Adaptive Array Signal Processing

[5SSC0]

Ruud van Sloun (responsible lecturer)
Iris Huijben (assistant)
Julian Merkofer (assistant)
Vincent van de Schaft (assistant)

Assignments Part 1

February 6, 2023

Contents

Chapter 1

1A: Adaptive algorithms

In this assignment optimum and adaptive filtering algorithms have to be implemented and evaluated. The objective is to evaluate and compare different facets of the performance of these adaptive filtering algorithms.
Fill in your answers on the predefined report 1A_form.docx or 1A_form.tex

---

1.1 Preview

1.1.1 Adaptive filtering

The adaptive filtering techniques covered in this course attempt to find a set of weights for a Finite Impulse Response (FIR) filter $\underline{\mathbf{w}} = (w_0, w_1)^t$ that minimizes an objective function J. The objective function is chosen such that it represents the squared difference between a filtered version of an input signal $x[k]$ and a given reference signal $e[k]$, resulting in $J = E(r[k]^2)$ with $r[k] = e[k] - \underline{\mathbf{w}}^t[k]\underline{\mathbf{x}}[k]$. The final FIR filter parameters will therefore give the best possible estimate of the reference signal based on $x[k]$.

1.1.2 Known statistics — Wiener/SGD/Newton

When the signal statistics are known, i.e., we collect the auto correlation of $x[k]$ in matrix $\mathbf{R}_x$ and the cross correlation between $x[k]$ and $e[k]$ in the vector $\underline{\mathbf{r}}_{ex}$, we can easily solve the optimization problem. As deducted in the course, we can compute these optimal weight by $\underline{\mathbf{w}}_0 = \mathbf{R}_x^{-1}\underline{\mathbf{r}}_{ex}$. This solution is also referred to as the **Wiener filter**.

Finding the inverse of $\mathbf{R}_x$ can be computationally expensive and we therefore require a less complex way to find $\underline{\mathbf{w}}_0$. The **Steepest Gradient Descent (SGD)** algorithm approaches $\underline{\mathbf{w}}_0$ by updating in the negative gradient direction: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{ex} - \mathbf{R}_x\underline{\mathbf{w}}[k])$, with α determining the rate of convergence. A drawback of the SGD algorithm is that the filter weights do not converge uniformly. The eigenvalue spread of auto correlation matrix $\mathbf{R}_x$ influences the rate of convergence of individual weights.

The Newton algorithm improves upon the SGD algorithm by undoing the coloration of $\mathbf{R}_x$. Newton's method applied to our case results in the following update rule: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\mathbf{R}_x^{-1}(\underline{\mathbf{r}}_{ex} - \mathbf{R}_x\underline{\mathbf{w}}[k])$. Obviously, both SGD and Newton's method are not always directly applicable in practice, because we may not know the signal statistics.

1.1.3 Unknown statistics — LS/LMS/RLS/FDAF

To overcome the practical problems of SGD and Newton's algorithm, we need a way to estimate the signal statistics from our input observations. Ideally, we would compute $\bar{\mathbf{R}}_x$ and $\bar{\underline{\mathbf{r}}}_{ex}$ of stationary signals as follows:

$$\bar{\mathbf{R}}_x = \lim_{L\to\infty} \frac{1}{L} \sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]\underline{\mathbf{x}}^t[k-i] \tag{1.1}$$

$$\bar{\underline{\mathbf{r}}}_{ex} = \lim_{L\to\infty} \frac{1}{L} \sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]e[k-i] \tag{1.2}$$

The Least Squares (LS) solution computes the Wiener filter based on the above estimates of the signal statistics. To avoid summing a large number of elements, several algorithms exist that estimate the statistics using less complex methods.

First of all, the Least Mean Squares (LMS) uses instantaneous estimates of the statistics given by: $\bar{\mathbf{R}}_x = \underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]$ and $\bar{\underline{\mathbf{r}}}_{ex} = \underline{\mathbf{x}}[k]e[k]$. To obtain $\underline{\mathbf{w}}_0$, the SGD update is applied at each iteration. This results in:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\underline{\mathbf{x}}[k]r[k] \tag{1.3}$$

Normalized Least Mean Squares (NLMS) extends LMS by using a normalized input. An estimate of the input signal power is maintained in $\hat{\sigma}_x^2$. Two example algorithms for updating the estimated signal power are given in the course. The NLMS update rule becomes:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \frac{2\alpha}{\hat{\sigma}_x^2}\underline{\mathbf{x}}[k]r[k] \tag{1.4}$$

normized

The Frequency Domain Adaptive Filter (FDAF) algorithm extends upon LMS by performing the LMS adaptive algorithm in the frequency domain. The filter w is preceded by a Fourier transform. As a results, the final filter weights will converge to $\underline{\mathbf{w}}_0$ combined with an inverse Fourier transform. Furthermore, FDAF uses the Newton update rule instead of SGD. For that purpose it maintains an estimate of the power spectral density for each frequency bin on the diagonal of the matrix $\mathbf{P}$. Recall that the PSD is a Fourier pair together with the autocorrelation function. Therefore, the multiplication with $\mathbf{R}_x^{-1}$ required by the Newton algorithm corresponds to a multiplication with $\mathbf{P}^{-1}$. This is a simple operation considering that $\mathbf{P}$ is a diagonal matrix. This results in the following update rule for FDAF. Refer to the course for a method to estimate $\mathbf{P}$.

$$\underline{\mathbf{W}}[k+1] = \underline{\mathbf{W}}[k] + \frac{2\alpha}{N}\mathbf{P}^{-1}\mathbf{X}^*[k]r[k] \tag{1.5}$$

Finally, the Recursive Least Squares (RLS) algorithm is a practical version of the LS algorithm. As opposed to an infinite window, RLS uses an exponentially decreasing sample window characterized by the 'forget factor' γ. Instead of inverting $\bar{\mathbf{R}}_x$, RLS maintains a recursive estimate of its inverse $\bar{\mathbf{R}}_x^{-1}[k]$. Combined with the estimated $\bar{\underline{\mathbf{r}}}_{ex}[k]$, the final filter weights are given by $\underline{\mathbf{w}}[k] = \bar{\mathbf{R}}_x^{-1}[k]\bar{\underline{\mathbf{r}}}_{ex}[k]$. The recursive update rule for the estimates of the signal statistics is given by:

$$\underline{\mathbf{g}}[k+1] = \frac{\bar{\mathbf{R}}_x^{-1}[k]x[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\bar{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \tag{1.6}$$

$$\bar{\mathbf{R}}_x^{-1}[k+1] = \gamma^{-2}(\bar{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\underline{\mathbf{x}}^t[k+1]\bar{\mathbf{R}}_x^{-1}[k]) \tag{1.7}$$

$$\bar{\underline{\mathbf{r}}}_{ex}[k+1] = \gamma^2\bar{\underline{\mathbf{r}}}_{ex}[k] + \underline{\mathbf{x}}[k+1]e[k+1] \tag{1.8}$$

1.2 Scenario 1: Known statistics

Consider the general filter optimization scheme as depicted in Fig. 1.1. In this part of the

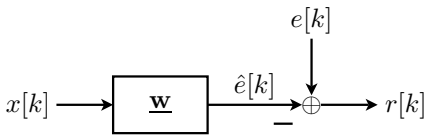

Figure 1.1: Part 1A, Scenario 1

assignment you have to calculate and design a two-tap FIR filter with $\underline{\mathbf{w}} = (w_0, w_1)^t$, by minimizing the MSE $J = E\{r^2[k]\}$. The following statistics are given:

$$\mathbf{R}_x = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \; ; \; \underline{\mathbf{r}}_{ex} = \begin{pmatrix} 0 \\ 3 \end{pmatrix}.$$

1.2.1 Wiener Filter

Fill in your answers on the predefined report 1A_form.docx and do not forget to include relevant calculations and figures in this predefined report!

a) Calculate by hand an expression for the MSE $J = E\{r^2[k]\}$ as a function of the weight vector $\underline{\mathbf{w}}$. Next, find the filter weights vector $\underline{\mathbf{w}}_o$ that minimize the MSE J.

b) Find an expression for the correlation $\underline{\mathbf{r}}_{xr}$ between $\underline{\mathbf{x}}[k]$ and $r[k]$. What should $\underline{\mathbf{r}}_{xr}$ be when $\underline{\mathbf{w}}$ equals the optimal Wiener filter $\underline{\mathbf{w}}_o$? What does this mean?

c) In practical cases, $\mathbf{R}_x$ and $\underline{\mathbf{r}}_{ex}$ may be unknown. How can you estimate these statistics from a set of observations $\underline{\mathbf{x}}[k]$ and $e[k]$? Which tradeoff do you have to make?

1.2.2 Steepest Gradient Descent

Computing the optimal Wiener filter involves inverting $\mathbf{R}_x$. To avoid the matrix inversion, one can use an iterative approach. The Steepest Gradient Descent (SGD) algorithm is such an algorithm.

d) When does the SGD algorithm reach steady-state? In other words, when the SGD reaches steady-state, what is the value of $\underline{\mathbf{w}}[k]$? Show this.

e) Calculate the range of α which makes the SGD algorithm stable.

f) Implement the filter update rule of the SGD algorithm in MATLAB. Take the following steps to do so.

- Extract the files in `Matlabs_1A.zip`. The file `Adaptive_filter.m` contains the class definition of an adaptive filter object. The hidden MATLAB programme `generate_input.p` takes care of generating `nsample` data samples of both $\underline{\mathbf{x}}[k]$ and $e[k]$. `1A.m` calls `generate_input.p` and performs `nsample` iterations of the adaptive filter algorithm. Additionally, `1A.m` plots the trajectory of the filter coefficients over as they evolve, together with a contour plot of the objective function J.

- Implement the SGD update rule in `update_filter.m` and run `1A.m`. Assume the same values for $\mathbf{R}_x$ and $\underline{\mathbf{r}}_{ex}$ as in Section 1.2.1. Use a value for α (the third argument of the constructor of the `adaptive_filter` object) that that ensures a smooth and stable convergence of $\underline{\mathbf{w}}$. Comment on the convergence of the two filter coefficients w_0 and w_1.

1.2.3 Newton's Method

Input coloration influences the performance of the SGD algorithm. To solve this issue, the Newton algorithm performs pre-whitening of the input signals.

g) Show that Newton's method makes all filter weights converge at the same rate.

h) For which α is Newton's method stable?

i) Implement the Newton filter update rule in `update_filter.m`. Again use an α that gives smooth convergence. Modify `1A.m` such that the filter uses the Newton update rule. Again run `1A.m` and comment on the convergence of the filter parameters.

1.3 Scenario 2: Unknown Statistics

Both the SGD and Newton method assume that $\mathbf{R}_x$ and $\underline{\mathbf{r}}_{ex}$ are known. However, in practical cases these are often unknown. The Least Mean Squares (LMS) and the Normalized LMS (NLMS) algorithms use an instantaneous estimate of the signal statistics.

1.3.1 LMS and NLMS

a) Implement the LMS filter update rule in the `update_filter.m`. Modify `1A.m` such that is uses the LMS update rule. Again, use a filter length of 2 and an appropriate α. What trade-off do you make when choosing α? Include the resulting plot and the most relevant line(s) in your answer form.

b) Repeat Scenario 2 a) for Normalized LMS. If you need to add extra state variables to the `adaptive_filter` object, simply add them as class properties in `adaptive_filter.m`. You can access them using `filter.myvar`, where filter is an instance of the `adaptive_filter` class.
Describe roughly the influence of the normalization factor.

1.3.2 RLS and FDAF

The RLS and FDAF algorithms both attempt to whiten their input using estimates of the signal statistics.

c) Explain in your own words how RLS and FDAF are related to the previous adaptive algorithms (LS/SGD/Newton/LMS/NLMS). How do RLS and FDAF solve the practical issues of their corresponding algorithm?

d) Implement the RLS update rule in MATLAB. If necessary, add extra state variables to the `adaptive_filter` class. Use $\gamma = 1 - 10^{-4}$ and include the resulting plot in your answer form. How does γ influence the convergence behavior?

e) Implement the FDAF update rule in MATLAB. Again, use appropriate values for α and β.

Hint 1: use `dftmtx` to create $\mathbf{F}$.

Hint 2: $\underline{\mathbf{r}}[k]$ is calculated as $\underline{\mathbf{w}}^t[k]\underline{\mathbf{x}}[k]$. Therefore, you should update the filter weights in $\underline{\mathbf{W}}[k]$ and compute $\underline{\mathbf{w}}[k] = \mathbf{F}\underline{\mathbf{W}}[k]$ to obtain a correct result.

f) Compare the advantages and disadvantages of LMS, NLMS, RLS and FDAF in terms of computational complexity and accuracy. Try to rank them (#1 is the best, #4 the worst). Give a brief explanation.

Some useful hints about working with the FDAF algorithm in Matlab:

- The main idea behind the assignment about the FDAF algorithm is to show that this adaptive algorithm can cope with colored input signals in a similar way as e.g. the RLS algorithm. The RLS algorithm uses the inverse of the autocorrelation matrix, while the FDAF uses a power normalization for each separate frequency bin. The result of the FDAF update algorithm is a weight vector $\underline{\mathbf{W}}[k]$. As denoted in the slides, this weight vector is the weight vector representation in the frequency domain. On the other hand, the calculation of the adaptive filter output $\hat{e}[k]$ in the Matlab programme is made in such a way that the output is calculated as $\hat{e}[k] = \underline{\mathbf{w}}^t[k] \cdot \underline{\mathbf{x}}[k]$. Thus, in order to obtain a correct result in this assignment when working with the FDAF algorithm, you have to apply an (extra) DFT operation: $\underline{\mathbf{w}}[k] = \underline{\mathbf{W}}[k] \cdot \mathbf{F}$.
- In Matlab the DFT matrix can be obtained easily by using the command: `F = dftmtx(nw);` Notice that you can calculate this matrix once in the initialization function and then store it in the `state` struct, which makes the implementation faster.
- Overall, your function could look like the following:

```
function state = update_1As2(state, x, r)

F = dftmtx(state.nw); % nw points DFT matrix
state.x_hist = ... % Something to store history in a column vector
X_freq = F*state.x_hist; % Frequency domain representation of x

w_new = state.w_old + f(X_freq, r); % Implement FDAF update f

state.w_old = w_new; % Store new values as old values for later
state.w = w_new*F; % Make sure that w*x => w*F*x = w*X
end
```

Chapter 2

1B: Array processing (beamformer design)

In this assignment narrowband and wideband beamformers have to be implemented and evaluated. The objective is to develop useful tools for evaluating the performance of beamformers with different configurations and objectives.

2.1 Preview

2.1.1 Beampattern (BP)

A standard tool for analysing the performance of a beamformer is the response for a given weight vector $\underline{\mathbf{w}}$ as a function of angle θ, known as beam response or array response. This angular response $r(\theta) = \underline{\mathbf{w}}^h \cdot \underline{\mathbf{a}}(\theta)$, is computed by applying the beamformer, with fixed $\underline{\mathbf{w}}$, to a set of array response vectors $\underline{\mathbf{a}}(\theta)$ from all possible angles, that is, $-90^o \leq \theta \leq 90^o$. Typically, in evaluating a beamformer, we look at the quantity $B(\theta) = \frac{\mathbf{1}}{J^2}|r(\theta)|^2$, which is known as the beampattern (BP) (in a dB plot we usually normalize the plot in such a way that the maximum is at 0 dB). A general expression for the steering vector is $\underline{\mathbf{a}}(\theta) = \mathrm{e}^{-\mathrm{j}\omega \cdot \frac{\mathbf{v}^t}{c} \cdot \underline{\mathbf{p}}}$, with c the speed of the wave in the medium (for sound in air $c \approx 340$ [m/sec]), $\omega = 2\pi f = 2\pi \frac{c}{\lambda}$ and wavelength λ. Furthermore $\underline{\mathbf{v}}$ is the normalized direction vector, thus $|\underline{\mathbf{v}}| = 1$, and $\underline{\mathbf{p}}$ denotes the sensor position vector.

2.1.2 Data Independent Beam Forming / Beam Steering

The BP is the spatial frequency response of a given beamformer, usually with a weight vector $\underline{\mathbf{w}} = \underline{\mathbf{1}}$. By choosing different weight vectors we can steer the beam. The beamforming weight vector that phase aligns a signal from direction θ_0 at the different array elements is the steering vector, which is simply the array response vector in that direction. The steering vector beamformer is also known as the spatial matched filter since the steering vector is matched to the array response of the signals impinging on the array from an angle θ_0. As a result θ_0 is known as the look direction. The use of the spatial matched filter is commonly referred to as conventional beamforming. The sidelobe levels of a beam can be further reduced by tapering the magnitude of the spatial matched filter. To this end, we employ a tapering vector $\underline{\mathbf{t}}$ that is applied to the spatial matched filter to realize a low sidelobe level beamformer. We refer to this beamformer as the tapered beamformer. Another type

of beamsteering is constrained beamsteering (also known as null-steering). This allows you to put constraints on the array response, so you can suppress undesired signals and amplify desired signals at the same time.

2.2 Scenario 1: Narrowband beamformers

Consider the array processing scenario in a two dimensional space as is depicted in Fig. 2.1. The sensors are located as a $J = 4$ elements uniform linear array (ULA) on the x-axis with

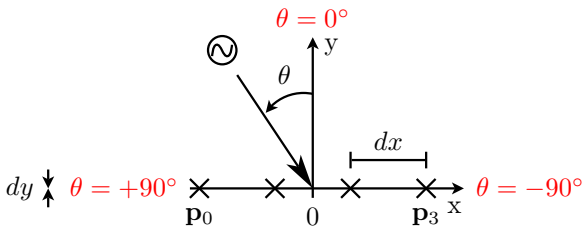

Figure 2.1: Part 1B, Scenario 1

a mutual spacing of $dx = 3.4$cm with the origin in the center of the array. The position of the i'th sensor is represented by the row vector $\vec{p}_i = [p_x, p_y]$. In the ULA in Fig. 2.1, the positions are given by $\vec{p}_i = [-\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{3}{2}]dx$. A desired narrowband source signal with frequency $f_d = 2.5$kHz is assumed to arrive at the sensor array from the far field under an angle of $\theta_d = 30°$. The medium in which the sound wave is traveling is air which results in a speed of sound of $c = 340$m/s.

2.2.1 Assignment 1B: scenario 1

Fill in your answers on the predefined report 1B_form.docx and do not forget to include relevant calculations and figures in this predefined report!

a) Find an expression for the array response vector $\underline{\mathbf{a}}(\theta)$ i.e. the response of each sensor as a function of θ in the frequency domain. Also derive an expression for the beampattern $B(\theta) = \frac{1}{J^2}|\underline{\mathbf{1}}^h \underline{\mathbf{a}}(\theta)|^2$.

b) Now derive an expression for $B(\theta)$ using only the two sensors p_1 and p_2. What is the highest source frequency f_{max} that does not cause spatial aliasing?

c) Calculate the beampattern in MATLAB and verify by plotting side-by-side with the results of a).

- Extract the files from `Matlabs_1B.zip` and open `Assign_1Bs1.m`. Set up the array parameters of the 4-sensor ULA at the start of the script.
- Implement the functions `array_response_vector.m` and `calc_nb_beampattern.m`. Verify them with your results at a) by filling in your expression for $B(\theta)$ in the `Assign_1Bs1.m` script and comparing the plots. Pay attention to both shape and amplitude of your result.
- Give the one line of code in which you define the elements of the array response vector in `array_response_vector.m`

d) Remove the `return` before `Assign 1Bs1 d)` in `Assign_1Bs1.m`. Implement the `beam_steering.m` function. Compute filter weights $\underline{\mathbf{w}}$ that give the array unit response at $\theta_d = 30°$. Plot

the result. Note that the Hermitian transpose is done using `A'` in MATLAB. `A.'` calculates the transpose without conjugating all elements.

e) Repeat d) with the ULA rotated by 90°. What differences do you notice, and how can this be explained from a geometrical point of view?

f) Rotate the ULA back again. An undesired source becomes active at $\theta_u = -60°$. Compute filter weights that will have unit response at θ_d and zero response at θ_u. Plot the steered response. What happens when you move θ_u closer to θ_d?

g) Implement the function `square_array.m` in the folder `+arrays` such that a four element rectangular array is obtained with the origin in the center of the array. Now use a 4-sensor rectangular array with $dx = 3.4cm$ and $dy = 1.7cm$, repeat f) for this array and compare the two steered responses.

h) The filter weights that you calculated are only valid for a small frequency band around $f_d = 2.5$kHz. Signals outside this frequency range will violate the constraints you posed on the steered response. Repeat g) with $f_d = 1$kHz and $f_d = 4$kHz and compare those three steered responses. Note that the filter weights should be kept equal whilst computing the responses.

i) According to what you have found in h), can you come up with an idea for steering wideband signals? Briefly describe your method.

2.3 Scenario 2: Delay and sum beamformers

In the previous exercises you calculated filter coefficients that steer the beam towards a source at a given DOA. These coefficient are generally only valid in a relatively small frequency range, i.e., they are samples of the wideband beamformer filter parameters. In general one has to divide the sensor signals into different frequency bands. Each frequency band is then filtered with its corresponding filter.

The Delay-and-Sum beamformer (DSB) is an exception on this rule. The DSB time-aligns the desired source in each of the sensor signals. This requires the sensor signals to be time-shifted by a delay filters d_n. The frequency-independent time-shift allows one to use one set of filters for the entire frequency range. The structure of a DSB is given in Fig. 2.2.

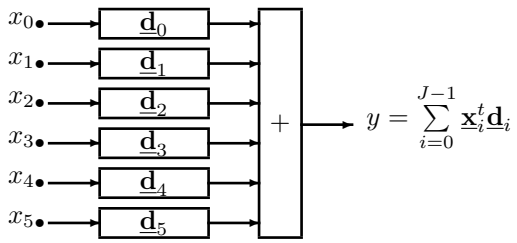

Figure 2.2: *General scheme of a delay and sum beamformer.*

In general the delays of the broadband setup can have any non-integer value. A way to cope with such non-integer delays is to use fractional delays. The general scheme of a fractional delay filter is given in Fig. 2.3. In this figure the fractional delay equals $z^{-i/L}$, with both i and L integer values. An efficient implementation of a fractional delay filter is given in the Matlab program `delay.m`. The function `h=delay(d, fd, L, dmax)` returns

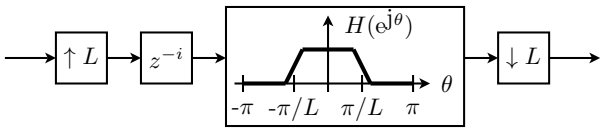

Figure 2.3: *General scheme of fractional delay .*

the fractional delay impulse response `h` of the delay $z^{-(d+\frac{fd}{L})}$, where the parameters `d`, `fd`, and `L` are all integers. The parameter `dmax` is the maximum delay in integer, i.e., `ceil(i/L)`, which is used to obtain delay filters with the same lengths and synchronization. You can find an example MATLAB code of using `delay.m` in Canvas.
In order to become familiar with the delay program you must report about the following steps:

a) If we want to delay a signal with delay τ [sec] (τ may not be an integer), we need to implement a delay filter with frequency response

$$D(e^{j\theta}) = e^{-j\tau\theta}. \tag{2.1}$$

Derive an analytical expression for the impulse response $d(t)$ of the delay filter, by computing the inverse Fourier-transform of $D(e^{j\theta})$.

b) Make in Matlab a figure containing the `plot`s the impulse responses $d_0(t)$ and $d_1(t)$ for the delays $\tau_0 = 3$ and $\tau_1 = 5.15$. Plot the sample numbers on the horizontal-axis and make sure that $t = 0$ corresponds to a time-shift of 0 seconds. Add your code to the report.

c) Generate the impulse responses $\hat{d}_0(t)$ and $\hat{d}_1(t)$, of the delays $\tau_0 = 3$ and $\tau_1 = 5.15$ by using the fractional delay program `delay.m`. In order to compare these results with the above derived theoretical results, use the Matlab function `stem` to plot these results into the previous figure. Explain how you cope with causality. Add your code to the report.

d) Implement the `Assign_1Bs2.m` script that generates impulse responses $d(t)$ that time-align the desired source in each sensor signal. Assume the same ULA array and DOA as in scenario 1 and a sample frequency of 25kHz. Filter each x_i in `Assign_1B_testsignals.mat` with its corresponding d_i and verify that all channels have been time-aligned. Add your code to the report.

Chapter 3

1C: Adaptive array processing

In this assignment narrowband direction of arrival (DOA) estimation algorithms and fully adaptive beamformers have to be implemented and evaluated. The objective is to develop useful tools and experience to solve such problems for more advanced configurations.

3.1 Preview

3.1.1 Signal model

We assume to have J sensors and P sources. For these far field sources the signal model is given as follows:

$$\vec{x}[k] = \sum_{i=1}^{P} \vec{a}(\theta_{s_i}) s_i[k] + \vec{n}[k]. \tag{3.1}$$

with length J column vectors $\vec{x}[k]$, $\vec{a}(\theta_{s_i})$ and $\vec{n}[k]$. The structure of the $J \times J$ correlation matrix has typically the following form:

$$\mathbf{R}_x = \mathbb{E}\{\vec{x}[k]\vec{x}^h[k]\} = \mathbf{A}\mathbf{R}_s\mathbf{A}^h + \mathbf{R}_n \tag{3.2}$$

where the $J \times P$ matrix $\mathbf{A}$ depends on steering vectors. Furthermore we have the $P \times P$ matrix $\mathbf{R}_s = \mathbb{E}\{\vec{s}[k]\vec{s}^h[k]\}$ the source correlation matrix and the $J \times J$ matrix $\mathbf{R}_n = \mathbb{E}\{\vec{n}[k]\vec{n}^h[k]\}$ is the noise correlation matrix, which are both assumed to be diagonal matrices.

3.1.2 Spatial power spectrum

The beampattern of a beamformer should not be confused with the steered response, which is the response on the array if we steer the array to all possible angles for a certain data set, i.e., observed signals $\vec{x}[k]$. Since the operation corresponds to measuring the power as a function of spatial frequency or angle, the steered response might be better defined as the spatial power spectrum:

$$P(\theta) = \frac{P_y(\theta)}{\vec{w}^h(\theta)\vec{w}(\theta)} = \frac{\mathbb{E}\{\left|\vec{w}^h(\theta)\vec{x}[k]\right|^2\}}{\vec{w}^h(\theta)\vec{w}(\theta)} = \frac{\vec{w}^h(\theta)\mathbf{R}_x\vec{w}(\theta)}{\vec{w}^h(\theta)\vec{w}(\theta)} \tag{3.3}$$

where the choice of the length J weight vector $\vec{w}(\theta)$ determines the type of spatial spectrum. In case of beamsteering the S/N ratio is maximized for $\vec{w}(\theta) \equiv \vec{a}(\theta)$ resulting in a spatial

power spectrum in which the power is maximized:

$$P_{PM}(\theta) = \frac{\vec{a}^h(\theta)\mathbf{R}_x\vec{a}(\theta)}{J}. \tag{3.4}$$

Thus a peak in $P_{PM}(\theta)$ is a DOA location.

3.1.3 Subspace techniques

High resolution spectral estimates can be obtained by using subspace techniques when applied on the appropriate signal model as in section 3.1.1. By applying subspace techniques we find the following decomposition of the sensor correlation matrix:

$$\mathbf{R}_x = \mathbf{U}_x\mathbf{\Lambda}_x\mathbf{U}_x^h = \mathbf{U}_s\mathbf{\Lambda}_{s+n}\mathbf{U}_s^h + \mathbf{U}_n\mathbf{\Lambda}_n\mathbf{U}_n^h \tag{3.5}$$

where $\mathbf{U}_x = [\vec{u}_1, \cdots, \vec{u}_J]$ and $\mathbf{\Lambda}_x = \text{diag}[\lambda_{x_1}, \cdots, \lambda_{x_J}]$. The signal subspace is given by:

$$\mathbf{U}_s = [\vec{u}_1, \cdots, \vec{u}_P] \quad \text{with} \quad \mathbf{\Lambda}_{s+n} = \text{diag}[\lambda_{s_1} + \sigma_n^2, \cdots, \lambda_{s_P} + \sigma_n^2] \tag{3.6}$$

and the noise subspace is given by:

$$\mathbf{U}_n = [\vec{u}_{P+1}, \cdots, \vec{u}_J] \quad \text{with} \quad \mathbf{\Lambda}_n = \text{diag}[\sigma_n^2, \cdots, \sigma_n^2] \tag{3.7}$$

An important property of these subspaces is that they are orthogonal with respect to each other, i.e.,

$$\mathbf{U}_s \perp \mathbf{U}_n \quad \Leftrightarrow \quad \mathbf{U}_s^h\mathbf{U}_n = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{U}_n^h\mathbf{U}_s = \mathbf{0} \tag{3.8}$$

This orthogonality property is specifically used in the following spectral-MUSIC algorithm:

a) Compute or estimate $\mathbf{R}_x$

b) Compute the EVD of $\mathbf{R}_x$ and split it in signal- and noise- subspace:

$$\mathbf{R}_x = \mathbf{U}_x\mathbf{\Lambda}_x\mathbf{U}_x^h = \mathbf{U}_s\mathbf{\Lambda}_{s+n}\mathbf{U}_s^h + \mathbf{U}_n\mathbf{\Lambda}_n\mathbf{U}_n^h$$

c) Compute the projection matrix

$$\mathbf{P}_n = \mathbf{U}_n\mathbf{U}_n^h$$

d) Evaluate the spectral MUSIC pseudo spectrum:

$$P_{SM}(\theta) = \frac{J}{\vec{a}^h(\theta)\mathbf{P}_n\vec{a}(\theta)}$$

e) DOA's of the P sources can be found from the P sharpest peaks in $P_{SM}(\theta)$.

3.2 Scenario 1: Maximizing steered beamformer power

Consider the array processing scenario in a two dimensional space which is now depicted in Fig. 3.1. The sensors are located as a $J = 6$ elements uniform linear array (ULA) on the x-axis with a mutual spacing of $dx = 3.4$cm with the origin in the center of the array. The position of the i'th sensor is represented by the row vector $\vec{p}_i = [p_x, p_y]$.

A desired narrowband source signal with center frequency $f_c = 2.5$kHz is assumed to arrive at the sensor array from the far field under an angle of θ_{s_1}. A second, interfering, source s_2 also has a center frequency of f_c but arrives from an angle θ_{s_2}. We know for

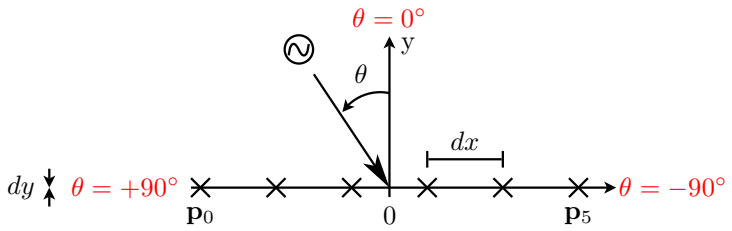

Figure 3.1: Part 1C, Scenario 1

the desired and interfering sources that the desired source is located more in front of the sensor array than the interfering source. Thus for DOA's $-90^0 \leq \theta \leq 90^0$ we know that $|\theta_{s_1}| < |\theta_{s_2}|$. The medium in which the sound waves are traveling is air which results in a speed of sound of $c = 340$m/s.

The observations are contaminated by spatially and temporally white Gaussian noise. We assume that the source variances $\sigma^2_{s_i}$ are the same for each source signal and the noise variances $\sigma^2_{n_j}$ are the same for each noise signal. Furthermore, we assume that all signals are uncorrelated with respect to each other.

The observations of a simulation of this DOA estimation scenario are available in the file `Observations_1C.mat`. The j'th signal in this file corresponds to the j'th sensor in the array. The sample rate is 10kHz, which is also specified in the mat file.

3.2.1 Assignment 1C: scenario 1

Fill in your answers on the predefined report 1C_form.docx and do not forget to include relevant calculations and figures in this predefined report!
The first algorithm is based on maximizing the power of a steered beamformer. If we apply a steered beamformer $\vec{w}(\theta)$ to the observations then we can identify the source angles of arrival as the maxima in the resulting spatial power spectrum.

a) Derive an expression of the sensor correlation matrix in terms of the signal variances, noise variances and array response vectors as functions of the angle of arrival parameter θ_{s_i}.

b) Apply beamsteering to the observations that are provided in the file `Observations_1C.mat`.

- Calculate the sensor correlation matrix R_x.
- Apply beamstering by computing $P(\theta) = \frac{\underline{\mathbf{a}}^h(\theta)\mathbf{R}_x\underline{\mathbf{a}}(\theta)}{J}$.

Plot the spatial spectrum for the full range of $-180° \leq \theta < 180°$ with a resolution of at least 0.5°. What are the angles of arrival for the desired and interfering sources?

Note: you may use the tools from the previous assignment 1Bs1 to calculate the desired beamformers.

c) Use only the four microphones that are closest to the origin by making the first and last filter coefficient equal to zero. Depict again the spatial spectrum for the full range of $-180° \leq \theta < 180°$. Discuss your results.

d) Using the equation you found at a), assuming 2 active sources and noise-free observations, we can derive $P(\theta_{s_1}) = J\sigma^2_{s_1} + \frac{1}{J}\underline{\mathbf{a}}^h(\theta_{s_1})\underline{\mathbf{a}}(\theta_{s_2})\sigma^2_{s_2}\underline{\mathbf{a}}^h(\theta_{s_2})\underline{\mathbf{a}}(\theta_{s_1})$. Do you think that the maximized steered response DOA estimate is accurate? Explain you answer.

3.3 Scenario 2: High resolution spectral estimation technique

One well known high resolution spectral estimation technique is the spectral-MUSIC algorithm. The spectral-MUSIC algorithm applies the steering vector for each angle of arrival to a projection matrix that is based on the noise subspace. The resulting pseudo spectrum contains very sharp peaks that represent the angles of arrival.

3.3.1 Assignment 1C: scenario 2

a) Use the subspace techniques to make an estimation of the noise variance.

b) Implement the spectral-MUSIC algorithm and apply it to the observations that are provided in the file `Observations_1C.mat`. Plot again the spatial spectrum for the full range of $-\pi \leq \theta < \pi$ with a resolution of at least $0.5°$. What are the angles of arrival for the desired and interfering sources? Compare your results with the results from Assignment 1C scenario 1.

c) What happens if you estimate the number of sources wrongly and then apply the spectral-MUSIC algorithm. For example, you estimate to receive either 1 or 3 sources, while 2 sources are present. Verify your expectations by applying these two scenarios to the provided observations.

3.4 Scenario 3: Adaptive minimum variance (Capon) beamforming

In this assignment you will exploit the minimum variance distortionless response (Capon) beamformer to adaptively suppress interference in delay-and-sum beamforming for a medical application: ultrasound image formation.

3.4.1 Assignment 1C: scenario 3

In ultrasound image formation, adequate beamforming ensures high image quality and therewith reliable diagnostics through active suppression of clutter (undesired interference). In this assignment we will make use of ultrasound recordings of synthetic point scatters. A high-quality beamformer should reveal clear and high-resolution points in the image.

Design a minimum-variance beamformer that follows the scheme in Fig. 3.2, in which the (real-valued) apodization weights w are updated adaptively based on the input x. Use the information in the paper "Adaptive Beamforming Applied to Medical.pdf" to see how you can design (and in a stable way) implement such a Capon beamformer for medical ultrasound imaging. Use the `image_reconstruction.m` and `beamformer.m` scripts as a backbone, and fill in the blanks. Try your beamformer on the `point_scat.mat` ultrasound dataset, and compare it to non-adaptive delay-and-sum beamforming. Include your relevant code, present the results and comment on your findings (1. the difference between using one of multiple angles; 2. the difference in terms of complexity and performance between adaptive vs non-adaptive).

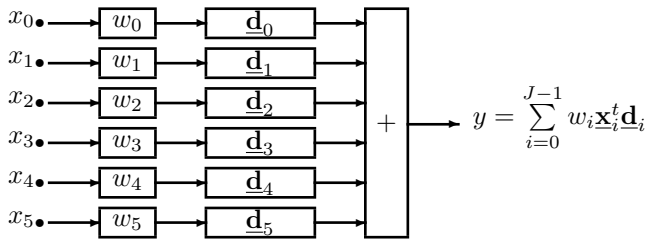

Figure 3.2: *Scheme of a delay and sum beamformer with adaptive weights w.*