Contents

Lab3: Beamformer design

In this assignment narrowband and multitone 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.

3.1 Preview

3.1.1 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 $\underline{\mathbf{w}}$ (usually $\underline{\mathbf{w}} = \underline{\mathbf{1}}$), 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 $B(\theta) = \frac{1}{J^2} \cdot |r(\theta)|^2$, which is known as the beampattern (BP). A general expression for the steering vector is $\underline{\mathbf{a}}(\theta) = \mathrm{e}^{-\mathrm{j}\omega \cdot \frac{\underline{\mathbf{v}}^t}{c} \cdot \underline{\mathbf{p}}}$, 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.

3.1.2 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 alligns 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.

3.2 Scenario 1: Narrowband beamformers

Consider the array processing scenario in a two dimensional space as is depicted in Fig. 1.1. The sensors are located as a $J = 4$ elements uniform linear array (ULA) on the x-axis with

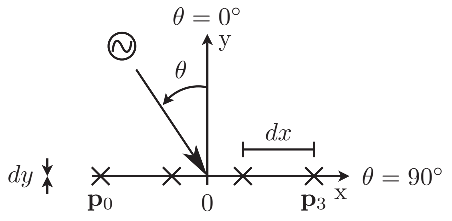

Figure 1.1: Part A2, Scenario 1

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 $\mathbf{p}_i = [p_x, p_y]$. 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.

3.2.1 Assignment A2: scenario 1

a) Calculate by hand the array response vector for the source coming from the $\theta_d = 30°$ direction.

b) The Matlab script `assign_A2s1.m` in `assign_A2s1.zip` helps evaluating the beamformer class that is contained in the class folder `@beamformer`. In the first part of the script the array configuration is setup. Choose the correct setting at the start of the script and verify that the sensors are located at the positions as is depicted in Fig. 1.1.

c) Implement the `array_response_vector.m` function in the folder `@beamformer` folder. Evaluate the `array_response_vector` function with the `assign_A2s1.m` script and verify the result with your answer at a).
NOTE: Don't forget to remove the `return` *command in the* `assign_A2s1.m`

d) Calculate by hand the narrowband beampattern of the beamformer in case all weights are chosen as $1/J$. Plot the result using Matlab.

e) Implement the `calc_nb_beampattern.m` function in the folder `@beamformer` folder. Evaluate the `calc_nb_beampattern` function with the `assign_A2s1.m` script and verify the result with your answer at d).

f) Implement the `beam_steering_nb.m` function in the folder `@beamformer` folder. Evaluate the `beam_steering_nb` function with the `assign_A2s1.m` script and verify the result by visual inspection.

g) An undesired source is now active, which is located at an angle of $\theta_u = -50°$. Design a beamformer that has a unit response at the desired direction $\theta_d = 30°$ and a zero response at the undesired response direction $\theta_u = -50°$.
NOTE: For this assignment you only need to change the `assign_A2s1.m` script at assignment g).

Now you have developed beamformering tools that can be used independently from the array configuration. This means that if we change the array configuration setting, then the tools can be used again.

h) Make a copy of the `assign_A2s1.m` script with the name `assign_A2s1_h.m` and rotate the sensor array in this script by 90° by changing the array specifications. Evaluate each sub-task in the new script and verify these results.

i) The function `ULA.m` in the folder `+arrays` generates an array with a uniform linear array (ULA) configuration. Implement the function `square_array.m` in the folder `+arrays` such that a four element square array is obtained with the origin in the center of the array. Verify the implementation with the plot function that belongs to the array class. Use the following parameters: $dx = 3.4$cm and $dy = 1.7$cm.

j) Make again a copy of the `assign_A2s1.m` script with the name `assign_A2s1_j.m` and change the array configuration on line 25 to `square_array`. Evaluate each sub-task in the new script and verify these results.

3.3 Scenario 2: Multi-tone beamformers

By evaluating narrow-band beamformers for a series of frequencies we obtain a multi-tone beamformer. This multi-tone beamformer is a simplified version of a wideband beamformer that needs to be used for example in an acoustic application, such as an hearing aid.

In this scenario we use the same setup as in Fig. 1.1 with the parameters as given in Sec. 1.2. A script with the name `Assign_A2s2.m` is available to guide you through the assignment.

a) Setup the array and beamformer settings in the script `Assign_A2s2.m`. For the multi-tone frequencies choose the range `mt_f = 100:100:5000`, i.e., from 100Hz to 5kHz in steps of 100Hz.

b) Implement and evaluate the `calc_mt_beampattern.m` in the `@beamformer` folder. The multi-tone weights are stored as column vectors per frequency in a matrix. The multi-tone beampattern has to be stored as row vectors per frequency in the property `b.mt_beampattern`.

c) Implement and evaluate the `beam_steering_mt.m` method in the `@beamformer` folder. The requirements are the same as in `Assign_A2s1 f)` but now the constraints have to hold for each frequency.

d) An undesired source is now active, which is located at an angle of $\theta_u = -50°$. Design a multi-tone beamformer that has a unit response at the desired direction $\theta_d = 30°$ and a zero response at the undesired response direction $\theta_u = -50°$. Evaluate the results.

e) Make a copy of the `assign_A2s2.m` script with the name `assign_A2s1_e.m` and change the array configuration on line 25 to `square_array`. Evaluate each sub-task in the new script and verify these results. Use the following parameters: $dx = 3.4$cm and $dy = 1.7$cm.