}_{xx}^{f} + \underline{\underline{R}}_{xx}^{b}\right) = \frac{1}{2}\left(\underline{\underline{R}}_{xx}^{f} + \underline{\underline{J}} \cdot \underline{\underline{R}}_{xx}^{f} \cdot \underline{\underline{J}}\right) = \frac{1}{2N}\left(\sum_{k=0}^{N-1} \underline{\underline{X}}_{b,k} \cdot \underline{\underline{X}}_{b,k}^{H} + \underline{\underline{J}} \cdot \sum_{k=0}^{N-1} \underline{\underline{X}}_{b,k} \cdot \underline{\underline{X}}_{b,k}^{H} \cdot \underline{\underline{J}}\right)$$

At last, we apply diagonal loading for stability:

$$\boxed{\underline{\underline{R}}_{xx}^{fb} = \frac{1}{2}\left(\underline{\underline{R}}_{xx}^{f} + \underline{\underline{R}}_{xx}^{b}\right) + \varepsilon \underline{\underline{I}}}$$

QR Decomposition

Once we have the estimated covariance matrix $\underline{\underline{R}}_{xx}^{fb}$ we will apply on it the eigen decomposition algorithm.

In Matlab we use “eig” but in real life we use the QR algorithm.

To boost the running time of the QR algorithm, we reduce the bottom-right element of $\underline{\underline{R}}_{xx}^{fb}$ from the main diagonal of $\underline{\underline{R}}_{xx}^{fb}$:

$$\tilde{\underline{\underline{R}}}_{xx}^{fb} = \underline{\underline{R}}_{xx}^{fb} - \overbrace{\underline{\underline{R}}_{xx}^{fb}\left[N_b - 1, N_b - 1\right]}^{\mu}\underline{\underline{I}} = \underline{\underline{R}}_{xx}^{fb} - \mu\underline{\underline{I}}$$

μ directly uses the bottom-right element of $\underline{\underline{R}}_{xx}^{fb}$ as the shift, which is a practical and often effective approximation of the Rayleigh quotient for the purpose of accelerating the QR algorithm. This method works particularly well when $\underline{\underline{R}}_{xx}^{fb}$ is symmetric and the eigenvalues are real, as the matrix iteratively approaches a diagonal form where each diagonal element is close to an eigenvalue.

Let’s look at a general matrix $\underline{\underline{A}}$

$$\tilde{\underline{\underline{A}}} = \underline{\underline{A}} - \mu\underline{\underline{I}} = \underline{\underline{A}} - A_{N,N}\underline{\underline{I}}$$

We use the Grahm-Schmidt process:

$$\underline{\underline{A}} = \left[\underline{a}_1, \ldots, \underline{a}_N\right]$$

$$\underline{u}_1 = \underline{a}_1 \rightarrow \underline{e}_1 = \frac{\underline{u}_1}{\left\|\underline{u}_1\right\|}$$

$$\underline{u}_2 = \underline{a}_2 - \left(\underline{e}_1^H \underline{a}_2\right)\underline{e}_1 \rightarrow \underline{e}_2 = \frac{\underline{u}_2}{\left\|\underline{u}_2\right\|}$$

$$\vdots$$

$$\underline{u}_{k+1} = \underline{a}_{k+1} - \sum_{i=1}^{k}\left(\underline{e}_i^H \underline{a}_{k+1}\right)\underline{e}_i \rightarrow \underline{e}_{k+1} = \frac{\underline{u}_{k+1}}{\left\|\underline{u}_{k+1}\right\|}$$

The $\underline{\underline{Q}}, \underline{\underline{R}}$ matrices:

$$\underline{\underline{Q}} = \left[\underline{e}_1, \ldots, \underline{e}_N\right]$$

$$\underline{\underline{R}} = \begin{pmatrix} \underline{e}_1^H \cdot \underline{a}_1 & \underline{e}_1^H \cdot \underline{a}_2 & \cdots & \underline{e}_1^H \cdot \underline{a}_N \\ 0 & \underline{e}_2^H \cdot \underline{a}_2 & \cdots & \underline{e}_2^H \cdot \underline{a}_N \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \underline{e}_N^H \cdot \underline{a}_N \end{pmatrix}$$

Eigen Decomposition

In the eigen decomposition algorithm we use the QR decomposition above:

Init :

$\underline{\underline{V}} = \underline{\underline{I}}$

for $i = 1 : N_{Iter}$

$\mu = A_{N,N}$

$\left[\underline{\underline{Q}}, \underline{\underline{R}}\right] = QR\left(\underline{\underline{A}} - \mu\underline{\underline{I}}\right)$

$\underline{\underline{A}} = \underline{\underline{R}} \cdot \underline{\underline{Q}} + \mu\underline{\underline{I}}$

$\underline{\underline{V}} = \underline{\underline{V}} \cdot \underline{\underline{Q}}$

$S = \sum_{i=2}^{N}\sum_{j=i+1}^{N}\left|A_{i,j}\right|$ % sum of the off all elements above the main diagonal

if $S < tolerance$

break

$\underline{\underline{D}} = diag\left(diag\left(\underline{\underline{A}}\right)\right)$

Signal Rank Estimation

Signal rank estimation is crucial since we rely on the separation of the signal and noise subspace.

First, we sort the eigen values from the strongest to the weakest, such that:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{N_b}$$

We re-arrange the eigenvectors $\underline{\underline{E}} = \left[\underline{\phi}_1, \ldots, \underline{\phi}_{N_b}\right]$ respectively to the sorted eigenvalues.

During the research we used 2 main approaches:

Minimum Description Length

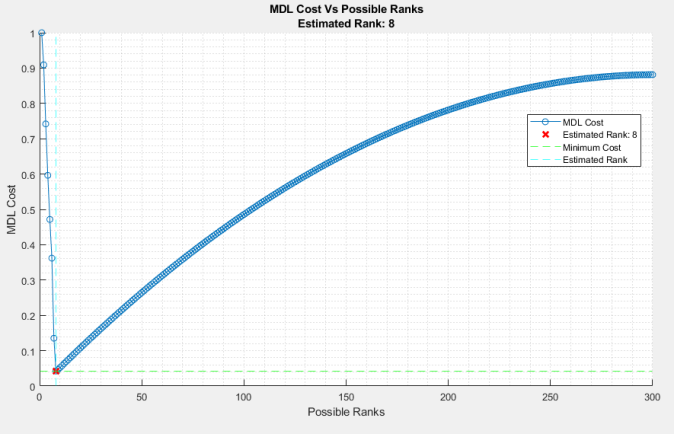
The first algorithm we implemented "Minimum Description Length" (MDL) which is taken from the paper [1].

$$MDL_{Cost}(k) = -N\left(\sum_{i=k+1}^{p} \log \lambda_i - (p-k)\log\left(\frac{1}{N-k}\sum_{i=k+1}^{p}\lambda_i\right)\right) + \frac{1}{2}k(2p-k)\log N$$

$$p = N_{block} = N_b$$

$$N = N_{Symb}$$

$$\hat{P}_{rank} = \arg\min_k \left\{MDL_{Cost}(k)\right\}$$



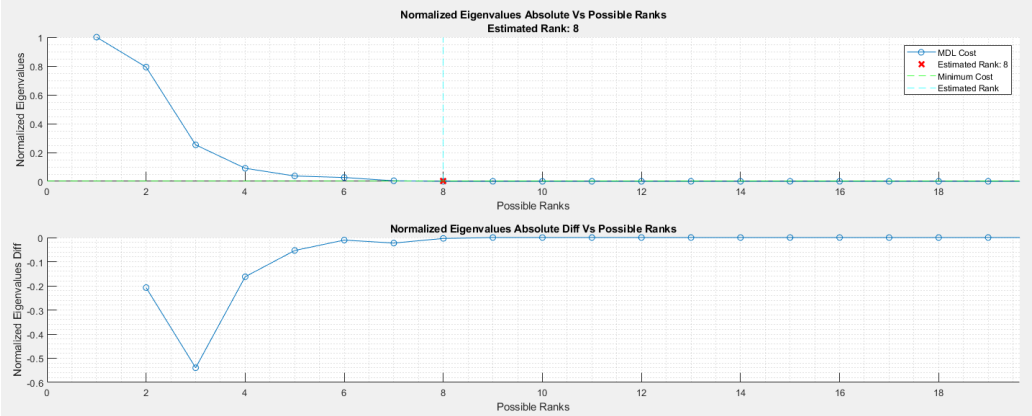
Eigenvalues derivative

The second mechanism we suggest here is to look at the derivative of the eigen values and to compare it to a threshold, when the derivative goes down below a certain threshold, we can say that we reached the noise subspace.

$$\underline{\lambda}' = \lambda[2:end] - \lambda[1:end-1]$$

$$\hat{P}_{rank} = find\left(\underline{\lambda}' < th, 1, 'first'\right)$$

$$th = 10^{-3}$$



We can see that the strongest eigenvalues are the first 8 eigenvalues, probably belong to the signal taps in τ_p

Signal and Noise Decomposition

We aim to divide the eigenvalues and eigenvectors into signal subspace and noise subspace.

$$\boxed{\underline{\underline{\tilde{R}}}_{xx}^{fb}} \triangleq \underline{\underline{R}}_{xx} = EDE^H = \sum_i \lambda_i \underline{\phi}_i \underline{\phi}_i^H = \overbrace{\sum_{i=0}^{\hat{P}_{rank}-1} \lambda_i \underline{\phi}_i \underline{\phi}_i^H}^{Signal\ Subspace} + \overbrace{\sum_{i=\hat{P}_{rank}}^{N_b-1} \lambda_i \underline{\phi}_i \underline{\phi}_i^H}^{Noise\ Subspace} =$$

$$\overbrace{\left(E_s\right)_{N_b \times \hat{P}_{rank}} \left(D_s\right)_{\hat{P}_{rank} \times \hat{P}_{rank}} \left(E_s^H\right)_{\hat{P}_{rank} \times N_b}}^{\underline{\underline{R}}_{ss} = Signal\ Subspace} + \overbrace{\left(E_n\right)_{N_b \times N_b - \hat{P}_{rank}} \left(D_n\right)_{N_b - \hat{P}_{rank} \times N_b - \hat{P}_{rank}} \left(E_n^H\right)_{N_b - \hat{P}_{rank} \times N_b}}^{\underline{\underline{R}}_{nn} = Noise\ Subspace} =$$

$$\boxed{\underline{\underline{R}}_{ss} + \underline{\underline{R}}_{nn}}$$

Because we are further ahead going to invert the signal subspace covariance matrix, we will apply on it the diagonal loading for stability:

$$\underline{\underline{\tilde{R}}}_{ss} = \underline{\underline{R}}_{ss} + \varepsilon \underline{\underline{I}} = E_s D_s E_s^H + \varepsilon \underline{\underline{I}}$$

Noise Covariance Whitening

We know that the noise a white Gaussian distributed, therefore its eigenvalues are all supposed to be equal to $\sigma_{\tilde{z}\tilde{z}}^2$.

We do not know $\sigma_{\tilde{z}\tilde{z}}^2$, therefore we estimate it using the sample mean estimator:

$$\underline{\underline{R}}_{xx} = EDE^H = E_s D_s E_s^H + E_n D_n E_n^H$$

$$D_n = diag\left(\left[\lambda_{\hat{P}_{rank}+1}, \ldots, \lambda_{N_b}\right]\right)$$

$$\boxed{\bar{\lambda}_n = \frac{1}{N_b - \hat{P}_{rank}} \sum_{i=\hat{P}_{rank}+1}^{N_b} \lambda_i}$$

$$D_n^{white} = \bar{\lambda}_n \underline{\underline{I}}_{N_b - \hat{P}_{rank}}$$

$$\underline{\underline{R}}_{xx}^{whiteNoise} = \underline{\underline{R}}_{xx}^{wn} = E_s D_s E_s^H + E_n D_n^{white} E_n^H + \varepsilon I$$

Peak Thresholding

For almost all the following algorithms we get a power spectrum where we search the strongest peaks for the estimated time delays.

For that we need to define a threshold that will differentiate the noise from the signal peaks.

We will take a CDF percentage that corresponds with the estimated rank of the signal, since we search for that number of peaks.

$$CDF_{Percent}\left[\%\right]=100\cdot\left(1-2\cdot\frac{\hat{P}_{rank}}{M}\right)$$

The threshold will be then:

$$CDF_{Th}=CDF\left(L,CDF_{Percent}\left[\%\right]\right)$$
$$Th=CDF_{Th}+median\left(L\right)$$

Where:

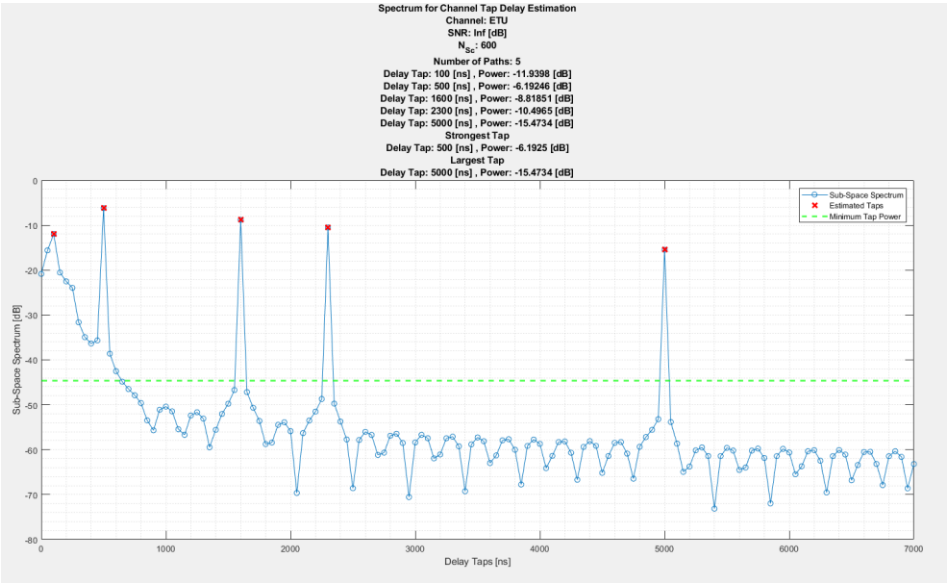
L – The cost vector on which we search the peaks

$\hat{P}_{rank}$ – the rank of the signal

M – the number of time output grid we serch the hypothesis domain of the delay tap time

We added a median of L to reduce the false alarm.

Example of the thresholding:



Delay Spread Estimation Algorithms

Channel Tap Delays Estimation

Modified MVDR

The Capon algorithm for optimal beamformer in the means of minimum variance distortionless response, has led us to the following variant.

The original beamformer is of the form:

$$\underline{x} = \underline{\underline{F}} \cdot \underline{b} + \underline{z} = \sum_{\ell=0}^{N_b-1} b_\ell \underline{v}_\ell + \underline{z}$$

$$\underline{\underline{F}} = \left[\underline{v}_0, \ldots, \underline{v}_{N_b-1} \right]$$

If we look at a single tap:

$$y_0 = \underline{w}_0^H \underline{x}_0 = \underline{w}_0^H \left(b_0 \underline{v}_0 + \underline{z} \right) = b_0 \overset{=1}{\underline{w}_0^H \underline{v}_0} + \overset{\rightarrow \min \text{var}}{\underline{w}_0^H \underline{z}}$$

Solving with Lagrange multipliers:

$$\underline{w}_\ell = \frac{R_{nn}^{-1} \underline{v}_\ell}{\underline{v}_\ell^H R_{nn}^{-1} \underline{v}_\ell}$$

Here we wish not to only bring the noise variance to minimum, but to also exploit the data of the signal and locate it with high resolution, therefore we will need the **signal covariance matrix** for that purpose, to focus the optimization on enhancing signal separation and resolution. This approach emphasizes **distinguishing between signals by maximizing the output for the signal of interest** while **minimizing interference from other signals**, leveraging differences in their covariance structures for super-resolution capabilities.

We suggest the following beamformer:

$$R_{nn}^{-1} \leftrightarrow R_{ss}^{-1} \Rightarrow \boxed{\underline{w}_\ell = \frac{R_{ss}^{-1} \underline{v}_\ell}{\underline{v}_\ell^H R_{ss}^{-1} \underline{v}_\ell}}$$

Relying on the fact that the **noise vector is perpendicular to the signal eigenvectors**.

$$y_\ell = b_\ell \underline{w}_\ell^H \underline{v}_\ell + \underline{w}_\ell^H \underline{z} \quad \overset{R_{ss}^{-1} = \left(R_{ss}^{-1}\right)^H}{=} \quad b_\ell \frac{\underline{v}_\ell^H R_{ss}^{-1}}{\underline{v}_\ell^H R_{ss}^{-1} \underline{v}_\ell} \underline{v}_\ell + \frac{\underline{v}_\ell^H R_{ss}^{-1}}{\underline{v}_\ell^H R_{ss}^{-1} \underline{v}_\ell} \underline{z} = b_\ell \frac{\overbrace{\underline{v}_\ell^H R_{ss}^{-1} \underline{v}_\ell}^{=1}}{\underline{v}_\ell^H R_{ss}^{-1} \underline{v}_\ell} + \frac{\overbrace{\underline{v}_\ell^H R_{ss}^{-1} \underline{z}}^{\rightarrow \min \text{var}}}{\underline{v}_\ell^H R_{ss}^{-1} \underline{v}_\ell}$$

$$\underline{v}_\ell \perp \underline{z} \Rightarrow R_{ss}^{-1} \underline{z} \rightarrow \min$$

Now we project the beamformer to the input signal $\underline{\underline{X}}_{b,0}$:

$$\underline{\underline{W}} = \left[\underline{w}_0, \ldots, \underline{w}_{N_b-1} \right]$$

$$\underline{L}_{M \times 1} = Average\_Symbols \left\{ \left(\underline{\underline{W}}^H \right)_{M \times N_b} \cdot \left(\underline{\underline{X}}_{b,0} \right)_{N_b \times N_{symb}} \right\}$$

MUSIC - Inverse

MUSIC was originally created to find angle of arrival for array processing of antennas. As we mentioned above there is a strong correlation between channel estimation in OFDM to beam steering. The sub-carriers' function like the receiving elements in phased array.

Beam Steering:

$$v_k(t) = \sum_{p=1}^{P} \overbrace{u_p(t)}^{Transmitted\ Signal}\ e^{-j2\pi f_0(k-1)\frac{d}{c}\sin\theta_p} + z_k(t)$$

$$\theta_p - Angle\ Of\ Arival$$

$$\theta_p \leftrightarrow \tau_p$$

$$\underline{v}_{N_b\times 1}(t) = \underline{\underline{A}}_{N_b\times P}\underline{u}_{P\times 1}(t) + \underline{z}_{N_b\times 1}(t)$$

$$\underline{\underline{A}} = \left[\underline{a}(\theta_0),\ldots,\underline{a}\left(\theta_{N_b-1}\right)\right]$$

$$\underline{a}\left(\theta_p\right) = \begin{pmatrix} 1 \\ e^{-j2\pi f_0\frac{d}{c}\sin\theta_p} \\ \vdots \\ e^{-j2\pi f_0(k-1)\frac{d}{c}\sin\theta_p} \end{pmatrix}$$

Channel Estimation:

$$H_k = \sum_{p=0}^{P-1} c_p \overbrace{e^{-j2\pi\cdot k\Delta f\cdot\tau_p}}^{s_k(\phi_p)}$$

$$c_p = h_p e^{-j2\pi\cdot f_0\cdot\tau_p}$$

$$\phi_p = 2\pi\cdot\Delta f\cdot\tau_p$$

$$\underline{c}_{p\times 1} = \begin{pmatrix} h_1 e^{-j2\pi\cdot f_0\cdot\tau_1} \\ \vdots \\ h_p e^{-j2\pi\cdot f_0\cdot\tau p} \end{pmatrix}$$

$$\underline{s}_{N_b\times 1}\left(\phi_p\right) = \begin{pmatrix} e^{-j2\pi\cdot\Delta f\cdot\tau_p} \\ \vdots \\ e^{-j2\pi\cdot(N_b-1)\Delta f\cdot\tau_p} \end{pmatrix} \Rightarrow \underline{\underline{S}}_{N_b\times p} = \left[\underline{s}(\phi_1),\ldots,\underline{s}\left(\phi_p\right)\right]$$

$$\underline{x}_{N_b\times 1} = \underline{\underline{S}}_{N_b\times p}\cdot\underline{c}_{p\times 1} + \underline{\tilde{z}}_{N_b\times 1}$$

$$\underline{\underline{R}}_{xx} = E\underline{x}\underline{x}^H = E\left(\underline{\underline{S}}\cdot\underline{c} + \underline{\tilde{z}}\right)\left(\underline{\underline{S}}\cdot\underline{c} + \underline{\tilde{z}}\right)^H = SR_{cc}S^H + \sigma_{\tilde{z}\tilde{z}}^2 I$$

The matrix S is of the form:

$$\left(S\right)_{k,p} = \alpha_p^{k-1}, \alpha = e^{-j2\pi\Delta f\tau_p} \in C$$

Therefore S is **Vandermonde** matrix:

$$\boxed{rank\left(R_{cc}\right)=P \Rightarrow rank\left(SR_{cc}S^{H}\right)=P}$$

Which means that we can extract the strongest $\hat{P}_{rank}$ eigenvalues and refer them to the signal subspace. And the $N_b-\hat{P}_{rank}$ weakest eigenvalues and refer them to the noise subspace.

$$\overbrace{\lambda_1 \geq \ldots \geq \lambda_{\hat{P}_{rank}}}^{Signal} \geq \overbrace{\lambda_{\hat{P}_{rank}+1} \geq \ldots \lambda_{N_b}}^{Noise}$$

$$\phi_\ell \leftrightarrow \lambda_\ell$$

We know that the signal is perpendicular to the noise space eigenvectors:

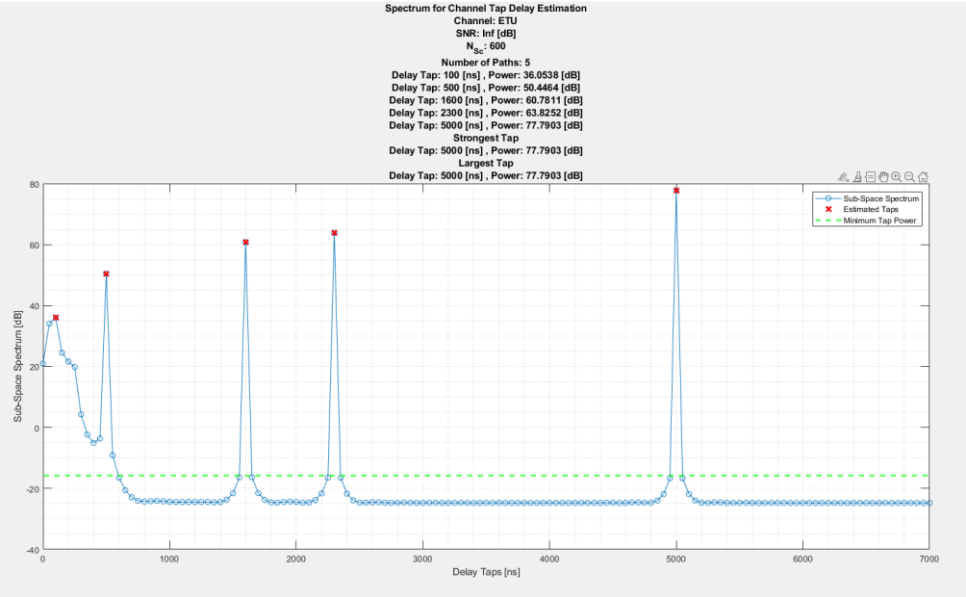
$$\overbrace{\left\{\underline{s}\left(\phi_1\right),\ldots,\underline{s}\left(\phi_{\hat{P}_{rank}}\right)\right\}}^{Signal} \perp \overbrace{\left\{\underline{\phi}_{\hat{P}_{rank}+1},\ldots,\underline{\phi}_{N_b}\right\}}^{Noise}$$

$$\underline{\underline{E}}_n=\left[\underline{\phi}_{\hat{P}_{rank}+1},\ldots,\underline{\phi}_{N_b}\right]$$

The **null spectrum** will be:

$$\boxed{L(\ell)=f_{null,\ell}=\left(\underline{v}_\ell^H \underline{\underline{E}}_n \underline{\underline{E}}_n^H \underline{v}_\ell\right)^{-1}}$$

We Will get a peak where the actual signals are because the correlation between the steering vectors $\underline{v}_\ell$ and the noise eigenvectors $\underline{\underline{E}}_n$ will be close to 0.



We can see the clean and strong peaks where the signal taps are.

MUSIC - Roots

To find the roots of the null space we will transform the covariance matrix to real form and look for the roots using the Newton's Method. We will classify the true roots as the ones inside the unit circle and closest to it.

We take the real of the covariance matrix:

$$R_{xx}^{\mathrm{Re}} = \mathrm{Re}\left\{U^H R_{xx} U\right\}$$

Where $\underline{\underline{U}}_{N_b \times N_b}$ is the real transformation matrix:

$$N_b - even:$$

$$U_{N_b \times N_b} = \frac{1}{\sqrt{2}} \begin{pmatrix} I & jI \\ J & -jJ \end{pmatrix}, \qquad J = flipud\left(I\right)$$

$$N_b - odd:$$

$$U_{N_b \times N_b} = \frac{1}{\sqrt{2}} \begin{pmatrix} I & 0 & jI \\ 0 & \sqrt{2} & 0 \\ J & 0 & -jJ \end{pmatrix}, \; J = flipud\left(I\right)$$

The null spectrum:

$$f_{null}^{\mathrm{Re}}\left(z\right) = s^T\left(\frac{1}{z}\right) \cdot UE_{n,\mathrm{Re}} E_{n,\mathrm{Re}}^T U^H s\left(z\right)$$

$$\underline{v}_\ell = \begin{bmatrix} z_\ell^0 \\ z_\ell^1 \\ \vdots \\ z_\ell^{N_b-1} \end{bmatrix}, \quad z_\ell = e^{-j\phi_\ell}, \qquad \phi_\ell = 2\pi\Delta f\, \tau_\ell$$

We find the roots using the Newton's method:

$$\underline{v}^{init} = \begin{bmatrix} e^{-j2\pi\Delta f \tau_0} \\ \vdots \\ e^{-j2\pi\Delta f \tau_{M-1}} \end{bmatrix}_{M\times 1}$$

$$for\ m = 1:M$$

$$z = v_m$$

$$for\ i = 1:N_{iter}$$

$$f'(z) = \frac{f(z+\delta) - f(z)}{\delta}$$

$$z_{new} = z - \frac{f(z)}{f'(z)}$$

$$if\ |z_{new} - z| < th$$

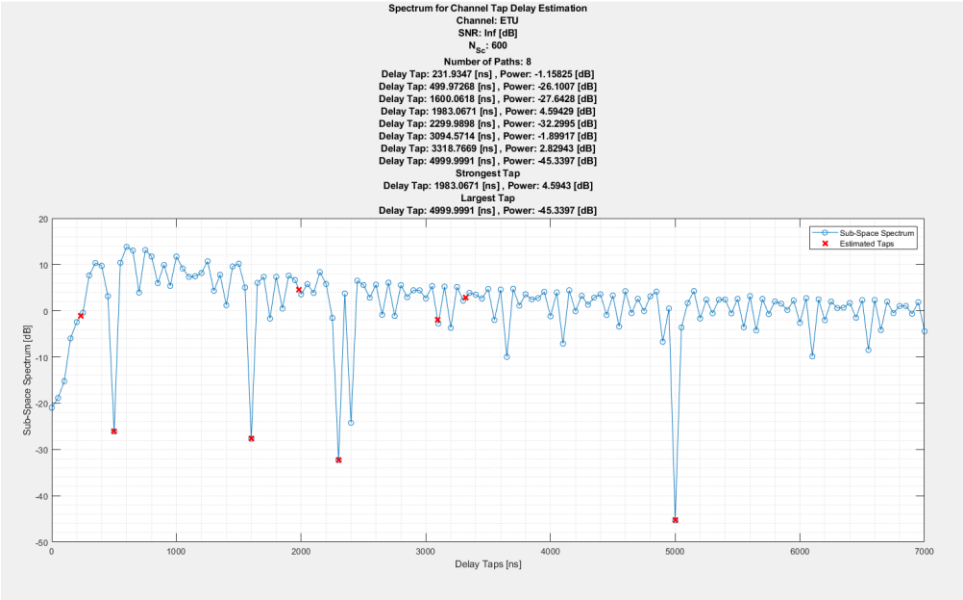
$$roots(m) = z_{new}$$

$$break$$

$$z = z_{new}$$

After we get all the roots, we take the closest $\hat{P}_{rank}$ roots to the unit circle

$$\overset{\leq 1}{1 - |roots|} \rightarrow \min$$



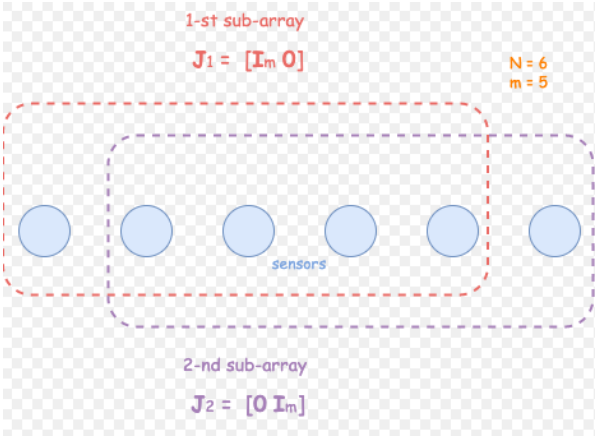
This method is less robust than the inverse version of MUSIC, due to the iterative Newton's method roots finding and taking only the real part of the covariance matrix, losing the information of the image part.

ESPRIT

Estimation of signal parameters via rotational invariance techniques.

The ESPRIT algorithm is motivated by the need to estimate the directions of arrival (DOAs) of multiple signal sources with high accuracy and resolution, especially in scenarios where signal sources are closely spaced. ESPRIT exploits the invariance properties of signal subspaces to estimate DOAs without needing to search over all possible directions, significantly reducing computational complexity compared to methods like MUSIC. It's particularly useful in array signal processing applications, offering efficient and precise DOA estimation.

Example of the sensor array division for ESPRIT:



For ESPRIT we need the signal space eigenvectors, and we have it at the output of the signal and noise decomposition block:

$$\underline{\underline{\tilde{R}}}_{xx}^{fb} \triangleq \underline{\underline{R}}_{xx} = E_s D_s E_s^H + E_n D_n E_n^H$$

$$\underline{\underline{\tilde{R}}}_{xx}^{fb} \Rightarrow E_s$$

We define 2 matrices: $\left(\underline{\underline{S}}_1\right)_{N_b \times \hat{P}_{rank}-1}, \left(\underline{\underline{S}}_2\right)_{N_b \times \hat{P}_{rank}-1}$

$$\boxed{\begin{aligned}\left(\underline{\underline{S}}_1\right)_{N_b \times \hat{P}_{rank}-1} &= E_s\left[1:N_b-1,:\right] \\ \left(\underline{\underline{S}}_2\right)_{N_b \times \hat{P}_{rank}-1} &= E_s\left[2:N_b,:\right]\end{aligned}}$$

ESPRIT uses the **rotational information between 2 consecutive subcarriers** to measure the phase shift that was caused by each channel delay tap.

$$s\left(\phi_p\right) = e^{-j2\pi \cdot \Delta f \cdot \tau_p}$$

$$\boxed{\psi_p = e^{-j\phi_p}}$$

Now we define the matrix Ψ as the least squares solution for ψ_p:

$$\boxed{\underline{\underline{\Psi}} = \left(\underline{\underline{S}}_1^H \underline{\underline{S}}_1\right)^{-1} \underline{\underline{S}}_1^H \underline{\underline{S}}_2}$$

The eigen decomposition of Ψ :

$$\underline{\underline{\Psi}} = \underline{\underline{E}}_{\psi} \underline{\underline{D}}_{\psi} \underline{\underline{E}}_{\psi}^{H}$$
$$\underline{\lambda}_{\psi} = diag\left(\underline{\underline{D}}_{\psi}\right)$$

Extract the time delay taps from the eigen values of Ψ :

$$\boxed{\tau_p = \frac{-\measuredangle \lambda_{\psi_p}}{2\pi\Delta f}}$$

This method is very effective since it is not searching a time search space for the signal taps but exploits the mathematical connection of the taps to the eigenvalues of the signal.

```
K>> chosenTaps*1e6

ans =

    0.0272
    0.0954
    0.0954
    0.2319
    0.4999
    1.6000
    2.3000
    5.0000
```

Modified OMP

The **Orthogonal Matching Pursuit** (OMP) algorithm is an iterative method used for sparse signal recovery, aiming to find the best matching projections of a signal onto a dictionary of basis functions. **OMP selects the dictionary element that is most correlated with the current residual**, adds it to the model, and updates the residual accordingly. This process repeats until a stopping criterion is met, such as a predefined number of iterations or a residual error threshold.

The original OMP algorithm [2], uses the over-complete dictionary of the FFT matrix:

$$F_{IK} = \begin{bmatrix} W_{0,0} & \cdots & W_{0,IK-1} \\ \vdots & \ddots & \vdots \\ W_{K-1,0} & \cdots & W_{K-1,IK-1} \\ \vdots & \ddots & \vdots \\ W_{IK-1,0} & \cdots & W_{IK-1,IK-1} \end{bmatrix} \Rightarrow F_{K\times IK} = \begin{bmatrix} W_{0,0} & \cdots & W_{0,IK-1} \\ \vdots & \ddots & \vdots \\ W_{K-1,0} & \cdots & W_{K-1,IK-1} \end{bmatrix}_{K\times IK}$$

$$W_{k,l} = e^{-j\frac{2\pi}{IK}kl}$$

The main objective is to maximize the projection power of the dictionary atoms (the linear phase exponents of the FFT and find the delay time taps of the signal).

Doing that with the **FFT matrix gives us no better resolution than using FFT over the subcarrier's axes**, so we integrate this approach with the **modified MVDR**, mentioned above for the following algorithm:

First, we define the over complete dictionary:

$$\left(\underline{\underline{F}}_{OC}\right)_{N_b\times M} = e^{-j2\pi \underline{f}_{grid}\cdot \underline{t}_{grid}^{T}}$$

$$\left(\underline{f}_{grid}\right)_{N_b\times 1} = \begin{bmatrix} 0 \\ \Delta f \\ \vdots \\ \Delta f\cdot\left(N_b-1\right) \end{bmatrix}_{N_b\times 1}$$

$$\underline{t}_{grid}^{T} = \left[0, \Delta t_p, \ldots, \Delta t_p\cdot\left(M-1\right)\right]_{1\times M}$$

Now we generate the Modified MVDR W matrix:

$$\boxed{\begin{aligned} \underline{\underline{W}} &= \left[\underline{w}_0, \ldots, \underline{w}_{N_b-1}\right] \\ \underline{w}_\ell &= \frac{R_{ss}^{-1}\underline{v}_\ell}{\underline{v}_\ell^{H} R_{ss}^{-1}\underline{v}_\ell} \end{aligned}}$$

Now the OMP algorithm:

$$Init:$$

$$\underline{\underline{R}}_0 = \underline{\underline{X}}_{N_b \times N_{symb}}$$

$$W_0 = \{\ \}$$

$$J_0 = \{\ \}$$

$$for\ t = 1 : \hat{P}_{rank}$$

$$\underline{L}_{M\times 1} = \left| Average\_Symbols\left\{ \left(\underline{\underline{W}}^H \right)_{M\times N_b} \cdot \underline{r}_t \right\} \right|$$

$$L_{\max} = \max\left\{ \underline{L} \right\}$$

$$j_t = \arg\max_j \left\{ L_j \right\}$$

$$if\ \ L_{\max} < Th$$

$$break$$

$$J_t = J_{t-1} \cup \left\{ j_t \right\}$$

$$\underline{\underline{W}}_t = \left[W_{t-1}, \underline{w}_{j_t} \right]$$

$$\left(\underline{\underline{\hat{a}}}_{LS} \right)_{t\times N_{symb}} = \arg\min_{\underline{a}} \left\| \underline{\underline{X}} - \underline{\underline{W}}_t \cdot \underline{a} \right\|_2^2 = \left(\underline{\underline{W}}_t^H \underline{\underline{W}}_t \right)^{-1} \underline{\underline{W}}_t^H \underline{\underline{X}}$$

$$\underline{\underline{\hat{X}}} = \underline{\underline{W}}_t \cdot \underline{\underline{\hat{a}}}_{LS}$$

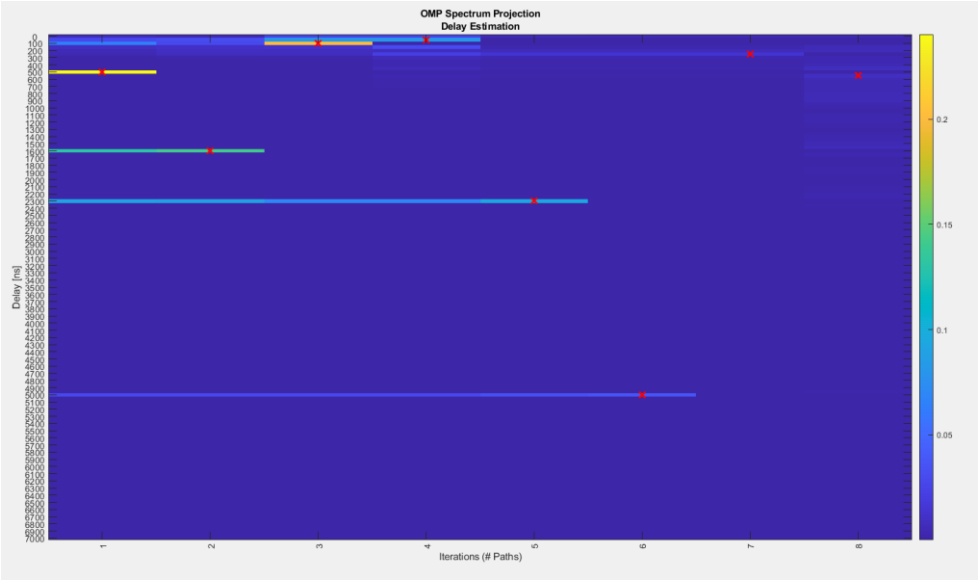
$$\underline{\underline{R}}_t = \underline{\underline{X}} - \underline{\underline{\hat{X}}} \qquad \% \text{ Remove the projection of the last found signal atom}$$

We can see that:

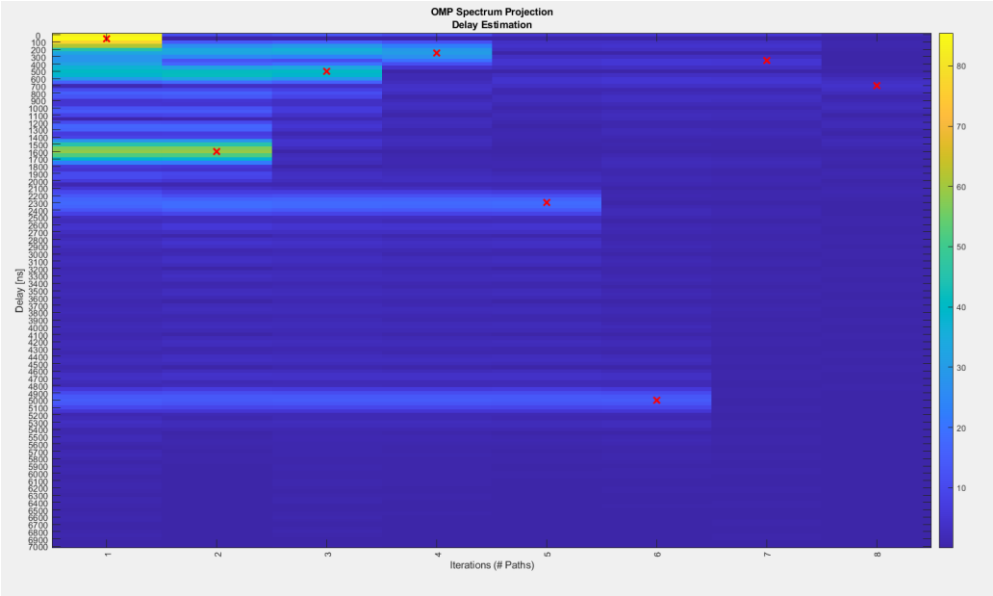
$$\underline{\underline{R}}_t \perp \underline{\underline{W}}_t$$

The chosen taps are the time output grid in the found indices:

$$\underline{t}_{grid}^T \left[J_t \right]$$



We can see what we would get if we used the original over-complete dictionary using the FFT matrix:



The resolution is the native resolution of the BW we are dealing with:

$$T_{res} = \frac{1}{BW} = \frac{1}{N_{sc} \cdot \Delta f} = \frac{1}{600 \cdot 15[KHz]} \approx 100[ns]$$

We can also see the sinc energy that is leaked to the adjacent time bins.

Channel Tap Phasors Estimation

Now that we estimated the tap delays: τ_p we need to estimate the phasors: c_p

We get the estimated matrix:

$$\underline{S}_p = e^{-j2\pi \underline{f}\cdot\hat{\tau}_p} = \begin{bmatrix} e^{-j2\pi 0\cdot\hat{\tau}_p} \\ e^{-j2\pi\Delta f\cdot\hat{\tau}_p} \\ \vdots \\ e^{-j2\pi\Delta f\left(N_{sc}-1\right)\cdot\hat{\tau}_p} \end{bmatrix}$$

$$\underline{\underline{S}}_{N_{sc}\times\hat{P}_{rank}} = \left[\underline{S}_0,\ldots,\underline{S}_{\hat{P}_{rank}-1}\right]$$

For a single symbol we solve the **least squares** problem:

$$\underline{x}_{N_b\times 1} = \underline{\underline{S}}_{N_b\times p}\cdot\underline{c}_{p\times 1} + \underline{\tilde{z}}_{N_b\times 1}$$

$$\underline{\hat{c}}_{LS} = \left(\underline{\underline{S}}^H\underline{\underline{S}}\right)^{-1}\underline{\underline{S}}^H\underline{x}$$

Now for all symbols:

$$\boxed{\underline{\underline{\hat{c}}}_{LS} = \left(\underline{\underline{S}}^H\underline{\underline{S}}\right)^{-1}\underline{\underline{S}}^H\underline{\underline{X}}}$$

Channel Delay Spread Estimation Performance

We measure the RMS of the delay taps for the delay spread estimation:

$$\overline{\tau}_1 = \frac{\sum_{p=0}^{P-1}\left|c_p\right|^2\tau_p}{\sum_{p=0}^{P-1}\left|c_p\right|^2}, \qquad \overline{\tau}_2 = \frac{\sum_{p=0}^{P-1}\left|c_p\right|^2\tau_p^2}{\sum_{p=0}^{P-1}\left|c_p\right|^2}$$

$$\overline{\tau}_{RMS} = \sqrt{\overline{\tau}_2 - \overline{\tau}_1^2}$$

The measurement of the error will be RMSE (so we still get the units of time in seconds):

$$RMSE = \sqrt{\frac{1}{N_{symb}}\sum_{i_{symb}=0}^{N_{symb}-1}\left|\hat{\overline{\tau}}_{RMS,i_{symb}} - \overline{\tau}_{RMS,i_{symb}}\right|^2}$$

Doppler Spread Estimation Algorithms

For doppler spread estimation we use similar methods to delay spread estimation, but now on the symbols' axes.

Because we deal here with doppler spread that is spread over a bandwidth we tried calculating the autocorrelation function (ACF) out of the diagonals of the covariance matrix and compare it to the modified Bessel function of order 0: J_0

$$y(t)=\sum_{p=0}^{P-1}s(t-\tau_p)\overbrace{h_p\sum_m e^{j\phi_{p,m}}\cdot e^{-j2\pi f_d\cos\alpha_{p,m}t}}^{a_p(t)\rightarrow\text{Minor MP}}$$

$$R(\tau)=E\left[a_p(t)a_p^*(t+\tau)\right]=E\left[h_p\sum_m e^{j\phi_{p,m}}\cdot e^{-j2\pi f_d\cos\alpha_{p,m}t}h_p^*\sum_k e^{-j\phi_{p,k}}\cdot e^{j2\pi f_k\cos\alpha_{p,k}(t+\tau)}\right]=$$

$$E\left[|h_p|^2\sum_m\sum_k e^{j(\phi_{p,m}-\phi_{p,k})}\cdot e^{-j2\pi(f_d\cos\alpha_{p,m}-f_d\cos\alpha_{p,k})t}\cdot e^{j2\pi f_d\cos\alpha_{p,k}\tau}\right]\overset{\phi_{p,m},\underbrace{\alpha_{p,m}\ iid}}{=}$$

$$\boxed{R(\tau)=|h_p|^2\sum_m E_\alpha e^{j2\pi f_d\cos\alpha_{p,m}\tau}}\Rightarrow J_0$$

$$\boxed{S(f)}=\int_0^T R(\tau)e^{-j2\pi f\tau}d\tau=\int_0^T|h_p|^2\sum_m E_\alpha e^{j2\pi f_d\cos\alpha_{p,m}\tau}e^{-j2\pi f\tau}d\tau=$$

$$|h_p|^2\sum_m E_\alpha\int_0^T e^{j2\pi f_d\cos\alpha_{p,m}\tau}e^{-j2\pi f\tau}d\tau=|h_p|^2\sum_m E_\alpha\overbrace{\int_0^T e^{-j2\pi(f-f_d\cos\alpha_{p,m})\tau}d\tau}^{\delta(f-f_d\cos\alpha_{p,m})}=$$

$$|h_p|^2\sum_m\frac{1}{2\pi}\int_0^{2\pi}\delta(f-f_d\cos\alpha_{p,m})d\alpha=|h_p|^2\sum_m\frac{1}{\pi}\overset{\substack{z=f_d\cos\alpha\\ dz=-f_d\sin\alpha\cdot d\alpha\\ \sin\alpha=\sqrt{1-\cos^2\alpha}=\sqrt{1-\frac{z^2}{f_d^2}}\\ [0,\pi]\rightarrow[f_d,\underbrace{-f_d}]}}{\int_0^\pi}\delta(f-f_d\cos\alpha_{p,m})d\alpha=$$

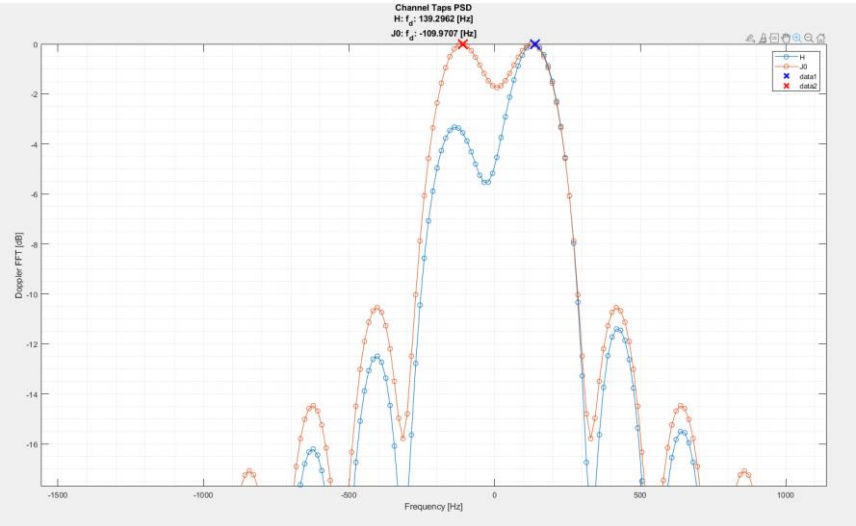
$$\frac{|h_p|^2}{\pi}\int_{-f_d}^{f_d}\frac{1}{f_d\sqrt{1-\frac{z^2}{f_d^2}}}\delta(f-z)dz=\boxed{\frac{|h_p|^2}{\pi}\cdot\frac{1}{f_d\sqrt{1-\frac{f^2}{f_d^2}}}\cdot 1_{|f|<f_d}}\Rightarrow \textit{Jakes Distribution}$$

The estimation of $R(\tau)$ is done by taking the **sample mean estimator over each of the diagonal lags** of the covariance matrix $\underline{\underline{\tilde{R}}}_{xx}^{fb}$

We deal with several methods for doppler spread estimation:

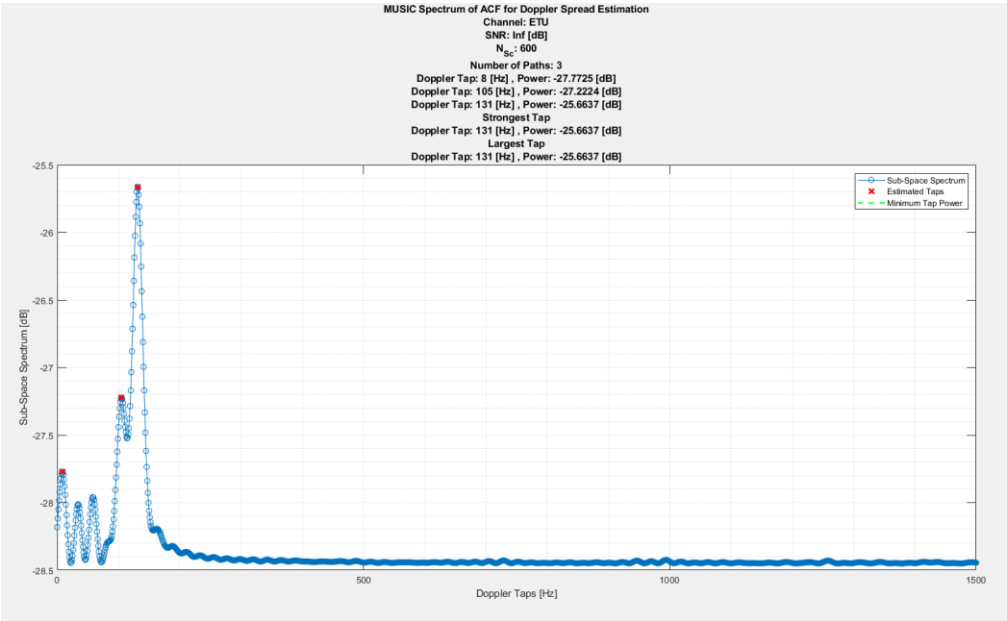
ACF FFT

As we have done in the above equations' derivations of the PSF of doppler spread, we apply FFT over the ACF to get the PSD and then we search for the largest peak (furthest in Hz) in frequency to estimate $\hat{f}_d$:



The estimated $\hat{f}_d$ is 139[Hz] where we simulated 124[Hz], we see also that the J_0 PSD itself gives 110[Hz] which is also not so close to 124[Hz]

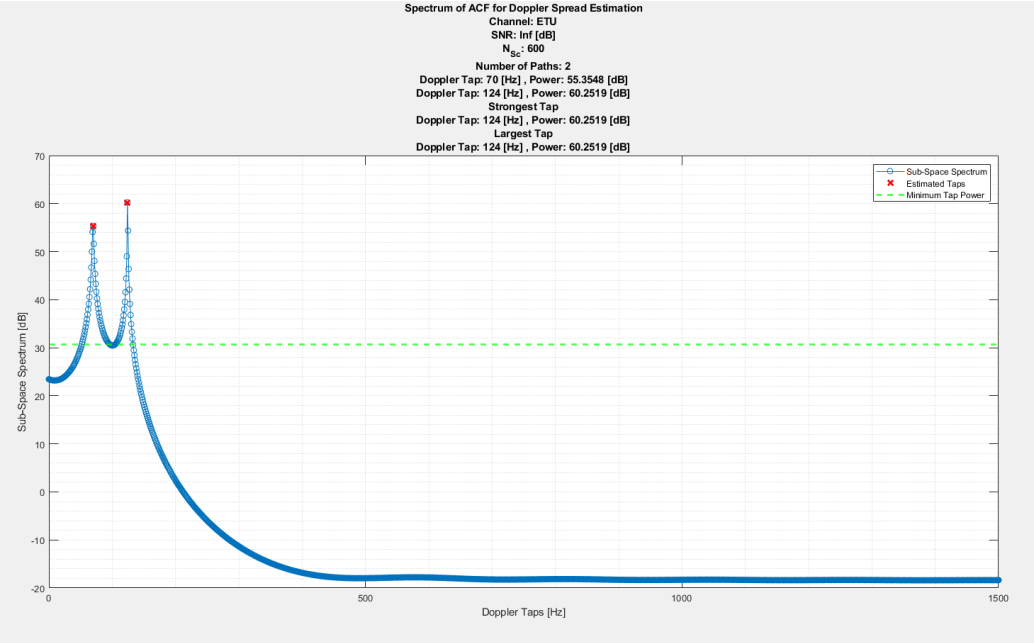
ACF Sub-Space



Like in the delay spread estimation we apply a sub-space algorithm like MUSIC on the ACF and get the above signal spectrum projection power.
We see that we are closer, but we needed to increase the number of symbols by 10 to get that close estimation.

Estimated Delay Taps Sub-Space

The algorithm we have suggested is the subspace algorithm like in the delay spread estimation, but we are doing it on the estimated taps from the delay estimation stage.



We get exactly 124[Hz] as simulated.

Channel Doppler Spread Estimation Performance

We measure the doppler spread estimation error by the normalized absolute error from the actual doppler spread:

$$err = \frac{\left| \hat{f}_d - f_d \right|}{f_d}$$

References

[1]: Determining the number of signals by information theoretic criteria
https://ieeexplore.ieee.org/document/1172389

[2]: Channel Estimation for OFDM Signals
https://repository.library.northeastern.edu/files/neu:cj82pr619/fulltext.pdf

[3]: Performance of ESPRIT and Root-MUSIC for Angle-of-Arrival(AOA) Estimation
https://ieeexplore.ieee.org/document/8690541

[4]: Doppler spread estimation in high mobility wireless communication - Zichen Wang
https://ro.uow.edu.au/cgi/viewcontent.cgi?article=5433&context=theses