INF5410: Signal Processing in Space and Time

Mandatory Exercise 2

Spring semester 2012

Part A: High Resolution Beamforming

- This mandatory exercise is based on the examples shown in figures 4b and 5 in the paper: H. Krim, M. Viberg, *Two decades of array signal processing research – The parametric approach*, IEEE Signal Processing Magazine, pp. 67–94, July 1996. You may download the document from here: http://dx.doi.org/10.1109/79.526899 (You need to be on the University of Oslo network to get access.)
- Problem 1–5, and 7 are mandatory.
- All other problems are mandatory for PhD students, but voluntary for other students.
- The subject is on estimation of the spatial spectrum for an $M = 10$ element uniform linear array with half-wavelength spacing. The input consists of two incoherent signals at 0 and -10 degrees in additive spatially white noise. The signal to noise ratio for both sources is 0 dB, and $N = 100$ samples are available of the input. The signal and noise model is described on page 73 of the paper.
- The MATLAB code for the generation of the input is found at: http://www.ifi.uio.no/~inf5410/2005V/proj2b.m

1) Estimate the spatial correlation matrix. Estimate and plot the spatial correlation.
2) Estimate the spatial spectrum using the conventional method (Figure 4b). Discuss why the sources are not separated.
3) Estimate the spatial spectrum for the same signal using the minimum variance beamformer (Capon's beamformer) (Figure 5). Discuss the differences from the conventional beamformer.
4) Plot the distribution of the eigenvalues of the correlation matrix and explain it on the basis of the signal and noise model.
5) Estimate the spectrum using the MUSIC algorithm (figure 5) assuming that the number of signals is known. Discuss the differences from the previous estimates.
6) Estimate the spatial spectrum by the eigenvector method (Johnson & Dudgeon eq. 7.8a). Discuss the differences from the MUSIC beamformer.
7) *Incorrect estimate of the number of sources*
Estimate the spatial spectrum with the MUSIC method (and eigenvector method)

when the number of signals is incorrectly estimated. Let the estimate of the number of signals be $0, 1$, and 3. Discuss the differences between the estimates for the various cases. (Which spatial spectrum estimator is the eigenvector method equivalent to when 0 signals are assumed to be present?)

8) *Sparse array and adaptive beamforming*
Analyze the best and the worst 6 element symmetrical thinned arrays that were found in mandatory exercise 1, in the same signal scenario. Compute the Co-arrays. Estimate the spatial correlation matrix and spatial spectrums using conventional and adaptive beamformers. Discuss the results.

9) *Coherent sources*
Modify the signal generator so that it generates coherent signals instead. Find the properties of the previous beamformers for coherent signals (You may have to change the angles of incidence also). Implement the various forms of averaging of the correlation estimate and see if this gives the methods a better ability to handle coherence.

Part B: Beamforming of data from a microphone array

Background information for Part B

Here you get access to a set of data that were recorded by an array of microphones. The data and the helper-functions you need may be downloaded from this location:
https://www.uio.no/studier/emner/matnat/ifi/INF5410/v11/squarehead_oblig_data/secret_sound_exercise.zip
This zip archive contains the following files: `secret_sound.mat, make_freq_axis.m, timeshift_signals.m, filter6db.m`

You are provided with the following information about the experimental situation:

- The array of microphones was manufactured by *Squarehead Technology* for conference use. See http://www.squarehead.no/Conference.html for a system description.
- There are 3 sound sources in the room. You are provided with a data file with the received time-signals on each microphone.
- The data, as well as information about the array element coordinates are stored in the file `secret_sound.mat`. After loading this into the MATLAB memory [`load('secret_sound.mat')`], you may list the variables it contains:

```
>> whos
  Name          Size                    Bytes  Class     Attributes
  Xcoord      275x1                      2200  double
  Ycoord      275x1                      2200  double
  data        275x88201              97021100  single
  fs            1x1                         8  double
```

  - In order to get the position $\vec{x}_m = (x_m, y_m, z_m)$ of each element m in meters, you use the variables `Xcoord, Ycoord, Zcoord`.
  - The sampling frequency f_s is stored in Hz in the variable `fs`.
  - Each row m of the matrix `data` consists of the sound that was recorded by microphone m.

- You may assume the speed of sound to be 340 m/s.
- To let MATLAB output sound from microphone #1 to the loudspeaker, using the sampling frequency `fs`, type `sound(double(data(1,:)), fs)`. Make sure that the output data is normalized so that the amplitude stays within $A \in [-1, 1]$. The conversion from single-precision to double-precision using `double` must be applied since `sound` doesn't accept single-precision input.

Exercises for Part B

1) **The array geometry**

   - Plot the array element positions. Use the command `axis equal; axis tight` in order to adjust the x and y scales.
   - How do the array elements seem to be arranged?
   - For a frequency of $f = 3$ kHz, the sources may mainly be considered to be the near-field of the array aperture. Motivate this by calculation of the approximate far-field limit.

2) **Sound from a single channel**

   - Play the sound from channel #1 by use of the MATLAB command `sound`. This command works only on `double` type variables, so before playing the variable `foo`, convert it from single-precision into double-precision by applying the command `sound(double(foo), fs)`.
   - Describe briefly what you hear.

3) **Straightforward channel summation without beam steering**

   - Sum the signals of all elements without adding any element steering delays. Play the resulting sound by use of `sound`. Remember to normalize the sum signal to make sure the amplitude of the output signal to the loudspeaker is be within the interval $A \in [-1, 1]$.
   - May you hear any difference compared to your experience in 2)?

4) **Delay-and-sum beamformer implementation**
   Remember that focusing in the near-field (focusing for spherical waves) towards the position $\vec{x}$ may be done by application of the element delays

$$\Delta_m = \frac{r^\circ - r_m^\circ}{c}, \tag{1}$$

where $r^\circ \triangleq |\vec{x}^\circ|$ is the distance from the origin to the source assumed to be located the position $\vec{x}^\circ = (x^\circ, y^\circ, z^\circ)$ that we steer towards. Sensor element m is at the position $\vec{x}_m = (x_m, y_m, z_m)$, while $r_m^\circ \triangleq |\vec{x}^\circ - \vec{x}_m|$ is the distance from the position we steer towards to element m. The speed of sound is denoted c. The origin may of course be chosen at will, for example at $(x = 0, y = 0, z = 0)$. Then (1) becomes:

$$\Delta_m = -\frac{|\vec{x}^\circ - \vec{x}_m|}{c} = -\frac{\sqrt{(x^\circ - x_m)^2 + (y^\circ - y_m)^2 + (z^\circ - z_m)^2}}{c}. \tag{2}$$

Our main task here will be to use delay-and-sum beamforming to estimate the positions of the 3 sound sources in the room. This will be done through steering of the receiver beam to a number of positions in the room, and calculating the beamformer output for each of the positions. We will then assume that the sources are located at positions which generate a high beamformer output.

The directivity of the array increases with increasing frequency. We will prefer to do the source localization by selecting frequencies around $f_0 = 3$ kHz, with a 20% two-sided bandwidth. This may be done by use of the provided MATLAB function `filter6db`.

(a) Use the function `[insignal, INSIGNAL, f] = filter6db(f0, relwidth, fs, insignal)` to filter all channels in the variable `data`. Name the resulting filtered data matrix `data_filt`. Plot the frequency spectrum of the first channel `data(1,:)` as a function of frequency. Plot the frequency spectrum of the `data(1,:)`, filtered around 2 kHz, with 20% bandwidth. The quantities you should plot as a function of frequency are thus:

- $20 \cdot \log_{10} |\mathcal{F}\{p_1(t)\}(f)|$, and
- $20 \cdot \log_{10} |\mathcal{F}\{p_{1,\mathrm{B}}(t)\}(f)|$,

for $p_1(t)$ being the signal at channel one, and $p_{1,\mathrm{B}}$ being the same signal but band-pass filtered. Plot only within the frequency interval $f \in [0, 4]$ kHz. Let the dynamic range be from 0 to 60 dB. The axis adjustments may be done by use of the command `axis([0 4 0 60])`. Make sure to scale the frequency axis to kHz.
Note that you also need to have the provided helper function `make_freq_axis` accessible.

(b) Test the provided function `timeshift_signals`, by plotting the signal at sensor 1 as a function of time t, and within the same plot show this signal but time-delayed by $\Delta t = \pm 1/4$ s.

(c) Write a function that performs delay-and-sum focusing. After calculation of the focusing delays Δ_m within the function, all delays should be adjusted by subtraction of the smallest delay to: $\hat{\Delta}_m = \Delta_m - \min(\Delta_m)$. Then apply $\hat{\Delta}_m$ to each of the elements before output of the data from your function. [This just corresponds to an adjustment of the phase center position, and does not change the focusing point. Imagine for example that you delay all element

signals equally by one second after focusing: the focusing point will still be the same.]

Your function should be named `delaySum`. It should have the following header:

```
function [data, delta_t] = delaySum(data, xm, ym, zm, x0, y0, z0, fs, c)
% Input
% -----
% data : Signals. Each row is data on a single channel
% xm, ym, zm : Vectors with x,y, and z positions of the sensors.
%              Length must be the same as the number of rows in data
% x0, y0, z0 : scalars with x,y, and z position we want to steer to
% fs : sampling frequency [Hz]
% c : speed of sound [m/s]
%
% Output
% ------
% data : time-delayed signal matrix
% delta_t : vector with the applied time-delays for each sensor
```

Hint: The time-delay calculation part of your `delaySum` function could *e.g.* be implemented through something like:
`delta_t = -sqrt((xm-x0).^2 + (ym-y0).^2 + (zm -z0).^2) / c;`
This constructs a delay vector that is of the same dimension as the equally-dimensioned input variables `xm, ym, zm`.

(d) Now we're ready to try to localize some sources. This will be done by analysis of the filtered data matrix `data_filt` around 3 kHz.

- We assume the sources to be located at $z^\circ = 0$, within the region $x^\circ \in [-5, 5]$ m, $y^\circ \in [-5, 5]$ m.
- Generate a vector `pos = -5:(1/2):5`, and construct a grid of "focusing test points" by use of `[X0, Y0] = meshgrid(pos, pos);`. The z position of the sources is assumed to be $z = 0$, so you may construct the height position vector as `Z0 = 0 * X0`. Depending on the efficiency of your computer system, you might change `1/2` into a lower value without getting a too long calculation time. The result shown in Fig. 1 employs a stepsize of `1/5`.
- In order to keep down the calculation time: do the processing based on for example only the last $2^{12} = 4096$ samples of the `data_filt` matrix. This may be done by applying *e.g.* `data_filt_red = data_filt(:, (end-4096+1):end)`.
- Then sweep your beam through all points described by `X0, Y0, Z0`, and calculate the Power of the beamformer output at each point: $P(\vec{x}^\circ) = \mathrm{E}\left\{|z(t)|^2\right\}(\vec{x}^\circ)$.
- Finally visualize this focus-point-dependent $P(\vec{x}^\circ)$ in dB by use of `imagesc`. In order to make the maximum dB value to be = 0, normalize your $P(\vec{x}^\circ)$

by its maximum value before applying $20\log_{10}$ on its absolute value before plotting. Accompany your `imagesc` plot with a `colorbar`.
Adjust your x and y scales to be equal by applying `axis equal; axis tight`. Adjust your coordinate system axis directions by typing `axis xy`.
Identify probable source location coordinates. You may get x, y coordinates of `S` points on an `imagesc` plot by use of the command `[a,b] = ginput(S)`. What coordinates do you assume the 3 sources to be located at?
Your plot should look something like the plot shown in Fig. 1. [However in your report, please provide the plot in color so we know that you have produced one yourself.]

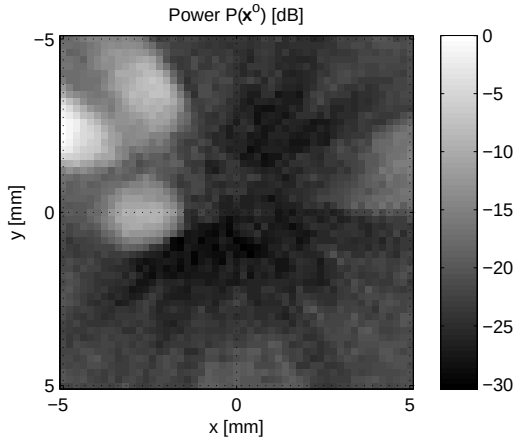

Figure 1: The beamformer output power as a function of steering position. Note that the axes might get mirrored / shifted depending on your coordinate system definition.

(e) Now you should focus the total un-filtered data onto your S identified source locations $\vec{x}_s^\circ,\ s = 0, \ldots, \vec{x}_{S-1}^\circ$.

- Calculate the beamformer output $z_s(t) = \sum_{m=0}^{M-1} y(t - \Delta_m(\vec{x}_s^\circ))$ for each of the S identified source locations $\vec{x}_s^\circ$.
- Listen to each of the $z_s(t)$ by use of `sound`. Remember to normalize each beamformer output in order to keep the amplitude within $[-1, 1]$. Describe what you hear from each source.

- Write a report with all plots, results and discussions. Convert your report to a single file in the PDF format.
- Name your report file `mandatory_2_LASTNAME.pdf`, where you replace `LASTNAME` with your last name. Include your MATLAB code printed as an appendix in the PDF report, and also as m-files.
- Name your MATLAB files following the pattern `mandatory_2_LASTNAME_MYFILETAG.m`, where you replace `LASTNAME` with your last name and `MYFILETAG` with whatever you prefer.
- Deliver your files to `svenpn@ifi.uio.no`, as a zip archive named `mandatory_2_LASTNAME.zip`, where you replace `LASTNAME` with your last name.

Please produce a tidy and structured, however not exhaustive, report: all plots must have correct axis scaling and naming, the important calculations must be included, the questions must be answered in the given order, etc.

Delivery deadline: Friday April 27 2012, at midday (12.00 noon).

All reports delivered after the deadline will be declined, unless you have explicitly asked for exception beforehand.