Signal Processing Overview

João Morais

July 18, 2019

Abstract

If Signal Processing could be described in one sentence it would be "How to find something impossible to find using an immense amount of mathematics that are not necessary to apply the final result". Signal Processing in Communications, more specifically, Array Signal Processing by applying some strong logic to the what is received in the sensors can estimate the channel and effectively receive a signal that has been attenuated, interfered by other signals in the same frequency and distorted by the environment and by itself. Now multiply this by the number of users in a multipath environment and it might become possible to imagine the tip of the iceberg.

Contents

1 Intro **4**

2 Coherent adding - The Matched filter **4**
2.1 Matched Filter vs Zero Forcing vs Wiener Filter 6

3 Important math **7**
3.1 Norms 7
3.2 Means and Variances 7
3.2.1 Variance = Power? SNR(σ) 9
3.3 Unitary Matrix, Isometry and Projections 9
3.4 Singular Value Decomposition 11
3.5 Matrices Perspectives and Spaces Relations 13
3.6 SVD geometrical Interpretation 15
3.7 Pseudo-Inverse 16
3.8 Short and Important considerations 18

4 Constant Modulus Algorithm **18**

5 CDMA **21**
5.1 Review: Convolution 22
5.2 Perspective: Spread Spectrum in 3G 23

6 OFDM **29**

7 Space-Time Coding **33**

8 All code explained **36**
8.1 First assignment 36
8.1.1 Instantaneous Model 36
8.1.2 Convolutive Model 41
8.2 Second Assignment 43
8.2.1 Receiver algorithms for the instantaneous Model 43
8.2.2 Direction of Arrival Estimation 45
8.2.3 Receiver algorithms for convolutive model 47
8.3 Third Assignment 47
8.3.1 Pilot-based Estimation 48

9 Blind Estimation **48**

10 Final Assignment **53**
10.1 Estimation of Directions and Frequencies 53
10.1.1 Signal Model 53
10.1.2 Estimation of directions 54
10.1.3 Estimation of frequencies 56
10.1.4 Comparison 56
10.1.5 Constant Modulus Algorithm 56

11 Extra Slides **57**
11.1 Channel Estimation and Hybrid Precoding for mmWaves . . . 57
11.2 UltraSound 57

12 Evaluation **58**
12.1 Relation of single values after sensors or samples change . . . 58
12.2 Wiener and ZF receivers 58
12.3 Blind Estimation deduction and nullspaces relations 58

1 Intro

What is Space-Time Processing?

Simply put, is processing in space and in time. If there is an array of sensors, there is space processing and if contributions to the same signal come at different time instants, is necessary time processing as well to decide what to account as part of the signal.

There are many advantages of using an array of sensors and STP (Space-Time Processing):

- Improve Capacity:
- Improve Coverage:
- Improve Quality:
- Reduces Co-channel Interference (CCI):
- Enhances Diversity:
- Improves Gain:

2 Coherent adding - The Matched filter

Nothing more than adding in a useful way.

When a signal comes from a given directions, the sensors in the array will sense different signals in the same time instant. They will also experience different noises. If we know what are we suppose to observe throughout the sensor array for a given signal, we can check if that is what we indeed received. If that is the case, is because that was the transmitted signal.

We can do even further enhancements: in a ULA (Uniform Linear Array), the difference of the signal received in adjacent elements is preserved. Therefore, one can some all the signals scaled by the inverse relation between them. For instance, if the first element receives s_0 and the second element receives $s_1 = s_0 \times a$ then we can sum them as $s_o + s_1/a$ and we will get double the first signal and we will cancel out the noise!

This is know as the Matched Filter.

If one cares to write more precisely:

$$x_i(t) = a(\theta)\beta s(t)e^{-j2\pi f_c r_i} \quad (1)$$

steering vector

Where $a(\theta)$ is the gain of the antenna in the direction of the signal, $s(t)$ is the transmitted signal (already not counting with the delay of propagation due to narrowband assumptions), f_c is the central frequency of the signal and r_i being the distance difference in reference with the first element.

The M received signals can be written as:

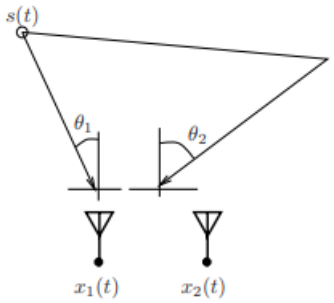
$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_M(t) \end{bmatrix} = \begin{bmatrix} 1 \\ e^{j2\pi\Delta_2 \sin(\theta)} \\ \vdots \\ e^{j2\pi\Delta_M \sin(\theta)} \end{bmatrix} a(\theta)\beta s(t) =: \mathbf{a}(\theta)\beta s(t)$$

Now, two cases are worth distinguishing:

- For more than one source, the received signal becomes a superposition of various signals (represented for two sources):

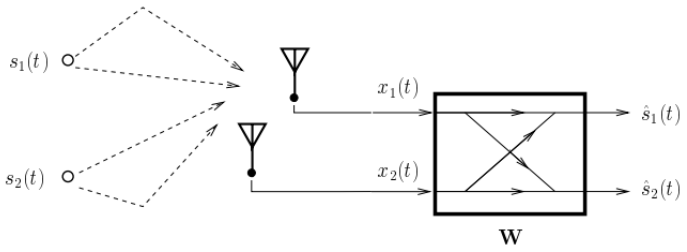
$$\mathbf{x}(t) = \mathbf{a}(\theta_1)\beta_1 s_1(t) + \mathbf{a}(\theta_2)\beta_2 s_2(t) = [\mathbf{a}(\theta_1) \quad \mathbf{a}(\theta_2)] \begin{bmatrix} \beta_1 & \\ & \beta_2 \end{bmatrix} \begin{bmatrix} s_1(t) \\ s_2(t) \end{bmatrix}$$

- For the same source with multipath, the received signal will be:



$$\begin{aligned} \mathbf{x}(t) &= \mathbf{a}(\theta_1)\beta_1 s(t) + \mathbf{a}(\theta_2)\beta_2 s(t) \\ &= \{\beta_1 \mathbf{a}(\theta_1) + \beta_2 \mathbf{a}(\theta_2)\} s(t) \quad = \mathbf{a} s(t) \end{aligned}$$

Now, accounting all together, we will have something like:



Mathematically: $\mathbf{x}(t) = \mathbf{a_1}s_1(t) + \mathbf{a_2}s_2(t) = A\mathbf{s}(t)$, where $\mathbf{s}(t)$ is a column vector with every source and A is the matrix that accounts for how the signals will be received.

Thus, converging to the simple following formula. If we take N samples, then we get the expression below.

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$$

$$\mathbf{X} = [\mathbf{x}(0), \cdots, \mathbf{x}(N-1)] \text{ and } \mathbf{S} = [\mathbf{s}(0), \cdots, \mathbf{s}(N-1)]$$

$$\mathbf{X} = \mathbf{A}\mathbf{S}$$

As we'll see, the matched filter is not the best estimator because it only works perfectly if the is orthogonality between the symbols. If there isn't, multiplying the transpose won't give identity. A better way of processing would be to multiply the inverse and this is what is called a Zero Forcing estimator.

2.1 Matched Filter vs Zero Forcing vs Wiener Filter

One is said to be perfect for just noise, the other perfect for just ISI and the last seems to find the best result of both. The expressions first:

(check these properly)

$$W^H = A^H \tag{2}$$
$$W^H = H^\dagger \tag{3}$$
$$W = R_X^{-1}H = (HH^H + \sigma^2 I)^{-1}\mathbf{H} \tag{4}$$

The matched filter can deliver results as good as the Zero Forcing if there is no noise. Without noise, the symbols should still be orthogonal and the inverse multiplied by the matrix that is not inverted will give the identity and so will the transposed only because of the orthogonality.

-c pseudo inverse can make the noise explode, that is why the ZF can make the noise worse. This is related with the condition number. If we simply take the hermitian, the noise can never explode because there is no

risk of any element be close to infinity. This is why MF minimises noise and ZF the interference: ZF focus on really inverting the matrix that will return the identity and make the symbols not interfere with each other by successfully undoing the mixing the channel has just done.

The wiener receiver accounts the noise as well. And tends to one of the other two receivers if the noise or interference are much bigger than the other. If there is too much noise, the wiener optimises it and becomes very close to the MF so that we can have the best compromise between optimisations.

3 Important math

3.1 Norms

Basically, there are two or three norms that we should keep in mind. One that applies to vectors and two that apply to matrices.

Let $\mathbf{x} \in \mathbb{C}^N$ be an N-dimensional complex vector.

The Euclidean norm (2-norm) of $\mathbf{x}$ is

$$\|\mathbf{x}\| := \left(\sum_{i=1}^{N} |x_i|^2\right)^{1/2} = \left(\sum_{i=1}^{N} \bar{x}_i x_i\right)^{1/2} = (\mathbf{x}^{\mathrm{H}}\mathbf{x})^{1/2}$$

Let $\mathbf{A} \in \mathbb{C}^{M \times N}$ be an $M \times N$ complex matrix.

The *induced matrix 2-norm* (spectral norm, operator norm) is

$$\|\mathbf{A}\| := \max_{\mathbf{x}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \quad \text{or} \quad \|\mathbf{A}\|^2 = \max_{\mathbf{x}} \frac{\mathbf{x}^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{A}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}}$$

The *Frobenius norm* of $\mathbf{A}$ is

$$\|\mathbf{A}\|_{\mathrm{F}} = \left(\sum_{i=1}^{M}\sum_{j=1}^{N} |a_{ij}|^2\right)^{1/2}$$

3.2 Means and Variances

The mean is very self-explanatory.

The variance is measure of how much and how frequently the signal varies from the mean:

Finish the review of the final assignment

learn about each

Population variance [edit]

In general, the ***population variance*** of a *finite* population of size N with values x_i is given by

$$\begin{aligned}\sigma^2 &= \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x_i^2-2\mu x_i+\mu^2\right)\\ &= \left(\frac{1}{N}\sum_{i=1}^{N}x_i^2\right)-2\mu\left(\frac{1}{N}\sum_{i=1}^{N}x_i\right)+\mu^2\\ &= \left(\frac{1}{N}\sum_{i=1}^{N}x_i^2\right)-\mu^2\end{aligned}$$

where the population mean is

$$\mu = \frac{1}{N}\sum_{i=1}^{N}x_i.$$

And some important properties about it:

$$Var(X) \geq 0 \tag{5}$$
$$P(X=a) = 1 \Rightarrow Var(X) = 0 \tag{6}$$
$$Var(X+a) = Var(X) \tag{7}$$
$$Var(aX) = a^2 Var(X) \tag{8}$$

Consider the following example. We have a vector of measurements with 100 elements and 50 of those elements have the value 100 and the other 50 have the value 200. From applying the variance we we'll see that the variance of this set of measurements is 50^2, so the standard deviation which is the square root of the variance, will be 50.

What does this mean? Well, the standard deviation is exactly that. However, what is standard is not what is expected. The expected is the mean or the average and they happen to coincide in this fortunate example.

Another question: why do we use the standard deviation and not the average deviation? Well, they give different results! The major difference between them is that using $\sqrt{E[(X-\mu^2)]}$ instead of $E[|X-\mu|]$ will give

an higher importance to bigger deviation while the latter will give the same important to all. If we think about it: it should be much more likely to be a small deviation, it should be more likely to be close to the mean. Therefore, there should be an importance scaling when we see a bigger deviation because it should be hard to come by so if it appeared, that should be considered. One can see it as a Pythagorean Theorem of statistics.

3.2.1 Variance = Power? SNR(σ)

Yes, if the signal has null mean.

The power of a signal can be obtained from: $P(X) = \frac{1}{N}\sum_i = 1^N x_i^2$, the total energy divided by the time intervals/number of samples.

Thus, to include noise in a signal such that it has a certain SNR, we need to calculate the signal power at the reception, lets called it S' since S was the original transmitted signal. Then: $SNR = \frac{S'}{P_N} = \frac{S'}{\sigma_N}$. And don't forget the SNR is in linear units. To obtain the variance the formula ust has to be inverted accordingly.

3.3 Unitary Matrix, Isometry and Projections

Lacking the time to properly explain, the slides do a excellent job summing it up:

Unitary matrix

- A square matrix $\mathbf{U}$ is called *unitary* if $\mathbf{U}^{\mathrm{H}}\mathbf{U} = \mathbf{I}$ and $\mathbf{U}\mathbf{U}^{\mathrm{H}} = \mathbf{I}$.
- Properties:
  - A unitary matrix looks like a rotation and/or a reflection.
  - Its norm is $\|\mathbf{U}\| = 1$.
  - Its columns and rows are orthonormal.

Isometry

- A tall rectangular matrix $\hat{\mathbf{U}}$ is called an isometry if $\hat{\mathbf{U}}^{\mathrm{H}}\hat{\mathbf{U}} = \mathbf{I}$.
  - Its columns are orthonormal basis of a subspace (not the complete space).
  - Its norm is $\|\hat{\mathbf{U}}\| = 1$.
  - There is an orthogonal complement $\hat{\mathbf{U}}^{\perp}$ of $\hat{\mathbf{U}}$ such that $\mathbf{U} = [\hat{\mathbf{U}}\ \hat{\mathbf{U}}^{\perp}]$ is unitary.

Projection

- A square matrix $\mathbf{P}$ is a projection if $\mathbf{PP} = \mathbf{P}$.
- It is an orthogonal projection if also $\mathbf{P}^{\mathrm{H}} = \mathbf{P}$.
  - The norm of an orthogonal projection is $\|\mathbf{P}\| = 1$.
  - For an isometry $\hat{\mathbf{U}}$, the matrix $\mathbf{P} = \hat{\mathbf{U}}\hat{\mathbf{U}}^{\mathrm{H}}$ is an orthogonal projection (onto the space spanned by the columns of $\hat{\mathbf{U}}$). This is the general form of a projection.
- Suppose $\mathbf{U} = [\underbrace{\hat{\mathbf{U}}}_{d} \quad \underbrace{\hat{\mathbf{U}}^{\perp}}_{M-d}]$ is unitary. Then, from $\mathbf{U}\mathbf{U}^{\mathrm{H}} = \mathbf{I}_M$:

$$\hat{\mathbf{U}}\hat{\mathbf{U}}^{\mathrm{H}} + \hat{\mathbf{U}}^{\perp}(\hat{\mathbf{U}}^{\perp})^{\mathrm{H}} = \mathbf{I}_M\,, \qquad \hat{\mathbf{U}}\hat{\mathbf{U}}^{\mathrm{H}} = \mathbf{P}\,, \qquad \hat{\mathbf{U}}^{\perp}(\hat{\mathbf{U}}^{\perp})^{\mathrm{H}} = \mathbf{P}^{\perp} = \mathbf{I}_M - \mathbf{P}$$

- Any vector $\mathbf{x} \in \mathbb{C}^M$ can be decomposed into $\mathbf{x} = \hat{\mathbf{x}} + \hat{\mathbf{x}}^{\perp}$, where $\hat{\mathbf{x}} \perp \hat{\mathbf{x}}^{\perp}$,

$$\hat{\mathbf{x}} = \mathbf{P}\mathbf{x} \in \mathrm{ran}(\hat{\mathbf{U}})\,, \qquad \hat{\mathbf{x}}^{\perp} = \mathbf{P}^{\perp}\mathbf{x} \in \mathrm{ran}(\hat{\mathbf{U}}^{\perp})$$

Projection onto the column span of A

- Suppose $\mathbf{A}$ is tall and $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ is invertible. Then

$$\mathbf{P}_{\mathbf{A}} := \mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}, \qquad \mathbf{P}_{\mathbf{A}}^{\perp} := \mathbf{I} - \mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}$$

  are orthogonal projections, onto the range of $\mathbf{A}$ and kernel of $\mathbf{A}^{\mathrm{H}}$, resp.

- Proof:

  Verify that $\mathbf{PP} = \mathbf{P}$ and $\mathbf{P}^{\mathrm{H}} = \mathbf{P}$, hence $\mathbf{P}$ is an orthogonal projection.

  If $\mathbf{b} \in \mathrm{ran}(\mathbf{A})$, then $\mathbf{b} = \mathbf{A}\mathbf{x}$ for some $\mathbf{x}$.

  Hence

$$\mathbf{P}_{\mathbf{A}}\mathbf{b} = \mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{A}\mathbf{x} = \mathbf{b}$$

  so that $\mathbf{b}$ is invariant under $\mathbf{P}_{\mathbf{A}}$.

  If $\mathbf{b} \perp \mathrm{ran}(\mathbf{A})$, then $\mathbf{b} \in \ker(\mathbf{A}^{\mathrm{H}})$, or $\mathbf{A}^{\mathrm{H}}\mathbf{b} = \mathbf{0}$. Hence $\mathbf{P}_{\mathbf{A}}\mathbf{b} = \mathbf{0}$.

- Given $\mathbf{A}$, $\mathbf{b}$, find

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \| \mathbf{A}\mathbf{x} - \mathbf{b} \|^2$$

- Solution:

  Write $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$, where $\mathbf{b}_1 \in \text{ran}(\mathbf{A})$, $\mathbf{b}_2 \perp \text{ran}(\mathbf{A})$.

  Then

$$\mathbf{b}_1 = \mathbf{P}_{\mathbf{A}}\mathbf{b} = \mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{b}$$

$$\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{A}\left\{\mathbf{x} - (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{b}\right\} - \mathbf{b}_2$$

  Note that the two terms are orthogonal. Thus

$$\| \mathbf{A}\mathbf{x} - \mathbf{b} \|^2 = \| \mathbf{A}\left\{\mathbf{x} - (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{b}\right\} \|^2 + \| \mathbf{b}_2 \|^2$$

  To minimize the error, set $\hat{\mathbf{x}} = (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{b}$.

3.4 Singular Value Decomposition

MERGE

Once again the slides sum this up perfectly and as such a direct use will be made:

- For any matrix $\mathbf{X}$, there is a decomposition

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}$$

  Here, $\mathbf{U}$ and $\mathbf{V}$ are unitary, and $\mathbf{\Sigma}$ is diagonal, with positive real entries.

- Properties:
  - The columns $\mathbf{u}_i$ of $\mathbf{U}$ are called the left singular vectors.
  - The columns $\mathbf{v}_i$ of $\mathbf{V}$ are called the right singular vectors.
  - The diagonal entries σ_i of $\mathbf{\Sigma}$ are called the singular values.
  - They are positive and real, and usually sorted such that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$$

- More specifically, for an $M \times N$ tall ($M \geq N$) matrix $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}} = [\hat{\mathbf{U}} \quad \hat{\mathbf{U}}^{\perp}] \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \sigma_d & & \\ \hline & & 0 & \\ & & & 0 \\ \hline 0 & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & 0 \end{array}\right] \begin{bmatrix} \hat{\mathbf{V}}^{\mathrm{H}} \\ (\hat{\mathbf{V}}^{\perp})^{\mathrm{H}} \end{bmatrix}$$

$$\mathbf{U} : M \times M, \quad \mathbf{\Sigma} : M \times N, \quad \mathbf{V} : N \times N$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > \sigma_{d+1} = \cdots = \sigma_N = 0$$

- *'Economy size' SVD:* $\mathbf{X} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^{\mathrm{H}}$, where $\hat{\mathbf{\Sigma}} : d \times d$, containing $\sigma_1, \cdots, \sigma_d$.

More exactly regarding the end, $\hat{\mathbf{U}}$ will have d column vectors in it and $\hat{\mathbf{V}}^H$ will have d row vectors in it.

Now some facts about SVD will introduce perfectly the next section where they are carefully explained.

Some SVD facts

- The rank of $\mathbf{X}$ is d, the number of nonzero singular values.

- $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{H}} \Leftrightarrow \mathbf{X}^{\mathrm{H}} = \mathbf{V}\Sigma\mathbf{U}^{\mathrm{H}} \Leftrightarrow \mathbf{X}\mathbf{V} = \mathbf{U}\Sigma \Leftrightarrow \mathbf{X}^{\mathrm{H}}\mathbf{U} = \mathbf{V}\Sigma$

  $\Rightarrow$ The columns of $\hat{\mathbf{U}}$ ($\hat{\mathbf{U}}^{\perp}$) are orthonormal basis for range of $\mathbf{X}$ (kernel of $\mathbf{X}^{\mathrm{H}}$).

  $\Rightarrow$ The columns of $\hat{\mathbf{V}}$ ($\hat{\mathbf{V}}^{\perp}$) are orthonormal basis for range of $\mathbf{X}^{\mathrm{H}}$ (kernel of $\mathbf{X}$).

- The norm of $\mathbf{X}$ or $\mathbf{X}^{\mathrm{H}}$ is $\|\mathbf{X}\| = \|\mathbf{X}^{\mathrm{H}}\| = \sigma_1$, the largest singular value.

  The norm is attained on the corresponding singular vectors $\mathbf{u}_1$ and $\mathbf{v}_1$:

$$\mathbf{X}\mathbf{v}_1 = \mathbf{u}_1\sigma_1 \qquad \mathbf{X}^{\mathrm{H}}\mathbf{u}_1 = \mathbf{v}_1\sigma_1$$

explain

3.5 Matrices Perspectives and Spaces Relations

The facts that should have been particularly hard to swallow where the ones related with the basis. These happen to be the easiest to explain!

In the intro of a matrix, it is explained the following:

$$\begin{pmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 6 \\ 7 \end{pmatrix} x + \begin{pmatrix} 2 \\ 5 \\ 8 \end{pmatrix} y + \begin{pmatrix} 3 \\ 4 \\ 9 \end{pmatrix} z \tag{9}$$

Therefore it becomes clear that the column space of the result will be a subset of the column space of the first matrix, because the column of the result will be a linear combination of those the columns of the first matrix. For bigger matrices being multiplied on the right, we'll have no more than more columns on the right of the result column.

In fact, the 3 matrix perspectives are:

- the first column of the result will be the a linear combination of the columns of B by the entrances along the first column C, exactly like above;
- for A = BC : from the above we see that only the first entrances of the columns of B matter for the first entrance of the resulting column,

therefore we see that multiplying the first (or the i-th) row of B with the first (or j-th) column of C we get the entrance (1,1) of A (or (i,j) $\rightarrow A_{ij}$);

- slightly harder to notice is now:

$$\begin{pmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ & & \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix} 1 + \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} 2 + \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} 3 = \begin{pmatrix} 6 & 12 & 18 \\ & & \end{pmatrix}$$

The rows of A will be the linear combination of the rows of C.

And these 3 perspectives are enough to relate various spaces of each matrix:

- The column span of A is a subset of the column span of B;
- The row span and the row kernel of A is a subset of the row span and the kernel of C

But let us take one other approach: Dimensions

If matrix has full rank, it has a rank that is equal to the smallest dimension.

Having in mind that A belongs in $\Re^{m \times n}$, being m the number of rows and n the number of columns, the following four spaces are going to exist if A is rank deficient.

- Column space of A - $C(A)$
- Row space of A - $C(A^{\mathrm{T}})$
- Nullspace of A - $N(A)$
- Left Nullspace of A - $N(A^{\mathrm{T}})$

Note that if A is rank deficient, even if m = n, there will be rows and columns without pivots when we put the matrix in reduced row echelon form. The "free variables", the variables without pivots in them, will belong to the nullspaces such that:

- if $r < m$, the row nullspace, or simply nullspace, will have dimension $m - r$
- if $r < n$, the column nullspace, or the left nullspace, will have dimension $n - r$

This is more formally known as:

The Fundamental Theorem of Algebra .

Give the relation presented before, one may go further and explain these:

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{H}} \quad \Leftrightarrow \quad \mathbf{X}^{\mathrm{H}} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{H}} \quad \Leftrightarrow \quad \mathbf{X}\mathbf{V} = \mathbf{U}\boldsymbol{\Sigma} \quad \Leftrightarrow \quad \mathbf{X}^{\mathrm{H}}\mathbf{U} = \mathbf{V}\boldsymbol{\Sigma}$$

$\Rightarrow$ The columns of $\hat{\mathbf{U}}$ ($\hat{\mathbf{U}}^{\perp}$) are orthonormal basis for range of $\mathbf{X}$ (kernel of $\mathbf{X}^{\mathrm{H}}$).

$\Rightarrow$ The columns of $\hat{\mathbf{V}}$ ($\hat{\mathbf{V}}^{\perp}$) are orthonormal basis for range of $\mathbf{X}^{\mathrm{H}}$ (kernel of $\mathbf{X}$).

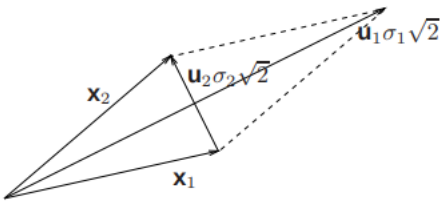
Since we can see any pair of matrices as only one:

- Because $\mathbf{U}$ and $\mathbf{V}$ are unitary, if they form a basis of a space, that basis will be a orthonormal basis.
- Given that we may write $\mathbf{X} = \hat{\mathbf{U}}\hat{\boldsymbol{\Sigma}}\hat{\mathbf{V}}^{\mathrm{H}}$, from what was mentioned before we can see that $\hat{\mathbf{U}}$ columns span the space of the columns of X like $\hat{\mathbf{V}}^{\mathrm{H}}$ spans the space of the rows of X, or equivalently, $\hat{\mathbf{V}}$ is an orthonormal basis for the space of the columns of $\mathbf{X}^{\mathrm{H}}$.
- Since $\hat{\mathbf{U}}^{\perp}$ is orthogonal to $\hat{\mathbf{U}}$, it will span an orthogonal space, and the space orthogonal to the columns of $\mathbf{X}$ is the kernel of $\mathbf{X}^{\mathrm{H}}$, which is the left nullspace or the nullspace of the columns of $\hat{\mathbf{U}}$. Similar with $\hat{\mathbf{V}}$

3.6 SVD geometrical Interpretation

Only for curiosity purposes:

Geometrical interpretation



Construction of the left singular vectors and values of the matrix $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2]$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ have *equal length*.

- The largest singular vector $\mathbf{u}_1$ is in the direction of the sum of $\mathbf{x}_1$ and $\mathbf{x}_2$: the 'common' direction of the two vectors.
  Singular value: $\sigma_1 = \|\mathbf{x}_1 + \mathbf{x}_2\|/\sqrt{2}$.
- The smallest singular vector $\mathbf{u}_2$ depends on the difference $\mathbf{x}_2 - \mathbf{x}_1$.
  Singular value: $\sigma_2 = \|\mathbf{x}_2 - \mathbf{x}_1\|/\sqrt{2}$.

One can see that the singular vectors are scaled by the singular values, that they are orthogonal to each other and that the singular vectors will span the same space as the vectors in X. Therefore, if there is a linear dependent vector in X, the number of singular vectors will continue the same, as many as the rank of X. The zeros in the central matrix is what allows the "addition" of the orthogonal matrices, because they will be multiplied to 0. The *economy size* SVD provides only left and right singular vectors such that there are non-zero singular values in the middle matrix.

3.7 Pseudo-Inverse

An extremely important topic: can you invert a non-square matrix? Well, obviously no, but you can do your best approximation of it.

The above sections need a slightly further explanation, but in essence, the idea is fully explained in the following 3 slides:

Full rank pseudo-inverse

- $\mathbf{X} : M \times N$, tall ($M \geq N$), full rank.

  The *pseudo-inverse* of $\mathbf{X}$ is $\mathbf{X}^\dagger = (\mathbf{X}^{\mathrm{H}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{H}}$.

- It satisfies $\mathbf{X}^\dagger\mathbf{X} = \mathbf{I}_N$ (i.e., $\mathbf{X}^\dagger$ is an inverse on the "short space").

- Also, $\mathbf{X}\mathbf{X}^\dagger = \mathbf{P}$: a projection onto the column span of $\mathbf{X}$.

Rank-deficient pseudo-inverse

- $\mathbf{X} : M \times N$, tall ($M \geq N$), rank-d, with 'economy size' SVD $\mathbf{X} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^{\mathrm{H}}$.

  The pseudo-inverse of $\mathbf{X}$ is $\mathbf{X}^\dagger = \hat{\mathbf{V}}\hat{\mathbf{\Sigma}}^{-1}\hat{\mathbf{U}}^{\mathrm{H}}$.

- It satisfies $\mathbf{X}\mathbf{X}^\dagger = \hat{\mathbf{U}}\hat{\mathbf{U}}^{\mathrm{H}} = \mathbf{P}_c$, $\quad \mathbf{X}^\dagger\mathbf{X} = \hat{\mathbf{V}}\hat{\mathbf{V}}^{\mathrm{H}} = \mathbf{P}_r$.

  - The norm of $\mathbf{X}^\dagger$ is $\|\mathbf{X}^\dagger\| = \sigma_d^{-1}$.
  - The *condition number* of $\mathbf{X}$ is $c(\mathbf{X}) := \frac{\sigma_1}{\sigma_d}$.

    If it is large, then $\mathbf{X}$ is hard to invert ($\mathbf{X}^\dagger$ is sensitive to small changes).

Interpretation of condition number

- The condition number gives the relative sensitivity of the solution of linear systems of equations.

- Illustration:

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} &\Rightarrow \quad \mathbf{x} &= \mathbf{A}^{-1}\mathbf{b} \\ \mathbf{b}^1 &= \mathbf{b} + \mathbf{e} &\Rightarrow \quad \mathbf{x}^1 &= \mathbf{x} + \mathbf{A}^{-1}\mathbf{e} \end{aligned}$$

Define $\sigma_1 = \|\mathbf{A}\|$, $\quad \sigma_N^{-1} = \|\mathbf{A}^{-1}\|$. $\quad$ Use $\|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|$.

Then

$$\begin{aligned} \|\mathbf{A}^{-1}\mathbf{e}\| &\leq \sigma_N^{-1}\|\mathbf{e}\| \\ \|\mathbf{b}\| &\leq \sigma_1\|\mathbf{x}\| \\ \frac{\|\mathbf{x}^1 - \mathbf{x}\|}{\|\mathbf{x}\|} &\leq \sigma_N^{-1}\frac{\|\mathbf{e}\|}{\|\mathbf{x}\|} \leq \sigma_N^{-1}\sigma_1\frac{\|\mathbf{e}\|}{\|\mathbf{b}\|} \end{aligned}$$

Rank approximation

- $\mathbf{X} : M \times N$, with SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}$.
- To improve the condition number of $\mathbf{X}$, we can set the small σ_i equal to zero.

  This leads to a low rank approximation of $\mathbf{X}$.
- Illustration:
  - Choose a threshold ϵ, and suppose d singular values are larger than ϵ.
  - $\hat{\mathbf{U}}$: first d columns of $\mathbf{U}$, $\hat{\mathbf{V}}$: first d columns of $\mathbf{V}$, $\hat{\mathbf{\Sigma}}$: top-left $d \times d$ block of $\mathbf{\Sigma}$.
  - Then $\hat{\mathbf{X}} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^{\mathrm{H}}$ is a rank-d approximant of $\mathbf{X}$, with error

$$\begin{aligned} \|\mathbf{X} - \hat{\mathbf{X}}\| &= \sigma_{d+1} \\ \|\mathbf{X} - \hat{\mathbf{X}}\|_{\mathrm{F}}^2 &= \sigma_{d+1}^2 + \cdots + \sigma_N^2 \end{aligned}$$

3.8 Short and Important considerations

- The maximum rank is the smallest of the dimensions: if it has more unknowns than equations, the maximum rank is the number of equations. If it has more equations than unknowns, the maximum rank is the number of unknowns;
- The Toeplitz structure is having the diagonal entrances, from left to right, top to bottom, equal. Like:

$$\begin{bmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{bmatrix}.$$

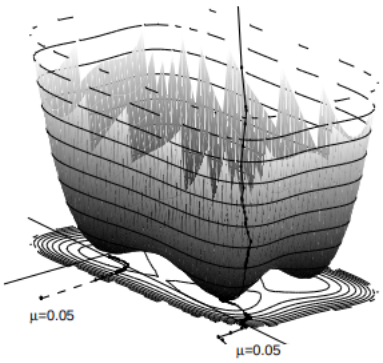
4 Constant Modulus Algorithm

Exploring the characteristic that many of our transmitted signals have constant modulus (FM, PH, FSK, PSK...), we can use this property in the estimation, by looking of a weights vector that leads to estimations of modulus 1.

$$y_k = \mathbf{w}^{\mathrm{H}}\mathbf{x}_k = \hat{s}_k$$

such that $|y_k| = 1$ for all k.

- Cost function for 2 real sources and 2 antennas:

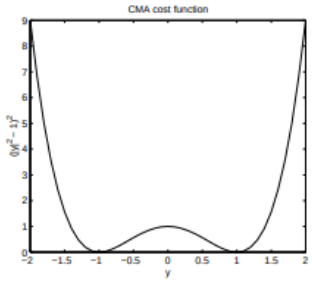


As should be expected, the cost function we want to minimise, may very well not be trivial to visualize.

- Possible optimization problem:

$$\min_{\mathbf{w}} J(\mathbf{w}) \quad \text{where} \quad J(\mathbf{w}) = \mathrm{E}\left[(|y_k|^2 - 1)^2\right]$$

- The CMA cost function as a function of y (for simplicity, y is taken real here):



- There is no unique minimum:

  if $y_k = \mathbf{w}^{\mathrm{H}}\mathbf{x}_k$ is CM, then another beamformer is $\alpha\mathbf{w}$, for any scalar $|\alpha| = 1$

A cost function can simply be the distance of the modulus of our prediction y_k to 1. Since we want to minimise that cost function, we will take little steps in the direction of the minimum. By definition, cost functions we tend to want to minimise. μ will be the step size and multiplied with the gradient, we get closer to one of the minimums. Which minimum we get closer to will depend on our step size but mainly on our initial beamformer.

It will be it stochastic gradient-descent techniques the minimisation will be performed. From the way our cost function is set, we want to make it 0. $J(\mathbf{w}) = 0$ will reconstruct the sources if we apply that beamformer to them. We can only manage that without noise.

The trick will be to update the beamformer contrary to the direction of the gradient. The gradient "points" the direction of the biggest increase, thus given a step in the opposite direction we get the biggest decrease.

- Cost function:

$$J(\mathbf{w}) = \mathrm{E}\left[(|y_k|^2 - 1)^2\right], \qquad y_k = \mathbf{w}^{\mathrm{H}}\mathbf{x}_k$$

- Stochastic gradient method:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \nabla J(\mathbf{w}_k), \qquad \mu > 0 \text{ is the step size}$$

- Computation of gradient (use $|y_k|^2 = y_k \bar{y}_k = \mathbf{w}^{\mathrm{H}}\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}\mathbf{w}$):

$$\begin{aligned} \nabla J(\mathbf{w}) &= 2\mathrm{E}\{(|y_k|^2 - 1) \cdot \nabla(\mathbf{w}^{\mathrm{H}}\mathbf{x}_k\mathbf{x}_k^{\mathrm{H}}\mathbf{w})\} \\ &= 2\mathrm{E}\left\{(|y_k|^2 - 1) \cdot \mathbf{x}_k\mathbf{x}_k^{\mathrm{H}}\mathbf{w}\right\} \\ &= 2\mathrm{E}\left\{(|y_k|^2 - 1)\, \bar{y}_k \mathbf{x}_k\right\} \end{aligned}$$

- Replace expectation by instantaneous value and absorbe the factor 2 in μ:

CMA(2,2):
$$\begin{cases} y_k &= \mathbf{w}^{(k)\mathrm{H}}\mathbf{x}_k \\ \mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} - \mu \mathbf{x}_k (|y_k|^2 - 1)\bar{y}_k \end{cases}$$

- Similar to LMS, but with update error $(|y_k|^2 - 1)y_k$.

Note that because we want to minimise the error function, we get a steepest-descent algorithm. If we wanted to maximise and we would take little steps in the direction of the maximum thus having a steepest-ascent algorithm.

There are however some constraints that come along with all advantages of this algorithm for the construction of a beamformer:

- Advantages
  - The algorithm is extremely simple to implement
  - Adaptive tracking of sources
  - Converges to minima close to the Wiener beamformers (for each source)

- Disadvantages
  - Noisy and slow
  - Step size μ should be small, else instable
  - Only one source is recovered (which one?)
  - Possible misconvergence to local minimum (with finite data)

The main disadvantage is that the beamformer will be return symbols a scalar away from the correct ones. If we are transmitting QPSK, the estimated symbols should be the same distance in an arc apart, however, because the algorithm only optimises for modulus 1, the phase can turn around quite a bit.

One important tradeoff to have in mind is that a smaller the step size will need more iterations to converge to a modulus 1 estimation but will result in estimation with a smaller variance around the true value.

5 CDMA

Code Division Multiple Access is a way of multiplexing several users into the same channel without having their signals colliding and interfering with each other.

As already known about this technology, it uses orthogonal codes multiplied to the current signal to generate the signal to transmit. These codes are called spreading codes or chip sequences. They have a frequency much higher than the data sequence. As a matter of fact, for each bit of data, there are P pulses of the code, i.e $PT_c = Ts$ where T_c is the period of a single chip and T_s is the period of a symbol.

Because these spreading codes have such an high frequency, what they will effectively do when multiplied to the signal is shifting a copy of the code multiplied with the signal in the frequency domain to a frequency that is the

sum of the two. And of course.... Maybe we should do a quick review on convolutions:

5.1 Review: Convolution

By definition, a convolution between two signals is presented in (10).

$$f(t) * g(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau \tag{10}$$

By far, the best way of getting a notion of the convolution is memorising the steps it takes to compute one have a good visual notion of it.

For a visual insight, have a look at wikipedia Convolution . Below, in figure 1 there will be an offline copy of the above with the most important parts.

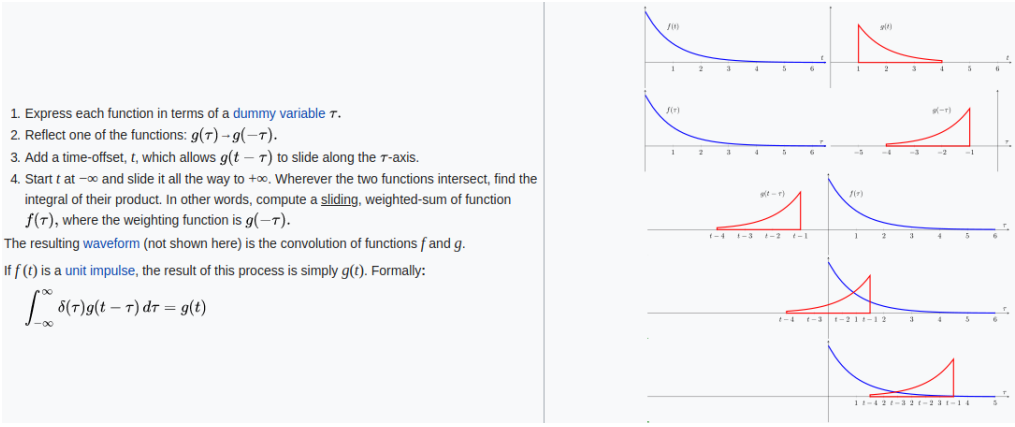


Figure 1: Visual explanation of a Convolution taken from Wikipedia

After this, it should be easier to think why a multiplication of cosines in time should give a signal shifted in frequency.

Getting back to CDMA, the multiplication of the sequence of bits from the data should give something much more spread in frequency. Actually, it will be somewhat of a mess with a center at DC (still not considering the multiplication by the carrier) and the highest frequency being the sum of

the frequencies of the two signals. In conclusion, in terms of bandwidth the signal will know occupy the biggest bandwidth of the two signals plus the bandwidth of the small signal to each side, equating to $f_c + 2 * f_s$. And let's not address that a rectangle in time is a sinc in frequency and that the bandwidths are not defined that clearly...

5.2 Perspective: Spread Spectrum in 3G

As seen in Cellular Networks: Radio Access Networks course, 3G uses Spread Spectrum techniques or CDMA to multiplex the user resources utilisation. The final chip rate must be 3.84 chips per second, so based on the required transmission bitrate, the spreading code used can have very few chips or many chips. The amount of chips per symbol to transmit is the Spreading Factor (SF) in 3G nomenclature or Processing Gain (PG) of the spreading code. The analysis is usually never done as bandwidth occupied but much more in terms of what spreading code multiplied to a bit rate to results in 3.84 Mchips/s.

In terms of bandwidth, all CDMA signals should fall into a 5 MHz bandwidth in 3G, therefore whatever math follows, you know that the bandwidth can't exceed these 5 MHz. Indeed, from a maximum symbol rate of 192 bits/s , half of 384 bits/s due to QPSK modulation, using a code rate of 0.41, one reaches a final channel rate of ≈ 469 bits/s. With this rate, it is used a spreading factor of 8, leading ..... where do we take the bandwidth from here??

Again, getting back to the point: the signal will be spread in frequency, seeming almost noise.

Considering the simplest case where we just received the signal transmitted, sampling in each chip period, we will receive:

$$\mathbf{x}[n] := \begin{bmatrix} x(nT_s) \\ x(nT_s + T_c) \\ \vdots \\ x(nT_s + (P-1)T_c \end{bmatrix} = \begin{bmatrix} c[0] \\ c[1] \\ \vdots \\ c[P-1] \end{bmatrix} s[n] = \mathbf{c}s[n]$$

Where $c[0]...c[P-1]$ will be the spreading code with length P, and is being multiplied to the current symbol $s[n]$. Additionally, for Q users using Q different orthogonal codes:

Q synchronized users with code vectors $\mathbf{c}_1, \cdots, \mathbf{c}_Q$:

$$\mathbf{x}[n] = \sum_{i=1}^{Q} \begin{bmatrix} c_i[0] \\ c_i[1] \\ \vdots \\ c_i[P-1] \end{bmatrix} s_i[n] = \sum_{i=1}^{Q} \mathbf{c}_i s_i[n]$$

Orthogonal codes:

$$\sum_{k=0}^{P-1} \bar{c}_i[k] c_j[k] = 0 \quad (i \neq j) \qquad \Leftrightarrow \qquad \mathbf{c}_i^{\mathrm{H}} \mathbf{c}_j = 0 \quad (i \neq j)$$

Note that orthogonal codes have an internal product of 0. This can be perceived as one not having got a projection in the other. Therefore, a good way of extracting a signal transmitted by user i is by doing internal products with the code that user uses with the received signal because the result will be only the signal that used that code.

$$y[n] = \mathbf{c}_i^H \mathbf{x}[n] = \sum_{j=1}^{Q} \mathbf{c}_i^H \mathbf{c}_j s_j[n] = P s_i[n] \tag{11}$$

And this is equivalent to the matched filter because we are simply undoing the process that took place before the transmission. Also, note that the signal after decoding will be $Ps_i[n]$. This is because the norm of a code should be unitary. Just by definition: the codes should only be +1 or -1, therefore doing $\frac{\sum_{i=1}^{P} |c_i|^2}{P} = 1$ and from here is clear why P times the norm will be P. The only little detail that is missing in case you didn't get it: $|c_i|^2 = c_i * \cdot c_i$ which will degenerate in the hermitian when we speak about vectors. A revision on the norms has been done in a previous chapter 3.1.

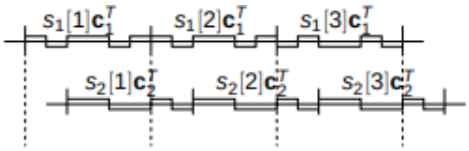
If the users have orthogonal codes and are perfectly synchronized, then there will be no Multi-user Access Interference (MAI). If there is noise and we apply the method above, we will get a signal with the signal of interest and with the noise after the internal product:

Signal of Interest	$Ps_i[n]$	variance P^2
Noise at output	$\sum_{k=0}^{P-1} \bar{c}_i[k] n(nT_s + kT_c)$,	variance $P\sigma^2$

Based on the Variance properties described in 3.2, is possible to confirm the variance of signal that has a constant multiplied to it, i.e the variance will be the constant squared multiplied with the variance of that signal. And in the lower part: the sum of the noise samples will have variance $P * \sigma^2$ because the samples are assumed independent the the variance of a sum of independent variables is the sum of the variances.

The last detail worth mentioning: here we can see the P processing gain due to the relations between variances and powers. The power is proportional to the variance - more exactly, the power is the variance multiplied to the length of the vector - therefore the processing gain will be evident.

Considering now that the users with different codes can be delayed. While wanting the symbol n, we would get interference from previous symbols and from the delays of current symbols.



Asynchronous users with delays $\{\tau_i\}$:

$$x(t) = \sum_{i=1}^{Q} \sum_{k=0}^{P-1} s_i[n] g_i(t - nT_s - \tau_i), \qquad g_i(t) = \sum_{k=0}^{P-1} c_i[k] p(t - kT_c)$$

Assume we synchronize user 1 ($\tau_1 = 0$) then

$$\mathbf{x}[n] = \begin{bmatrix} x(nT_s) \\ x(nT_s + T_c) \\ \vdots \\ x(nT_s + (P-1)T_c) \end{bmatrix} = \begin{bmatrix} c_1[0] \\ c_1[1] \\ \vdots \\ c_1[P-1] \end{bmatrix} s_1[n] + \begin{bmatrix} 0 \\ 0 \\ \hline c_2[0] \\ \vdots \end{bmatrix} s_2[n] + \begin{bmatrix} \vdots \\ c_2[P-1] \\ \hline 0 \\ 0 \end{bmatrix} s_2[n-1] + \cdots$$

Matched filter: $y[n] = \mathbf{c}_1^H \mathbf{x}[n] = P s_1[n] + \text{MAI}$

And now, if we try to receive a given signal with the matched filter, not considering the noise, we would get interference, more specifically MAI.

One way of fighting this was using codes that keep being orthogonal even if they are out of phase. Gold codes were invented for this purpose and they still are used in CDMA applications like GPS, so that different satellites can transmit at the same time and the phone distinguish between them.

However, these Gold codes are not miraculous and there will be some MAI still. Also, for these codes to work, Q must be much smaller than P, which is a strong constraint. If we had to the mix that so far the signals were assumed to be received with equal strength, one can see clearly that if the interference is received with a stronger signal it will cause stronger MAI. It becomes impossible to do any power control to safeguard this because fast fading will cause this either way and the power control would be too complicated to be implemented at a millisecond scale.

Thus, we should use a multi-user detector. Accounting for the delays of various codes, we can formulate the previous problem as follows and use one of the well know estimators:

$$\mathbf{x}[n] = [\mathbf{c}_1 \ \mathbf{c}_2^{\downarrow} \ \mathbf{c}_2^{\uparrow} \ \cdots] \begin{bmatrix} s_1[n] \\ s_2[n] \\ s_2[n-1] \\ \vdots \end{bmatrix} + \mathbf{n}[n] = \mathbf{C}\mathbf{s}[n] + \mathbf{n}[n]$$

- ZF: $\hat{\mathbf{s}}[n] = \mathbf{C}^{\dagger}\mathbf{x}[n] = (\mathbf{C}^{H}\mathbf{C})^{-1}\mathbf{C}^{H}\mathbf{x}[n]$
- MMSE: $\hat{\mathbf{s}}[n] = \mathbf{W}^{H}\mathbf{x}[n]$ where $\mathbf{W} = \mathbf{R}_x^{-1}\mathbf{C} = (\mathbf{C}\mathbf{C}^{H} + \sigma^2\mathbf{I})^{-1}\mathbf{C}$

  $\Rightarrow \quad s_1[n] = \mathbf{c}_1^{H}(\mathbf{C}\mathbf{C}^{H} + \sigma^2\mathbf{I})^{-1}\mathbf{x}[n]$

These two new estimators have the great advantage of not needing to meet $Q \ll P$, but rather $2Q - 1 \leq P$, for this specific example where 2 interferences per user are considered. More generally, this method just needs C to be tall or square (hence the equal), so the number of rows of C, which is P, must be equal or greater than than the number of columns (the unknowns) for the matrix to be square or tall, respectively.

One other note is that subsequent estimations are not connected while they should because the signal transmitted in a certain point in time may be interfering with some signals that arrived at the same or after that time. One estimator that does this is the Maximum Likelihood Sequence Estimator or the Viterbi Estimator. The Wikipedia has a page that explains this algorithm very well and there is plenty of code available online.

Furthermore, let us consider a multipath channel where is possible that even the code from the same user can be interfering with the transmission. Considering attenuations of α_i and delays of τ_i of each i-th path, one may write:

Discrete multipath model:

$$x(t) = \sum_{i=1}^{L} \sum_{k=0}^{P-1} s_1[n]\alpha_i g_1(t - nT_s - \tau_i) + \cdots$$

$$\mathbf{x}[n] := \begin{bmatrix} x(nT_s) \\ x(nT_s + T_c) \\ \vdots \\ \hline x((n+1)T_s) \\ x((n+1)T_s + T_c) \\ \vdots \end{bmatrix} = \underbrace{\begin{bmatrix} | & | & | \\ & \mathbf{C_1} & \\ | & | & | \end{bmatrix}}_{\mathbf{h_1}} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_L \end{bmatrix} s_1[n] \quad + [\cdots] s_1[n-1] + \cdots$$

Where the previous and current symbols are considered for the estimation of every symbol.

To receive with a matched filter, we would have to multiply the transposed channel but that won't wield correct results because the matrices C_1 multiplied won't give the identity in the first place. And also, the delays and complex path gains have to be estimated.

The alternative is using equal gain combining ($\alpha_i = 1$).

A short algorithm analysis:

- The **optimal** approach is to use MLSE, i.e a joint estimation of all users bits and channels.
  Downside : exponential complexity

- Linear multi-user receivers: invert joint channel like decorrelating (using zero-forcing) and MMSE receiver. **Downside** : The amount of required training symbols is linear with the number of users. Could be worse but it will need pilot training.

- To prevent noise and interference estimation with training symbols, one can use blind multi-user receivers that use the *code structure* instead of the training symbols. This is: instead of trying to estimate the channels, try to estimate the equalizer that optimises the estimation.

The blind approach is based on assuming discrete delays equal to multiples of our sampling intervals (that are multiples of the period of a single chip because we just want to sample all chips once, not more than that).

Assume $\tau_i = iT_c$ and use channel structure:

$$\mathbf{h}_1 = \begin{bmatrix} h(0) \\ h(T_c) \\ \vdots \\ h(LT_c) \end{bmatrix} = \begin{bmatrix} c_1[0] & & \\ \vdots & \ddots & \\ c_1[P-1] & & c_1[0] \\ & \ddots & \vdots \\ & & c_1[P-1] \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_L \end{bmatrix} = \mathbf{C}_1\mathbf{a}_1$$

The code matrix $\mathbf{C}_1$ is known, and tall.

Algorithm

– Find the column span $\hat{\mathbf{U}}$ of $[\mathbf{x}[0]\ \mathbf{x}[1]\ \cdots]$. It contains the vector $\mathbf{h}_1$.

– Find which vector in $\hat{\mathbf{U}}$ can be written as $\mathbf{C}_1\mathbf{a}_1$. This identifies $\mathbf{h}_1$.

– Find a suitable equalizer (MMSE).

Without noise, this algorithm finds all user channels *exactly*.

It is very worth to properly analyse the proposed algorithm.

First, the matrix C_1 is build, it has P + L rows and L columns where each column is equal to the code of length P and zeros before or after that code, depending on the column. The first column supposes no delay, so the first column only has zeros after. The second is one chip delayed so the first element of that column is 0, the next P are the code and then zeros again. And so on and so forth until we have L delays which is going to be the considered length of our channel meaning that we assume that we don't get any signal more delayed than L chips. We know how to construct $\mathbf{C}_1$ because we have the code c1 and the instructions above. We need to find the path gains.

Note that $x[n] = \mathbf{h}_1 s[n]$, if there is no noise, will find the users exactly because what is happening is nothing more than correlation with various different delays and scaled by the path gain. But we need to find the correct $\mathbf{h}_1$. For this, we find the column span $\hat{\mathbf{U}}$ of $\mathbf{x}$ because it will contain $\mathbf{h}_1$ for sure. This happens practically by definition! See 3.5.

6 OFDM

OFDM stands for Orthogonal Frequency Division Multiplexing. It is called like this because orthogonality in time and frequency are used to transmit data.

Techniques to undo the actions of the channel are called equalization techniques.

Because of Multipath, symbols apart from the one being currently transmitted will interfere with the current symbol being received. L is the channel length, meaning that L symbols (one current and L-1 past symbols) will have effect in the current symbol being received. One way of preventing this would be to add a guard interval such that there is no previous symbol to interfere with the current one. Thus, these equalization techniques come in handy.

To prevent ISI, zero padding is usually done between symbols. But this does nothing regarding how difficult is to invert a convolutive channel. However, there is a quick solution to this! Bring the signal to the Frequency Domain and Equalize there! This way, instead of inverting a convolution in time is possible to invert a function in frequency and perform point-wise multiplication/division as shown below:

$$\hat{X}(f) = G(f)H(f)X(f)\,, \qquad G(f) = H^{-1}(f)$$

There is a very important trick here! If a cyclic prefix is added to the symbols before sending them, the equalization can be done using a circular convolution instead of using a linear one. Moreover, circular convolving a signal is the same operation as simply calculating its DFT.

————**Proof that DFT = Circular Convolution** : ————

Statement : performing the DFT of two signals, multiply them and then performing the IDFT will have the same result as performing a circular convolution between them.

A **proof by example** first and then a more formal version.

Consider: x(n) = [1,2,3,4] and h(n) = [1,2,3].

Remembering that the DFT and IDFT can be computed like:

$$\text{DFT} = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N} \text{IDFT} = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{j2\pi nk/N} \tag{12}$$

Then: $X(k) = DFT(x(n)) = \sum_{n=0}^{3} x(n)e^{-j2\pi nk/4}$

And: $H(k) = DFT(\widetilde{h}(n)) = DFT([1,2,3,0]) = \sum_{n=0}^{3} \widetilde{h}(n)e^{-j2\pi nk/4}$

So: $Y(k) = H(k)X(k)$

Performing $y(n) = IDFT(Y(k)) = \frac{1}{4}\sum_{n=0}^{N-1} Y(k)e^{j2\pi nk/4}$

We notice that $y(n) = \widetilde{h} \otimes x(n)$.

In fact, one can do a circular convolution with a matrix, by organizing one of the vectors as follows:

$$\widetilde{h}(n) \otimes x(n) = \begin{bmatrix} h_1 & 0 & h_3 & h_2 \\ h_2 & h_1 & 0 & h_3 \\ h_3 & h_2 & h_1 & 0 \\ 0 & h_3 & h_2 & h_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{13}$$

With the values attributed to h(n) and x(n), one finds the following results:

$$\begin{aligned} X(k) &= [10 \ -2+2j \ -2 \ -2-2j] \\ H(k) &= [6 \ -2-2j \ 2 \ -2+2j] \\ Y(k) &= [60 \ 8 \ -4 \ 8] \\ y(n) &= [18 \ 16 \ 10 \ 16] \end{aligned}$$

And computing y(n) by the Fourier Transforms will lead to the same result as doing so with the matrix.

Now a more **formal proof** : consider the DFTs of x_1 and x_2, $X_3(k)$ is the DFT of their convolution, in other words, the product of their DFTs. We want to compute $x_3(n)$ and find out that it has the same expression as the circular convolution of x_1 and x_2. This is done with great precision in this video.

-**Proof that CP transforms Linear into Circular Convolution** : -

Let's start by the statement of each discrete convolution, the formulas for their computations:

The Linear Convolution:

- General formula:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m] \tag{14}$$

An example: [1234] $\circledast$ [123], you turn around the [568] and start multiplying and shifting.

The first entry is going to be: 5×1.
The second entry is going to be: $5 \times 2 + 6 \times 1$
The third entry: $5 \times 3 + 6 \times 2 + 7 \times 1$
Reaching the result: [5 16 34 52 45 28]
Note that the dimensions, if either vector is empty, will be: len(a) + len(b) - 1. In this case, 4 + 3 - 1 = 6

- Table method:
  Simply write one of the vectors on the top row and the other on the first column. Multiply the entrances. Draw diagonals and sum the entrances in each separation. That sum of each one will be each entrance of the result and we are doing the exact same thing as above.

	1	2	3	4
5	5	10	15	20
6	6	12	18	24
7	7	14	21	28

- The matrix method:

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 \\ 7 & 6 & 5 & 0 \\ 0 & 7 & 6 & 5 \\ 0 & 0 & 7 & 6 \\ 0 & 0 & 0 & 7 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 16 \\ 34 \\ 52 \\ 45 \\ 28 \end{bmatrix} \tag{15}$$

Note that the constructed matrix has len(a) + len(b) - 1 rows.

The Circular Convolution:

- General Formula:

$$x_1 \otimes x_2 = \frac{1}{N} \sum_{n=0}^{N-1} x_1(n) x_2((m-n)_N) \tag{16}$$

Where $(m-n)_N$ means modulo N of what is inside parentheses.

- The matrix method:

  By organizing one of the signals in the matrix before multiplying it to the other, we can emulate a circular convolution. This was shown in equation 13. Note that if one of the signals is shorter than the other it should be padded with zeros to make them the same size.

One can note that if we pad with zeros both signals until they have the length of the result of the linear convolution, circularly convolving them will result in the same as linearly convolving them without the padding. Actually, we just need to pad the one that we will use to build the matrix.

What we need to realise is that if we have N symbols to transfer, instead of zero padding them with L-1 zeros to achieve ISI, given that the channel length is L (meaning that only the current and previous L-1 samples matter to the current received one), we can fill that space with a cyclic prefix that will enable us to use a cyclic convolution at the reception as an alternative to attempt to invert a linear convolution that the channel naturally creates. This cyclic prefix will help us because we can make a circular convolution only with the symbols we are expecting to receive, not with the CP, and get the same result.

In matlab:

```
x = [1 2 3 4];
h = [1 2 3];
x_with_cp = [3 4 1 2 3 4];

received = conv(h, x_with_cp); % = [3 10 18
    16 10 16 17 12]

rec_circ = cconv([1 2 3], [1 2 3 4], 4) % = [18
   16 10 16]
```

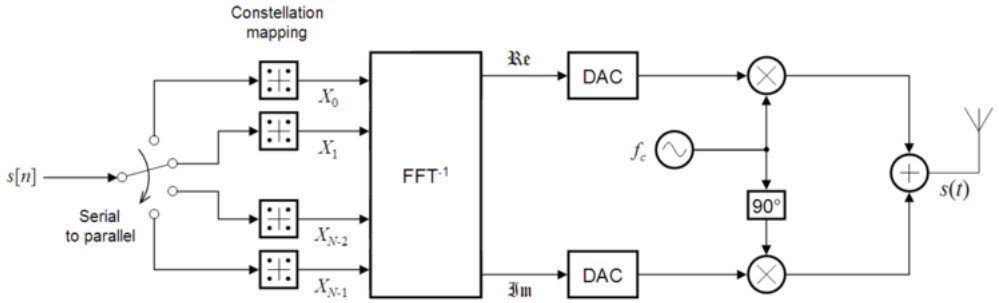
Thus we see that the middle terms are the same! Therefore, from the four transmitted symbols we can get estimations with 4 coefficients regarding the channel state.

Having accepted the previous proofs, we may proceed. Those were necessary to comprehend how the transmission and reception are conducted. The signal to send will be a superposition of all symbols to send in the different frequencies. More formally:

$$x_k = \sum s_i e^{j2\pi f_i k} \tag{17}$$

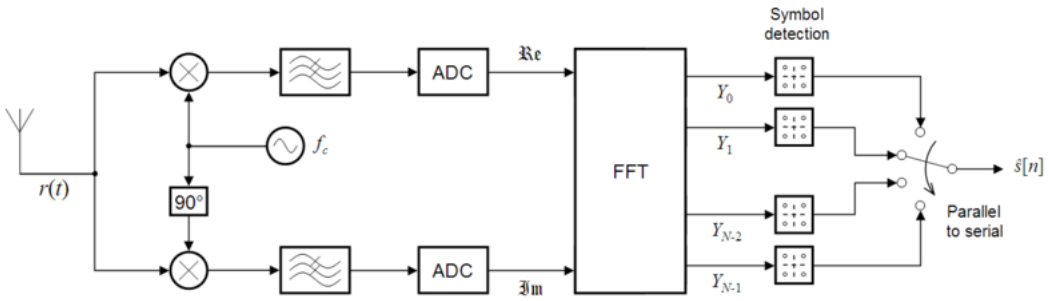
This is true because each source will be modulating one subcarrier of central frequency f_i hence the frequency shift of f_i.

The transmitter:



The symbols will be modulated in different carriers and an IDFT is used to do that modulation (basically multiplying a coefficient to that frequency and then using the IDFT to bring the signal back to time domain).

The receiver:



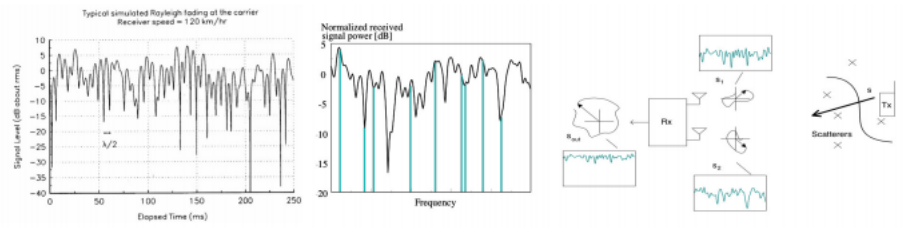
Simple zero-padding could be used to stop ISI, however if we use CPs instead, we can transform the convolutions and also obtain the zero-padding effect because we will simply disregard the received signal related with the CP. Thus, time synchronization is key.

7 Space-Time Coding

Diversity is a way of combating fading in communications. A summary:

Signal power in a wireless channel fluctuates (or “fades") with time/frequency/space.

Diversity is used to combat fading - "independent" fading links are combined.



- **Time diversity**: successive transmission of the same symbol (reduces symbol rate, more energy per symbol)
- **Frequency diversity**: transmission of the same narrowband signal on different frequencies (requires more bandwidth and power)
- **Spatial (receiver) diversity**: requires more hardware, but no extra bandwidth or time required

Diversity gain is related to the number of independent fading branches.

In essence, Space-time coding is a technique to exploit the diversity of multipath and increase the reliability of the channel. In these considerations, it is considered that there is only one receiver with one antenna. We consider a scenario like this because it is the simplest one and is scalable for MIMO scenarios. Consider also 2 transmitting antennas, thus a MISO situation. Space-Time coding is nothing more than a way of multiplexing the signals to transmit in the timeslots and antennas such that they can be received more reliably than simply by transmitting one after the other by exploring different versions of the same data.

Is possible to think about how receiving more copies of the same signal improves its quality in the way that it makes more likely to receive copies with an higher SNR. Space-time coding however does better than this and combines the signals in an optimal way.

One well known problem of simply transmitting in the two antennas the same signal is that there can be destructive interference at the receiver. One better way of doing it would be to send the symbols in orthogonal ways such that they don’t interfere with each other. For 2 and for 3 transmitters, the correspondent Alamouti Codes are below:

For 2 transmitters:

In time slot 1, transmit: $\mathbf{s} = [s_1, s_2]^T$ so that the received signal is

$$x_1 = h_1 s_1 + h_2 s_2 + n_1$$

In time slot 2, transmit: $\mathbf{s} = [-\bar{s}_2, \bar{s}_1]^T$ so that the received signal is

$$x_2 = -h_1 \bar{s}_2 + h_2 \bar{s}_1 + n_2 \Rightarrow \bar{x}_2 = -\bar{h}_1 s_2 + \bar{h}_2 s_1 + \bar{n}_2$$

The codebook is

$$\mathbf{S} = \begin{bmatrix} s_1 & -\bar{s}_2 \\ s_2 & \bar{s}_1 \end{bmatrix}$$

For 3 transmitters:

$$\mathbf{S} = \begin{bmatrix} s_1 & s_2 & \frac{s_3}{\sqrt{2}} \\ -\bar{s}_2 & \bar{s}_1 & \frac{s_3}{\sqrt{2}} \\ \frac{\bar{s}_3}{\sqrt{2}} & \frac{\bar{s}_3}{\sqrt{2}} & \frac{-s_1 - \bar{s}_1 + s_2 - \bar{s}_2}{2} \\ \frac{\bar{s}_3}{\sqrt{2}} & -\frac{\bar{s}_3}{\sqrt{2}} & \frac{s_2 + \bar{s}_2 + s_1 - \bar{s}_1}{2} \end{bmatrix}$$

Note that $\mathbf{S}^{\mathrm{H}}\mathbf{S} = \alpha I$, with $\alpha = \sum_i^N |s_i|^2$, thus they are orthogonal. For the case of a code for 2 antennas, $alpha = |s_1|^2 + |s_2|^2$.

However, only for the case of 2 transmitters is possible to achieve a full code, after that is necessary to sacrifice data rate to code information that way. Meaning that is necessary to take more time slots than symbols are used to transmit in orthogonal ways all symbols. For instance, for 3 antennas, the achieved coding rate is 3/4 meaning that is required to send 3 symbols in 4 different time slots.

Additionally, note that is possible to send much more than just 2 symbols in 2 antennas and many more than 3 in 3 antennas, this was just the minimal example. The codes can then be repeated for 4, 6, 8, (ect) symbols for the case of 2 antennas, and 3, 6, 9, 12 (etc) for the case of 3 and so on.

To finish, what is done at the reception:

The received signal

$$\begin{bmatrix} x_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 \\ \bar{h}_2 & -\bar{h}_1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ \bar{n}_2 \end{bmatrix} \Leftrightarrow \mathbf{x}' = \mathbf{H}'\mathbf{s} + \mathbf{n}'$$

The channel matrix is now orthogonal and is assumed to be known at the receiver. Therefore, the receiver simply implements

$$\hat{\mathbf{s}} = \mathbf{H}'^{T}\mathbf{x}'$$

The power is split equally across the transmit antennas, so SNR per symbol is given by

$$\mathrm{SNR}_i = \frac{|h_1|^2 + |h_2|^2}{2\sigma^2}$$

8 All code explained

8.1 First assignment

8.1.1 Instantaneous Model

The first part of this assignment consists on the creation of an instantaneous MIMO model. More exactly, in each antenna are received different versions of the same signals delayed by the propagation and by the displacement of the array sensors in relation to the source of the signal. If we consider the receiver is in the far-field of the transmitter where in incoming waves are practically plane waves and if we assume narrowband signals where if the delay of propagation is much smaller than the inverse bandwidth of the signal, then according to the result after some deductions presented in 1 one may write the received signal in each sensor for various sources as:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$$

$$\mathbf{X} = [\mathbf{x}(0), \cdots, \mathbf{x}(N-1)] \text{ and } \mathbf{S} = [\mathbf{s}(0), \cdots, \mathbf{s}(N-1)]$$

$$\mathbf{X} = \mathbf{A}\mathbf{S}$$

We will work with the last expression as we are in the digital domain.

We start by creating the A matrix, a matrix that has in each column a steering vector. Actually the only thing the function should do is creating

the steering vector, but if it receives as an argument more than one theta it creates A with those steering vector.

function a = gen_a(M, Delta, theta)

- **a** is the returned vector/matrix, size M x d
- **M** is the number of elements of the array of sensors and will be the number of elements in the steering vector.
- **Delta** is the spacing between the array elements in wavelengths. This is important because phase difference of the received signal in each element will depend on how far apart they are. In other words, it will influence the steering vector elements.
- **theta** will be the direction of arrival. It can be a column or row vector of dimension dx1 or 1xd ,respectively.

Then, we should see the priority that a certain direction of space will be given compared to other directions when a certain beamformer is applied. In practice, the response calculation is nothing more than plotting the signal processed after being received:

$$y(t) = \mathbf{w}^H \mathbf{x}(t) = \mathbf{w}^H \mathbf{a}(\theta)\beta s(t)$$

The response of the array to a unit-amplitude signal, $|\beta s(t)| = 1$, from direction θ is

$$|y(t)| = |\mathbf{w}^H \mathbf{a}(\theta)|$$

Therefore is enough to calculate the result of that modulus of that multiplication for each theta and plot it after.

function y = spat_response(w, Delta, theta_range)

- **w** is the beamformer. Is nothing more than a set of weights that will be multiplied to the received signal as an attempt to recover the transmitted signal.
- **Delta** is necessary to generate the steering vector for a given theta.
- **theta_range** is the theta values for which the calculation will be made. It should be of the type:

```
theta_range = linspace(-90, 90, N_points);
```

Where N_points will be the number of points of our spatial response plot.

- **y** is the absolute value of the beamformer multiplied by the steering vector for every theta, so is ready to plot against all thetas.

Meaning and Results : we should expect to see the results from the slides when we apply an unitary beamformer. It should give us something every equal to the plot we use to compute the antenna pattern. Remember that the antenna pattern can be drawn from the the steering vector plot. There will be repetition if Delta is bigger than $\frac{1}{2}$, i.e ambiguity. And the zeros will be at 2pi / M apart except from the in the main lobe. Note additionally that is possible to have an "offset" of the main lobe! Namely if we calculate a beamformer that consider that direction to be the direction of arrival of the signal, like $\mathbf{w} = \mathbf{a}(30\,\text{deg})$

Moreover, it is also possible to **scan the directions of the sources** with this approach. But now we won't use the perfect signal having the steering vector as the propagation environment, but instead the received signal. We just need to apply the beamformer to the received signal. **So to do this, we are not going to use this function.**

$$y(t) = \mathbf{w}^{\mathrm{H}}\mathbf{x}(t) = \mathbf{w}^{\mathrm{H}}\mathbf{a}(\theta)\beta s(t)$$

The response of the array to a unit-amplitude signal, $|\beta s(t)| = 1$, from direction θ is

$$|y(t)| = |\mathbf{w}^{\mathrm{H}}\mathbf{a}(\theta)|$$

Therefore, we should find a way of creating a plausible signal with N symbols of it and have it coming from d directions with noise.

$$\mathbf{X} = \mathbf{A}_\theta \mathbf{S} + \mathbf{N} \tag{18}$$

function X = gen_data(M, N, Delta, theta, SNR)

- **X** will be our data vector, M by N, having N symbols for each of the M sensors.
- $\mathbf{A}_\theta$ is M by d for the d directions;
- **S** is d by N, and is the signal. Because we won't try to estimate the signal yet, this function just put there some value. In this case, a normal distribution is sampled.

- theta will be the set of angles/directions from which the signal comes from, d by 1

- **SNR** parameter will scale the variance of entrances of **N** . It will be the SNR of the signal that is received. If the transmitted signal has power 1, because the noise power and noise variance are closely connected, one may set the variance of noise generation to achieve the desired SNR.

As the variance of a random vector is given by: $Var(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$ and the power of that vector is given by the square of the norm, therefore $P(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i)^2$, we can see that the power is equal to the variance because the variance has zero mean.

Therefore, one can find the variance that should use to generate N such that the final signal ends up with the required SNR this way: $SNR = \frac{1}{Var(\mathbf{N})} \Leftrightarrow Var(\mathbf{N}) = \frac{1}{SNR}$. Remember that the SNR there is linear, not in dBs. And that because we are generating a noise vector for each of the M sensors, the number of elements to consider here is the number of sensors M.

To properly explain what was and should be done in this assignment, have in mind these two graphs:

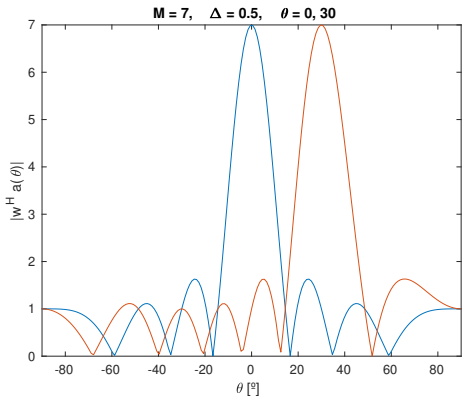


Figure 2: A figure

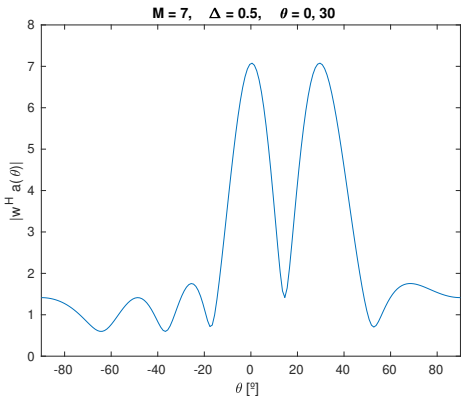


Figure 3: Another figure

For figure 2 I've applied one beamformer to a changing signal. The beamformers were exactly the same in structure, a matched filter, meaning $\mathbf{w} = \mathbf{a}(\theta)$, but one had $\theta = 0°$ and the other $\theta = 30°$ while the signal changed in directions, from -90 to 90 degrees.

This is NOT what you are suppose to do to estimate the spatial response of your beamformer. It is a way of knowing how much will you be able to take from the signal when it comes from all those directions but you are using a beamformer specific for one. In other words, that is evaluating a

beamformer for a specific direction. But won't tell you how the principle of the beamformer performs faced with a signal that comes for more than 1 direction.

Because we can change the signal very easily, by changing some parameters and generating a new set of data, we should rather check how will the beamformer perform in all directions if we are given a dataset that has signal coming from 2 directions.

In essence, it is the difference between checking what a specialized beamformer will take from the signal for signals coming from different directions if the beamformer is built for a certain θ versus how the beamformer will respond if crafted for different directions given that the signal comes from 2 in specific. Figure 2 is the first case, Figure 3 is the second case.

Finally the truth : neither case is what we are suppose to do, so why did I get the correct result?

When we did $\mathbf{w} = \mathbf{a}(\theta_1)+\mathbf{a}(\theta_2)$ and proceeded to change the source for different thetas, $\mathbf{a}(\theta)$, the response we were calculating was $y(\theta) = |\mathbf{a}(\theta_1)^{\mathrm{H}}\mathbf{a}(\theta)+ \mathbf{a}(\theta_2)^{\mathrm{H}}\mathbf{a}(\theta)|$.

What we were suppose to do is check how would the matched filter receives the signal, thus, we should make $\mathbf{w} = \mathbf{a}(\theta)$ and the signal $x = \mathbf{a}(\theta_1) + \mathbf{w} = \mathbf{a}(\theta_2)$. Now, the response will be $y(\theta) = |\mathbf{a}(\theta)^{\mathrm{H}}\mathbf{a}(\theta_1) + \mathbf{a}(\theta)^{\mathrm{H}}\mathbf{a}(\theta_2)|$.

Note that if we don't consider the absolute value of each response, they are the hermitian of each other. Considering the absolute value, the response will be the absolute (no pun intended) same! That was why the wrong thoughts resulted in the correct response. Luck, or bad luck, perspective dependent.

The conclusions with the changes in the DOA, M, N and Delta is:

- changing Delta affects only the ambiguity of our response as the array spatial response (a short transformation away for being the array radiation diagram) will be repeated more often. This will lead to wider lobes because only the non repetitive interval will be mapped into the array radiation diagram and if we plot the unambiguous interval of angles we see that the lobes will be wider.

- changing M influences the number of lobes. Them become more narrow as well because a bigger number of lobes must fit in the same unambiguous interval.

- varying N increases the resolution of the graph.

- The closer the two signals have Directions Of Arrival, the harder it will be to distinguish them and closer will be the maximum of the spatial responses.

Finally, we were suppose to vary DOA, M, and the SNR to see changes in the singular values of X. First and foremost, we need to calculate and plot these values.

Singular Value Decomposition (SVD) is something that is very important. Again, remoting to other section: see 3.4 and 3.5 for a deeper svd insight and the explanation behind some conclusions.

There is a Matlab functions that calculates the SVD and the singular values will be in the diagonal of the central matrix. Plotting this values we see that:

- decreasing the SNR will increase the singular values that should be 0;
- increasing the number of samples will increase all singular values because we'll have more contributions, simply to larger singular values, but the same singular vectors. That is why all singular values grow in the exact same proportion with N. It may help suppress the noise because threshold selection will be easier;
- the closer the DOA get, the more mixed with the noise will the singular values be. The second singular value will be closer to the noise;
- increasing M will increase only the singular values of interest because we'll be able to distinguish better the DOA and given them a bigger gain.

An important notion is that the more elements we have, the best directivity we can achieve paying by the increase in the number of secondary lobes. However, it will be far easier to distinguish directions. Actually, a formula that comes from Antenna Theory is that for a broadside array, where the main lobe is along the direction perpendicular to the orientation of the antenna elements: Main Beam Width = 2 arcsin $\frac{1}{M\triangle}$, where the $\triangle$ represents the distance in wavelengths between array elements.

8.1.2 Convolutive Model

Firstly, a pulse needs to be generated, the pulse to be transmitted. This can be done with a mathematical function of the pulse.

function g = pulse(tau, L, P)

- **tau** is the delay of the pulse. For tau = 0, the pulse should start in the beginning. For different delays, it should start later (there are no negative delays);
- **L** will be the channel length: there won't be considered any events passed L symbols from the present. In other words, the L-th symbol sent before the current one can still influence what is received. Before L, we don't consider the influence of any other symbol.
- **P** will be the amount of samples taken in between symbols. This is a way of guaranteeing the correct reception because we'll create a much taller **X** matrix.

The pulse function will be useful to construct the channel response. The channel response will be pulses scaled by each β_i and delayed by each τ_i, where β_i and τ_i characterise the gain and delay of the path i from the sender to the receiver. Thus, the channel function will need these informations and will use the pulse function.

What it will do is add the pulses with the τ delays, β scalings to a single signal which will be the channel. Each pulse required the channel taps length L to have enough zeros and P to sample the signal at those intervals.

function h = channel(tau, beta, L, P)

One fulcral question to ask : what if the inserted delays are not multiples of $\frac{1}{P}$? How will we do the pulses and thus the channel? Well, because we are only interested in the sampling times, we will simply sample the signal at those instances. The delay will influence the value of that pulse at those sampling intervals and that should be accounted.

As expected is necessary to create a signal to be transmitted. Changing the phase in quarters is enough to create a QPSK signal. This function returns N symbols of the source.

function s = source(N)

From the signal created, we must receive it. Thus passing it through the channel oversampled by a factor of P will create our complete vector of samples. It is important to note that is usual to separate the multipath effects (like we are experiencing here) from the spatial effects from having the signal coming from different directions. The macroscopic channel model is general enough to include both effects, we simply don't consider them at the same time for the simplicity of not working with matrices with more than 2 dimensions.

Indeed that is what happens, we've been calling the instantaneous model to the model where we don't oversample and don't consider multipath delays and calling convolutive model to the model where we do this yet only with one antenna.

function x = gen_data1(h,s,P,N)

This function creates a vector of convolutions:

1. Firstly, fills the source signal with P-1 zeros in between each symbol. This is a necessary step for the convolution with the channel;

2. Then, convolves the resulting vector with the channel. This way, the channel is simulated and the oversampling as well.

The result of this function is a vector of length $N \times P$. To create the matrix at the end, we need to reorganize this vector in columns of P elements.

The final question was about the rank of the matrix A, with P rows and N columns. Well, the maximum rank is the smallest of the dimensions: if it has more unknowns than equations, the maximum rank is the number of equations. If it has more equations than unknowns, the maximum rank is the number of unknowns.

8.2 Second Assignment

8.2.1 Receiver algorithms for the instantaneous Model

As usual, we start with the instantaneous model, no time dependences. According to the data generation function previously built we are asked to implement a Matched filter, a Zero-forcing receiver and a Weiner Filter. Note that contrary to the first assignment where the vector of weights was a vector, now it will be a matrix because we want to apply different vectors of weights to extract different directions of arrival of the signals. To apply receiver, one must only do the following:

$$\hat{\mathbf{S}} = \mathbf{W}^{\mathrm{H}}\mathbf{X} \tag{19}$$

It is important know why they are different and in what they differ. After knowing that, simply by the way they act one can tell which type of receiver is implemented.

The Matched Filter is nothing more than $\mathbf{W} = \mathbf{A}$. It minimises the noise by "turning the channel around". Because the channel has zero mean, it optimises the reception of the symbols with noise. $A^H(AS + N) = A^HAS + A^HN$. It is a cheap receiver Note however that the transpose of A normally is not its inverse (This would only be the case if the matrix was

why?because is less computational intensive?

orthogonal which doesn't happen in most situations). Thus this receiver is not expected to deliver the best results since it only addresses noise and disregards ISI.

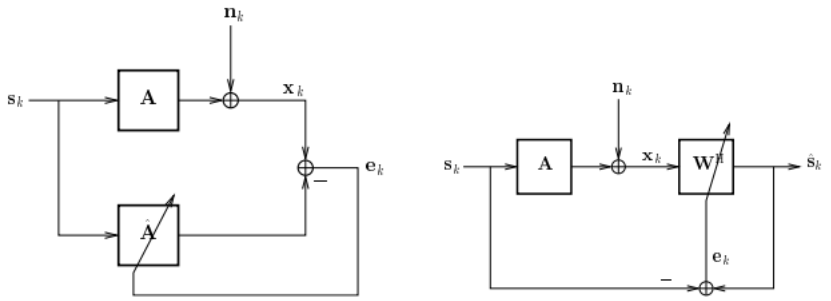
In a nutshell, the next receivers follow:

- *Model matching:* minimize residual

$$\min_{\mathbf{S}} \|\mathbf{X}-\mathbf{AS}\|_{\mathrm{F}}^2, \qquad \text{or} \qquad \min_{\mathbf{A}} \|\mathbf{X}-\mathbf{AS}\|_{\mathrm{F}}^2$$

- *Output error minimization:*

$$\min_{\mathbf{W}} \|\mathbf{W}^{\mathrm{H}}\mathbf{X}-\mathbf{S}\|_{\mathrm{F}}^2,$$



The first refers to the Zero-forcing receiver, and is named like this because it tries to zero the difference between the received signal and the model, by changing A or S.

The Zero-forcing Receiver in a line is: $\mathbf{W^H} = \mathbf{A}^\dagger = (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}$, the pseudo-inverse of A. This, by definition, will invert A and thus nullify ISI.

The second alludes to the Weiner Filter. Instead of minimising the difference between the received and estimated, it minimises the error between the estimated and the true symbols.

The Weiner Receiver , also called the *Linear Minimum Mean Square Error* (LMMSE) receiver, implements $\mathbf{w} = (\mathbf{A}\mathbf{A}^{\mathrm{H}} + \sigma^2\mathbf{I})^{-1}\mathbf{A}$, a compromise between interference and noise, maximising the SINR.

Linearly Constrained Minimum Variance LCMV, or *Minimum Variance Distortionless Response* (MVDR) is practically a type of problem. We need to minimise something while constraint that some other thing hold. In particular, we want that $\mathbf{w}^H\mathbf{a} = 1$. Via the Lagrange Multipliers, we reach the answer:

$$\mathbf{w} = \mathbf{R}_x\mathbf{a}(\mathbf{a}^{\mathrm{H}}\mathbf{R}_x^{-1}\mathbf{a})^{-1} \tag{20}$$

The generalisation is as follows:

Generalization

- Introduce a constraint matrix $\mathbf{C}$: $M \times L$ ($M > L$) and an L-dimensional vector $\mathbf{f}$
- The general LCMV or MVDR problem can then be written as

$$\min_{\mathbf{w}} \mathbf{w}^{\mathrm{H}}\mathbf{R}_x\mathbf{w} \qquad \text{such that} \quad \mathbf{C}^{\mathrm{H}}\mathbf{w} = \mathbf{f}$$

- Solution:

$$\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{C}(\mathbf{C}^{\mathrm{H}}\mathbf{R}_x^{-1}\mathbf{C})^{-1}\mathbf{f}$$

This will be particularly important for the next part of the assignment.

8.2.2 Direction of Arrival Estimation

This section shows how to apply the beamformers, but note that what we truly want is estimate the direction of arrival of the signal. Therefore, we will plot all the values instead of picking only the maximum.

With the standard beamformer, the direction of arrival can be found like:

The classical beamformer

- The *classical beamformer* (Bartlett beamformer) is $\mathbf{w} = \mathbf{a}(\theta)$.
- This corresponds to the matched filter assuming spatially white noise.
- Find $\mathbf{w} = \mathbf{a}(\theta)$ that maximizes the output power

$$\hat{\theta}_0 = \max_{\theta} \frac{\mathbf{a}(\theta)^{\mathrm{H}}\mathbf{R}_x\mathbf{a}(\theta)}{\mathbf{a}(\theta)^{\mathrm{H}}\mathbf{a}(\theta)}.$$

  - For finite data, replace $\mathbf{R}_x$ by the sample covariance matrix $\hat{\mathbf{R}}_x$.
  - With known colored noise, replace denominator by $\mathbf{a}(\theta)^{\mathrm{H}}\mathbf{R}_n\mathbf{a}(\theta)$.

With the MVDR beamformer, the direction of arrival can be found this way:

MVDR

- In MVDR we try to minimize the output power, while constraining the power towards the direction θ:

$$\hat{\theta}_0 = \max_{\theta}\{\min_{\mathbf{w}} \mathbf{w}^{\mathrm{H}}\hat{\mathbf{R}}_x\mathbf{w} \qquad \text{subject to} \quad \mathbf{w}^{\mathrm{H}}\mathbf{a}(\theta) = 1\}\,.$$

This yields

$$\mathbf{w} = \frac{\hat{\mathbf{R}}_x^{-1}\mathbf{a}(\theta)}{\mathbf{a}(\theta)^{\mathrm{H}}\hat{\mathbf{R}}_x^{-1}\mathbf{a}(\theta)}$$

$$\hat{\theta}_0 = \max_{\theta} \frac{1}{\mathbf{a}(\theta)^{\mathrm{H}}\hat{\mathbf{R}}_x^{-1}\mathbf{a}(\theta)}$$

- For multiple signals, choose again the d largest local maxima.

With MUSIC, the direction of arrival is found following:

MUSIC (Multiple SIgnal Classification) algorithm

- Eigenvalue-based technique (assume $d < M$):

$$\mathbf{R}_x = \mathbf{A}\mathbf{R}_s\mathbf{A}^{\mathrm{H}} + \sigma^2\mathbf{I}_M = \mathbf{U}_s(\mathbf{\Lambda}_s + \sigma^2\mathbf{I}_d)\mathbf{U}_s^{\mathrm{H}} + \mathbf{U}_n(\sigma^2\mathbf{I}_{M-d})\mathbf{U}_n^{\mathrm{H}}$$

$$\text{span}(\mathbf{U}_s) = \text{span}(\mathbf{A})\,, \qquad \mathbf{U}_n^{\mathrm{H}}\mathbf{A} = 0\,, \qquad \text{where} \;\; \mathbf{A} = [\mathbf{a}(\theta_1)\,,\, \cdots\,,\, \mathbf{a}(\theta_d)].$$

- Choose $[\theta_1\,,\, \cdots\,,\, \theta_d]$ to make $\mathbf{A}$ *fit* span$(\mathbf{U}_s)$:

$$\mathbf{U}_n^{\mathrm{H}}\mathbf{a}(\theta_i) \;=\; 0\,, \qquad (1 \le i \le d)$$

- Choose the d lowest local minima of the cost function

$$J_{MUSIC}(\theta) \;=\; \frac{\|\hat{\mathbf{U}}_n^{\mathrm{H}}\mathbf{a}(\theta)\|^2}{\|\mathbf{a}(\theta)\|^2} \;=\; \frac{\mathbf{a}(\theta)^{\mathrm{H}}\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}\mathbf{a}(\theta)}{\mathbf{a}(\theta)^{\mathrm{H}}\mathbf{a}(\theta)}$$

  - In a graph, we plot the inverse of $J_{MUSIC}(\theta)$.
  - If number of sources smaller than number of sensors ($d < M$), we get the *exact* DOAs for $N \to \infty$ or SNR $\to \infty \Rightarrow$ *statistically consistent* estimates.

Then we just need to plot them in dB. By changing the number of elements in the array there is a gain in resolution. If we relax the proximity of the directions, to [0°,60°] for instance, there will be a very clear peak identified by each algorithm.

8.2.3 Receiver algorithms for convolutive model

If we construct the Wiener Receiver for each Row of S_L, we should take each column of H to do that!

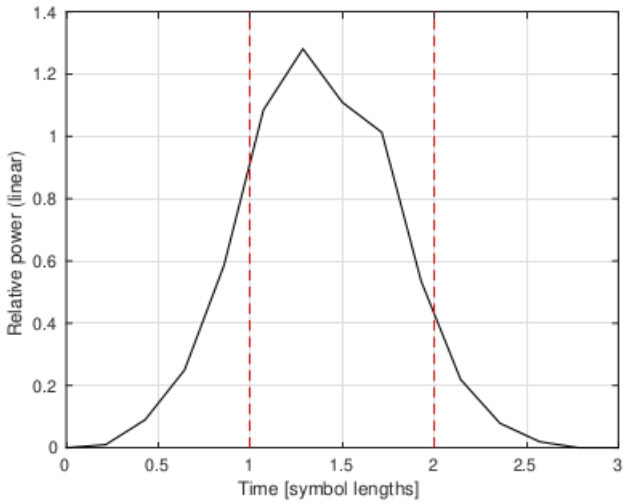
Therefore, the beamformer would be:

$$\mathbf{w} = (\mathbf{H}_i \mathbf{H}_i^{\mathrm{H}} + \sigma^2 + \mathbf{I})^{-1} \mathbf{H}_i \tag{21}$$

And the row we will be able to receive the best it he second one because the each column of H (remember H is P by L) has a symbol period, thus from where the most power is we can tell where the best response will be.

Now, why do we take the columns of **H** as the beamformer for the rows of S_L? Simple, because that is what the beamformer does: it takes the columns of A to do exactly that. Note however that this beamformer uses the columns when estimating the symbol that came in direction d, using the a column to estimate a row. Thus, by analogy, we can use the same.

From the picture below, taken from the report submitted for this assignment, is possible to see that the most power is concentrated between the first and second symbol intervals:



8.3 Third Assignment

Similarly to the last part of the previous assignment, we should create a matrix of data with P rows and N columns, where along the rows are the samples between symbols and along a row are the symbols.

8.3.1 Pilot-based Estimation

Firstly, the previous function **gen_data1** now creates a vector of $M \times N$ samples where the noise is also included, this will be the received signal.

Then with that vector, knowing what is being transmitted, knowing as well what is being received, we should be able to extrapolate the channel.

Pilot-based Estimation is nothing more than estimation of the channel knowing what was transmitted.

From $\mathbf{X} = \mathbf{HS}$, we know that $\mathbf{H} = \mathbf{XS}^{-1}$, therefore we need to calculate the inverse of S. However, S is not a square matrix, thus we must do a pseudo-inverse! Matlab has a function to compute this pseudo-inverse and gives very accurate results. Depending on the SNR, the estimation will be more or less accurate. Below the results of this estimation for two SNR values.

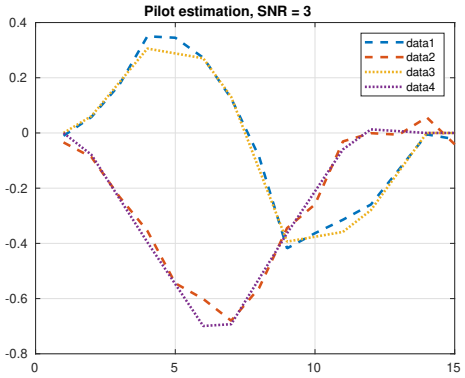


Figure 4: Pilot estimation, SNR = 3dB

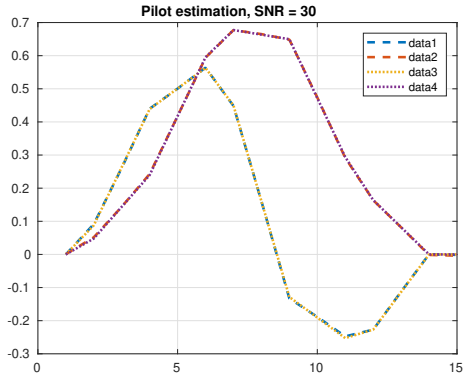


Figure 5: Pilot estimation, SNR = 30dB

The rest of the homework can have its own section

9 Blind Estimation

This is an important section. Without knowing what was transmitted or anything about the channel, just by knowing what was received, we may be able to extrapolate information from the channel. More precisely, from $\mathbf{X}$ we can tell if that is possible or not.

First, we compute SVD ()$\mathbf{X}$) and get:

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{H}} = [\hat{\mathbf{U}} \quad \hat{\mathbf{U}}^{\perp}] \left[\begin{array}{cc|cc} \sigma_1 & & & \\ & \sigma_d & & \\ \hline & & 0 & \\ & & & 0 \\ \hline 0 & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & 0 \end{array}\right] \begin{bmatrix} \hat{\mathbf{V}}^{\mathrm{H}} \\ (\hat{\mathbf{V}}^{\perp})^{\mathrm{H}} \end{bmatrix}$$

$$\mathbf{U} : M \times M, \quad \boldsymbol{\Sigma} : M \times N, \quad \mathbf{V} : N \times N$$

Recalling a slide:

- The starting point of our blind estimation methods is the SVD of $\mathcal{X}$:

$$\mathcal{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{H} = \mathbf{U}_s\boldsymbol{\Sigma}_s\mathbf{V}_s^{H} + \mathbf{U}_n\boldsymbol{\Sigma}_n\mathbf{V}_n^{H}$$

  where $\boldsymbol{\Sigma}_n$ is only non-zero if there is some noise.

- $\mathbf{U}_n$ describes the subspace orthogonal to the columns of $\mathcal{X}$ and $\mathbf{V}_n$ describes the subspace orthogonal to the rows of $\mathcal{X}$:

$$\mathbf{U}_n^{H}\mathcal{X} = \mathbf{0} \qquad \mathcal{X}\mathbf{V}_n = \mathbf{0}$$

- Because the columns of $\mathcal{X}$ are linear combinations of the columns of $\mathcal{H}$, we have

$$\mathbf{U}_n^{H}\mathcal{X} = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{U}_n^{H}\mathcal{H} = \mathbf{0}$$

- Because the rows of $\mathcal{X}$ are linear combinations of the rows of $\mathcal{S}$, we have

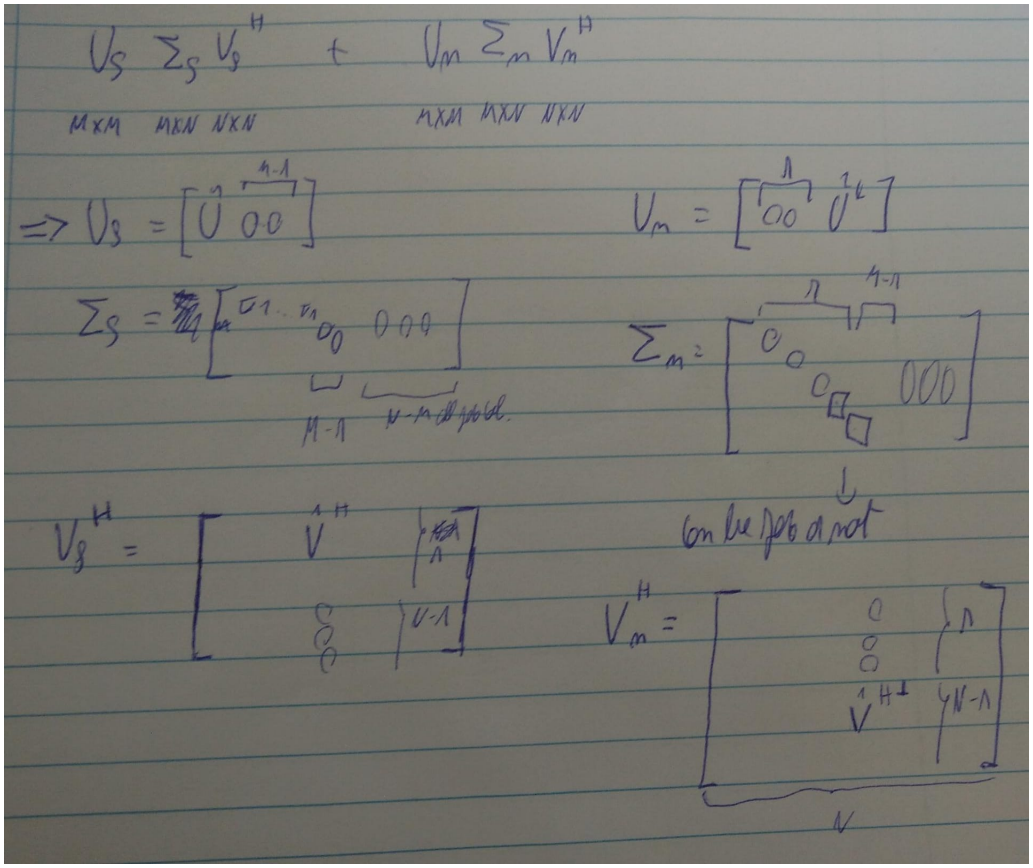
$$\mathcal{X}\mathbf{V}_n = \mathbf{0} \quad \Leftrightarrow \quad \mathcal{S}\mathbf{V}_n = \mathbf{0}$$

- The blind methods are now obtained by exploiting the structure in $\mathcal{H}$ and $\mathcal{S}$.

And some important parts to take into account more carefully are that $\boldsymbol{\Sigma}_n$ is only non-zero in presence of noise.

This seems a weird way of writing the previous SVD formulation because what is now a sum was previously part of the same matrices, thus, how can them be summed together? Are them padded to have the same dimensions? Because zero padding matrices is ok as it doesn't change the space they span or anything like that.

Thus, the matrices with a subscript, like $\mathbf{U}_n$ and $\mathbf{V}_n^{\mathrm{H}}$ will in fact have the same dimensions as the matrices obtained in the first SVD but with many more zeros. If the rank of $\mathbf{X}$ is r, then the matrices sizes will be:



One important thing to notice is that it is completely useless to add columns or rows of zeros. This may be the reason that sum is made in the slides without further justifications, because one can add them practically without notice because they don't change anything in the matrix.

With this said, one can more simply declare the matrices with subscripts:

- $\mathbf{U}_s = \hat{\mathbf{U}}$
- $\mathbf{U}_n = \hat{\mathbf{U}}^{\perp}$
- $\mathbf{V}_s^{\mathrm{H}} = \hat{\mathbf{V}}^{\mathrm{H}}$
- $\mathbf{V}_n^{\mathrm{H}} = \hat{\mathbf{V}}^{\mathrm{H}\perp}$

Now, from the relations presented in the slide above, we have matricial equations to find the channel or the symbols (depending the estimation we want to do). Because of the Toeplitz structure of H and S, we can massively simplify the problem.

For Channel Estimation:

- For a single user, we can transform $\mathbf{U}_n^H \mathcal{H} = \mathbf{0}$ into

$$\begin{bmatrix} \mathbf{U}_{n,1}^H & \cdots & \mathbf{U}_{n,m}^H \end{bmatrix} \begin{bmatrix} \mathbf{h}_0 & \cdots & \mathbf{h}_{L-1} & & \mathbf{0} \\ & \ddots & & \ddots & \\ \mathbf{0} & & \mathbf{h}_0 & \cdots & \mathbf{h}_{L-1} \end{bmatrix} = \mathbf{0}$$

$$\Leftrightarrow \underbrace{\begin{bmatrix} \mathbf{U}_{n,1}^H & & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{U}_{n,m}^H & & \mathbf{U}_{n,1}^H \\ & \ddots & \vdots \\ \mathbf{0} & & \mathbf{U}_{n,m}^H \end{bmatrix}}_{\mathbf{U}_{n,T}} \begin{bmatrix} \mathbf{h}_0 \\ \vdots \\ \mathbf{h}_{L-1} \end{bmatrix} = \mathbf{0}$$

- This can be solved by finding the right null-space of $\mathbf{U}_{n,T}$ (through the SVD).
- If the null-space has dimension one, there is a solution up to a scalar ambiguity.
- If the null-space has a larger dimension, there are too many solutions.

For Symbols Estimation:

- For a single user, we can transform $\mathcal{S}\mathbf{V}_n = \mathbf{0}$ into

$$\begin{bmatrix} s_0 & \cdots & s_{N-1} \\ \vdots & & \vdots \\ s_{-L-m+2} & \ddots & s_{N-L-m+1} \end{bmatrix} \mathbf{V}_n = \mathbf{0}$$

$$\Leftrightarrow \begin{bmatrix} s_{-L-m+2} & \cdots & s_{N-1} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{V}_n & & & \\ & \mathbf{V}_n & & \\ & & \ddots & \\ & & & \mathbf{V}_n \end{bmatrix}}_{\mathbf{V}_{n,T}} = \mathbf{0}$$

- This can be solved by finding the left null-space of $\mathbf{V}_{n,T}$ (through the SVD).
- If the null-space has dimension one, there is a solution up to a scalar ambiguity.
- If the null-space has a larger dimension, there are too many solutions.

What both of these have in common in this clever construction is that we can gain many more equation than unknowns to make our problem solvable. Being solvable is having a solution or at least having a fundamental constraint not o have one. Lack of rank is not a fundamental constraint as with this organisation we can basically control the rank of the matrix we are multiplying to the channel or to the symbols. A fundamental constrain is when we can "add more rank" because increasing the number of rows that way won't add anything else ot the signal because the pattern is there already.

So, it is very important to understand well what m is. m will dictate the number of times we are going to repeat our original matrices in case of $\mathbf{V}_n$

or our original columns of $\mathbf{U}_s$. $\mathbf{U}_s$ will be m*P x (m*P-r) thus its transpose can always be divided in m sets of P columns.

Then, we need to calculate the row or columns nullspace of the resulting matrix to know if there is a unique and solution to our problem or if there are too many that we can't find the one we are looking for. This last part is related with the fundamental constraint.

So, after obtaining the matrices from the first SVD, we should build yet another matrix and choose an m that allows us to have full rank on the matrix we are looking for and then calculate the svd of that matrix and extrapolate if the problem is solvable or not. If it is, then the solution will be in the columns or rows nullspace of the matrix we just constructed. If that space and dimension 1, then we are a scalar away from the solution, meaning the symbols norm or angle needs to be scaled to "reality". That is not a big problem at all. However if we get a nullspace with dimension bigger than 2, it is already impossible to achieve the solution we are looking for.

Note that the slides posted above, about channel and symbol estimation already say what subspace you should find: right nullspace for UnT, left nullspace for VnT. This because the Vn,T Matrix has a matrix on its left, while Un,T matrix has a matrix on its right. Thus the solution to those equations will be the nullspaces that when the matrix is used to multiply another at that nullspace side, it results in 0. AB=0 thus, solutions will lie in A right nullspace and B left nullspace.

The solutions of an homogeneous matricial equation lie in the left nullspace of the matrix on the right and on the right null space of the matrix on the left. If we decompose a matrix thought the SVD, U gives us the span of the columns but also the left nullspace, the nullspace of the columns. These are orthogonal. And so are the span of the rows and the right nullspace, the nullspace of the rows. There is a relation between spans of columns(rows) and nullspaces of columns(rows), they are complementary:

- **if the rank is smaller than the amount of columns, then the span of columns won't possible generate the full space, thus there will be a null space that complements the span, the left nullspace;**

- **if the rank is smaller than the number of rows, the span of the rows of the matrix can't possible have enough independent vectors to span the space of rows, and we call the right nullspace, nullspace of the rows or simply nullspace of the matrix to that subspace;**

The left nullspace is the kernel of the matrix transposed and the nullspace simply the kernel of the matrix, by definition. With this said, after calculating the SVD of the matrix constructed, one must check the dimensions of the correspondent nullspace. If the dimension is 1, then there can be found a solution. If there isn't, then there are way to many and the difference between them will practically always be more than a scalar.

Important to notice as well is the m: m is only the amount of time we will repeat the matrices. m is set by us as a mean of having an invertible H. We create a new, more exotic, calligraphic H that has the channel coefficients repeated in each row. Thus, we had M more equations and 1 unknown for each unit we increase m. This way, we will end up with a calligraphic H matrix that is $m \times M \times P$ by $L + m - 1$ and we just have to have an m sufficiently high that leads this matrix to be tall and thus invertible. This is where the conditions $m\ M\P \geq L + m - 1$ comes from. Further it simplifies to: $m \geq \frac{L-1}{MP-1}$.

This is very important because with $MP = 5$, we can only keep an $m = 1$ until the L reaches 5. However, increase the channel can bring other problems besides invertibility, even if it is only increased with zeros...

10 Final Assignment

10.1 Estimation of Directions and Frequencies

10.1.1 Signal Model

The data generation is basically:

- using gen_a for generating the steering vectors, i.e matrix A, for the directions of arrival;
- generating the signal as described. It has always unitary energy by the assignment, thus the power will be unitary and we won't need to go through what we've done to get the signal power to know the variance of the noise to apply;
- Despite source energy is 1, and the elements of A having modulus 1, the signal A*S doesn't have power 1. Thus, we need to use that power to compute the SNR and find the variance of the noise to be added to the signal to achieve the required SNR.

Then, as a good practice, we should plot the singular values and see if the expected variations are taking place.

10.1.2 Estimation of directions

ESPRIT is the best way of estimation directions of arrival of the signals. It doesn't need to sweep the entire space of directions.

The method is very well outlined in the slides.

1. We need a signal with a certain structure of its column space. Because its column space is a subset of the space spanned by A, in case of $X = AS$, then A needs to have a specific structure. That structure is shift-invariance: the element below should always be a factor away from the above.

   - For a uniform linear array, $\phi = e^{j\Delta 2\pi \sin(\theta)}$, and thus

   $$\mathbf{a}(\theta) = \begin{bmatrix} 1 \\ \phi \\ \phi^2 \\ \vdots \\ \phi^{M-1} \end{bmatrix} \begin{matrix} \} \mathbf{a}_x(\theta) \\ \} \mathbf{a}_y(\theta) \end{matrix}$$

   - Shift-invariance property:

   $$\mathbf{a}_x(\theta) := \begin{bmatrix} 1 \\ \phi \\ \vdots \\ \phi^{M-2} \end{bmatrix}, \quad \mathbf{a}_y(\theta) := \begin{bmatrix} \phi \\ \phi^2 \\ \vdots \\ \phi^{M-1} \end{bmatrix}, \quad \text{so that} \quad \mathbf{a}_y(\theta) = \mathbf{a}_x(\theta)\phi$$

   This is important because this property is a space assumption that if met can simplify the computations tremendously.

2. We need to start grouping from the matrix X as we did in the above example. The matrix X is M by N, sensors by samples, and will be divided in a new X and a Y:

- Let us group the first and last $M-1$ antennas

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_{M-1}(t) \end{bmatrix}, \quad \mathbf{y}(t) = \begin{bmatrix} x_2(t) \\ \vdots \\ x_M(t) \end{bmatrix}, \quad \begin{aligned} \mathbf{X} &= [\mathbf{x}(0) \quad \cdots \quad \mathbf{x}(N-1)] \\ \mathbf{Y} &= [\mathbf{y}(0) \quad \cdots \quad \mathbf{y}(N-1)] \end{aligned}$$

- From the shift-invariance property:

$$\begin{aligned} \mathbf{x}(t) &= \sum_{k=1}^{d} \mathbf{a}_x(\theta_k)\beta_k s_k(t) && \Rightarrow \mathbf{X} = \mathbf{ABS} \\ \mathbf{y}(t) &= \sum_{k=1}^{d} \mathbf{a}_y(\theta_k)\beta_k s_k(t) = \sum_{k=1}^{d} \mathbf{a}_x(\theta_k)\phi_k\beta_k s_k(t) && \Rightarrow \mathbf{Y} = \mathbf{A\Theta BS} \end{aligned}$$

where

$$\mathbf{A} = [\mathbf{a}_x(\theta_1) \quad \cdots \quad \mathbf{a}_x(\theta_d)], \quad \mathbf{\Theta} = \begin{bmatrix} \phi_1 & & \\ & \ddots & \\ & & \phi_d \end{bmatrix}, \quad \phi_k = e^{j2\pi\Delta\sin(\theta_k)}$$

This new X and Y construct, from taking all but the last and all but the first rows of the received samples vector, respectively. They can be obtained in a different way, but that is what we are trying to estimate! **Note:** our objective will be the matrix Θ with the ϕ. From those entrances in the diagonal we invert the to retrieve the thetas.

3. Stack them:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{A\Theta} \end{bmatrix} \mathbf{BS}$$

4. Compute the SVD of **Z**. **HOWEVER:** Bear in mind that what we really want in the economy size svd. Because of noise that won't be possible, thus we have to take from U the parts that matter to get the signal that interests us.

$$\mathbf{Z} = \hat{\mathbf{U}}_z \hat{\mathbf{\Sigma}}_z \hat{\mathbf{V}}_z^{\mathrm{H}} \tag{22}$$

The dimensions of Z should be: 2 × (M-1) by N.

Thus, $\hat{\mathbf{U}}_z$ should be 2 × (M-1)

Note that ESPRIT can estimate very effectively and efficiently a vector with the structure of a steering vector, i.e the next element can be obtained from the previous by a multiplication from a ϕ. ESPRIT returns that ϕ. Thus, one may do the same for the transmitted symbols because they depend on the frequency. One can do the same to delays because they can be represented in the frequency domain by a phase shift. And one can even find the frequencies of signals which the symbols don't depend that directly on frequency, but knowing that each signal can be decomposed in a Fourier Series with many frequencies which will be represented as exponentials in time.

10.1.3 Estimation of frequencies

In particular, for this frequency estimation one simply had to note the similarities between the two problems.

If $\mathbf{X} = \mathbf{AS}$, allows the complete θ estimation from A given X, then $\mathbf{X}^T = \mathbf{S}^T\mathbf{A}^T$ should allow the complete frequency estimation, given that we provide the algorithm with X transposed.

This not only will put the symbols in a column, achieving the structure we need, but will make X transposed have its range as a subspace of the span of the columns of S^T, thus making the estimation possible. Because we are going to estimate the factor that relates consecutive elements, we then need to get the frequencies out.

10.1.4 Comparison

From the previous derived ESPRIT implementations for estimating frequencies and DOA, we can construct the matrices S and A, respectively. From there, knowing σ of the noise, one may create the ZF beamformers.

One of them will perform better than the other, but why?

The answer lies in what we are estimating. From the esprit for thetas, we estimate the thetas based on the columns of A, which are are determined exactly and span the subspace of columns of X. On the other hand, determining S will make us able to estimate A instead of determining it precisely, thus resulting in worse results in terms of interference.

10.1.5 Constant Modulus Algorithm

After the implementation, in accordance with the respective section in this document one will notice:

- the resultant beamformer will depend on $\mathbf{w}_{init}$, on μ (the step size) and on the number of samples
- Depends on the number of samples because it needs enough iterations to converge the estimations into modulus 1; The amount of samples it takes to converge depends also on the step size: it converges faster for a bigger step size. However, it may not converge if the step size is too big;
-

11 Extra Slides

11.1 Channel Estimation and Hybrid Precoding for mmWaves

Millimeter Waves need beamforming or else the additional attenuation wouldn't make the communication worth the higher frequencies and probably even worse than many other solutions that we've right now.

To do beamforming, we need an array of antennas, the more the better. And ideally, an array of radiofrequency chains coming directly from the the baseband processing that is done para a computer. However, this is tremendously expensive. What's done instead is using less radiofrequency chains with pre-defined precoders, like phase shifters. But one can't use just one RF chain and a lot of phase shifters because that would take away all the flexibility. An hybrid approach requires a new set of estimation algorithms to account these phase shifters and play with them to estimate and optimise the channel utilization.

11.2 UltraSound

There a tradeoff between frequency (or SNR) and resolution. Higher frequencies penetrate worse but provide a better resolution due to their smaller wavelength, one can better distinguish different scatterers. This is why resolution gets worse the deeper we try to see.

12 Evaluation

12.1 Relation of single values after sensors or samples change

Knowing that the sensors double or that the amount of samples double, what will be the effect on the single values? One my know! Changing the amount of sensors increases the size of the H matrix, increasing the number of samples increases the size of the S matrix. None of the sizes matters for the multiplication. Therefore, doubling either one will simply double the size of X.

How does doubling the size of X affects X Frobenius Norm? Know that the formula is given by: $\sum_{i,j} |x_{ij}|^2$, by increasing one of the sizes we simply double the size of X thus, multiplying that sum by 2. X can be decomposed in singular values and the norm of those matrices will be the norm of the central matrix because the other two are unitary, thus have norm one and the norm of the total is the multiplication of the norm of each one.

In this case, because we are increasing the dimensions by 2, the sum of the squares along the diagonal which is the Frobenius Norm of the matrix will have double the elements, thus the sum is doubled. Considering that they are equal (and there are 3 of them):
$3\sigma_{new}^2 = ||X_{new}||_F^2 = 2||X_{old}||_F^2 = 2 \cdot 3\sigma_{old}^2 \Leftrightarrow \sigma_{new} = \sqrt{2}\sigma_{old}$

12.2 Wiener and ZF receivers

The next question was about the Weiner and ZF receivers. By remembering what equations each one is trying to minimise, one can deduce them. By knowing S, knowing that the ZF attempts to minimise $||X - \hat{A}S||$, then $\hat{A} = XS^\dagger$. And thus, the beamformer is $W^{\mathrm{H}} = A^\dagger = (XS^\dagger)^dagger$.

Note that this is very different than the beamformer presented $W^H = SX^\dagger$. This beamformer comes from the minimisation of $||W^H X - S||$, thus being a Wiener Receiver.

12.3 Blind Estimation deduction and nullspaces relations

The last and the question I performed the worse was the one where a deduction for Blind Estimation was necessary.

For the symbols, V_n has the nullspace along its columns. As V_n^{H} is N-1 by N-1-(r) where r is the number of singular values due to noise, zero in its absence.

So, for the matrices to match, and because V_n^{H} has the row nullspace along its rows (the rightnullspace), $V_n^H X^H = 0 \Leftrightarrow XV_n = 0$.

Don't forget how to multiply things to obtain zero from the nullspaces

This makes sense because: the rows of V_n^{H} have the nullspace of rows of X, thus these rows need to by multiplied to the rows of X to get 0, therefore is needed to user X transposed (remember that rows of the left matrix multiply to columns of the right one). Likewise, when he transpose to V_n, now we have the nullspace along the columns. Then this matrix needs to be on the right to originate something that is a linear combination of the columns. By having the rows of X make this product, having X on the left, the "interception" will occur and because they are orthogonal, give 0.