DoA-Estimation

Zike,Xu Zhiyi,Wang Dinghao,Chen

December 12, 2020

Contents

1 Narrowband Estimation

We have obtained a stimulation data with $J = 4$ sensors in *Observations_nb.mat*, and we want to estimate angles of the source with MUSIC algorithm.

In MUSIC, there're some basic arguments, J represents the number of sensors, and P represents the number of sources ($P < J$). The signal is given in the form of

$$s(t) = A(t)e^{j\omega_c t+\phi(t)} \tag{1}$$

when $A(t)$ is the envelop, $\phi(t)$ is the phase. In narrowband, they varies slow than $e^{j\omega_c t}$,

$$s(t-\tau) = A(t-\tau)e^{j\omega_c(t-\tau)}e^{j\phi(t-\tau)} \approx A(t)e^{j\omega_c(t-\tau)\phi(t)} = e^{-j\omega_c\tau} \cdot s(t) \tag{2}$$

thus, in narrowband, Time delay $\approx$ Phase shift.

Position J sensors on x-y axis, their relative position can be $p_i = (x, y)^t$. Consider just one far field signal passes the system with angle θ(with y axis), which can be expressed as $\underline{v} = (\sin\theta, \cos\theta)^t$. The time delay of every sensors is pv. Convert it into phase shift, its $e^{-j2\pi\omega_c\Delta t}$, denote it as $a(\omega, \theta)$. Thus, we can construct a matrix with $a(\omega, \theta)$ which can shift the original signal as θ changes.

$$s[k, \underline{p_i}] = s[k] \cdot e^{j\omega\tau_i} = s[k] \cdot a_i(\omega, \theta) \tag{3}$$

For narrowband signal, we can acquire its f_c(center frequency) in advance, so $a_i(\omega, \theta)$ can be written as $a_i(\theta)$.

We model the system as

$$\begin{gathered}\underline{x}[k] = (x_1[k], x_2[k], x_3[k], \cdots, x_J[k])^t \\ \underline{n}[k] = (n_1[k], n_2[k], n_3[k], \cdots, n_J[k])^t \\ \underline{a}(\theta) = (a_1(\theta), a_2(\theta), a_3(\theta), \cdots, a_J(\theta))^t\end{gathered}$$

in which

$$\underline{x}[k] = \underline{a}(\theta) \cdot s[k] + \underline{n}[k] \tag{4}$$

for multiple sound sources, add them toghther to get

$$x_i[k] = \sum_p a_i(\theta_p) \cdot s_p[k] + n_i[k] \Rightarrow \underline{x}[k] = A \cdot \underline{s}[k] + \underline{n}[k] \tag{5}$$

A is a $J \times P$ matrix, and $\underline{s}[k]$ is the column vector consists of different sound sources.

Denoted the covariance matrix as R_x,

$$R_x = \mathbb{E}\{\underline{x}[k]\underline{x}^h[k]\} \tag{6}$$

expand it with $\underline{x}[k] = A \cdot \underline{s}[k] + \underline{n}[k]$,

$$\mathbb{E}\{\underline{x}[k]\underline{x}^h[k]\} = \mathbb{E}\{(A \cdot \underline{s}[k] + \underline{n}[k])(A \cdot \underline{s}[k] + \underline{n}[k])^h\} \Rightarrow A\mathbb{E}\{\underline{s}[k]\underline{s}^h[k]\}A^h + \mathbb{E}\{\underline{n}[k]\underline{n}^h[k]\} \tag{7}$$

$R_s = \mathbb{E}\{\underline{s}[k]\underline{s}^h[k]\}$ is a Hermite positive definite matrix, so

$$rank(R_s) = rank(A) = rank(AR_sA^h) = P_{source}$$

The rank of the matrix equals to its number of non-zero eigenvalues, so AR_sA^h contains $J-P$ zero eigenvalues.

Let $\underline{u_i}$ be the eigenvector corresponding to zero eigenvalues, so

$$\underline{u}_i^h AR_sA^h\underline{u}_i = \underline{0} \Rightarrow (A^h\underline{u}_i)^h R_s(A^h\underline{u}_i) = \underline{0} \Rightarrow A^h\underline{u}_i = \underline{0} \tag{8}$$

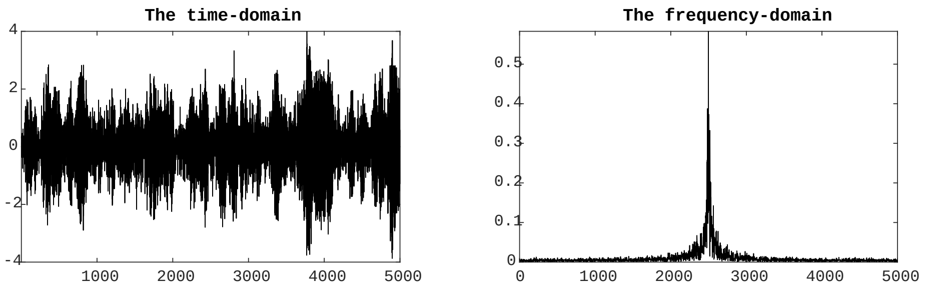
Which claim that the $J-P$ zero eigenvectors is orthogonal to those steering vectors.

We traverse θ to get the minimal $\sum_i \underline{a}^h(\theta_p)\underline{u}_i$, and define

$$P_{music}(\theta) = \frac{1}{\sum_{i=1}^{J-P}\left|\underline{a}^h(\theta_p)\underline{u}_i\right|^2} \tag{9}$$

and the maximum point is the direction of arrival(DoA).

1.1 Find f_c through frequency domain



We can figure out from the graph that $f_c = 2500$Hz.

1.2 Estimate the DoA

The MUSIC algorithm is implemented through MATLAB:

Listing 1: **narrowband.m**

```
clear all; close all;
%% Load data
load("..\data\Observations_nb.mat");
% load data
[Frame,nSensors] = size(X);
f_domain = (-Frame/2:Frame/2-1)*fs/Frame;
%% Plot waveform

figure
subplot(1, 2, 1);
plot(real(X(:, 1)), 'k', 'LineWidth', 0.5);
axis([-inf inf -4 4]);
title("The time-domain");
subplot(1, 2, 2);
plot(f_domain, abs(fftshift(fft(X(:, 1)))/Frame), 'k', 'LineWidth', 0.5);
```

```
title("The frequency-domain");
axis([0 5000 0 inf ]);

%% Array setup
% number of sensors
J = nSensors;
% inter-sensor distance in x direction (m)
dx = 3.4*10^-2;
% sensor distance in y direction (m)
dy = 0;
% sound velocity (m/s)
c = 340;
% number of sources
n_source = 2;
Index = linspace(0,J-1,J);
% sensor position
p = (-(J-1)/2 + Index.') * [dx dy];

%% Plot sensor positions
linspec = {'rx','MarkerSize',12,'LineWidth',2};
figure
plot(p (:,1), p (:,2), linspec {:});
title('Sensor positions');
xlabel('x position in meters');
ylabel('y position in meters');
disp('The four microphones are ready !');


%% DoA estimation (MUSIC)
% determine the angular resolution(deg)
stride = 0.5;
% grid
theta = -90:stride:90;
% center frequency (Hz)
f_c = 2500;
% autocorrelation estimate
R_x = X'*X/Frame;
% direction vector
v = [sin(theta*pi/180); -cos(theta*pi/180)];
% steer vector
a_theta = exp(-1i*2*pi*f_c*(p*v)./c);

% implement eigen-decomposition

[V, D] = eig(R_x);
eig_val = diag(D);
[eig_val, Idx] = sort(eig_val);
% noise subspace (columns are eigenvectors), size: J*(J-n_source)
Un = V(:, Idx(1:J-n_source));
% pseudo music power
```

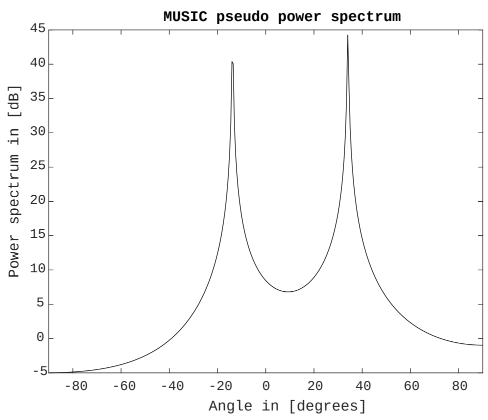
```
P_sm = 1./diag(a_theta'*(Un*Un')*a_theta);

%% Plot the MUSIC pseudo power spectrum
figure;
linspec = {'k-','LineWidth', 0.5};
plot(theta, 10*log10(abs(P_sm)), linspec{:});
title('MUSIC pseudo power spectrum')
xlabel('Angle in [degrees]');
ylabel('Power spectrum in [dB]');
xlim([-90,90]);

%% Find the local maximum and visualization
P_middle = abs(P_sm(2:end-1));
P_front = abs(P_sm(1:end-2));
P_back = abs(P_sm(3:end));
logic_front = (P_middle - P_front)>0;
logic_back = (P_middle - P_back)>0;
logic = logic_front & logic_back;
P_middle(~logic) = min(P_middle);
P_local = [abs(P_sm(1));P_middle;abs(P_sm(end))];
[~,doa_Idx] = maxk(P_local,n_source);
doa = theta(doa_Idx);
[~,minIdx] = min(abs(doa));
doa_source = doa(minIdx);
[~,maxIdx] = max(abs(doa));
interfer = doa(maxIdx);

disp(['The desired source DOA with MUSIC is: ',num2str(doa_source),' deg']);
disp(['The interfering DOA with MUSIC is: ',num2str(interfer),' deg']);
```

The result of the algorithm is:



Which claims that:

```
The desired source DOA with MUSIC is: −14 deg
The interfering DOA with MUSIC is: 34 deg
```

2 Broadband Estimation

Since broadband speech signal can be non-stationary, we need STFT(Short-time Fourier Transform) to keep information on frequency domain and time domain at the same time.

The ISSM(Incoherent Signal Subspace Method), which perform STFT on the original signal, and do MUSIC with its result in row. We've read the original paper of ISSM, and ... Whatever, here I'm going to prove the map between frequency and the index in FFT.

In MATLAB, function FFT will perform

$$Y[k] = \sum_{j=1}^{n} X[j] e^{-i2\pi(j-1)(k-1)/n} \tag{10}$$

Thus, in case that the original signal $X[j] = e^{i2\pi f \cdot (j-1)/f_s}$,

$$Y[k] = \sum_{j=1}^{n} e^{i2\pi(j-1)\left[\frac{f}{f_s} - \frac{k-1}{n}\right]}$$

The modulus of the result can reach its peak when k equals to some number. Let $u = \frac{f}{f_s} - \frac{k-1}{n}$, using Euler's formula to transform it into ($u \neq 0$)

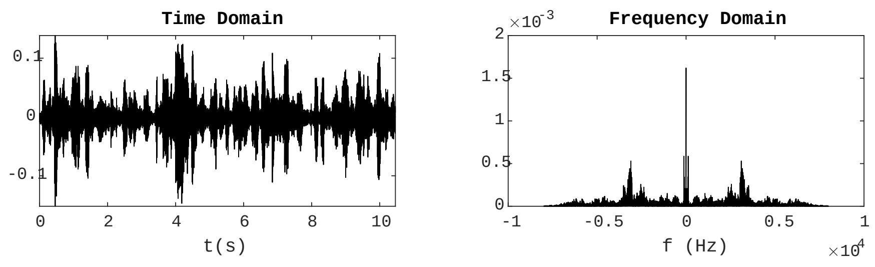
$$\mathrm{Re}\{Y[k]\} = \sum_{j=1}^{n} \cos[2\pi u \cdot (j-1)] = \frac{1}{2}(\csc(\pi \cdot u)\sin[(2n-1)\pi \cdot u] + 1)$$

$$\mathrm{Im}\{Y[k]\} = \sum_{j=1}^{n} \sin[2\pi u \cdot (j-1)] = \frac{1}{2}\csc(\pi \cdot u)(\cos(\pi \cdot u) - \cos[(2n-1)\pi \cdot u])$$

Consider the graph of csc function, and other sin or cos part limit it with an approximated 1 value, when u approach to 0, the modulus grows up quickly. And when $u = 0$, as $Y[]$

2.1 Plot the wave and frequency response

The figure:



Its easy to figure out that the frequency covers a wide span of frequency and can not find a f_c.

2.2 Perform STFT and MUSIC with spectral density matrix

The ISSM algorithm is implemented through MATLAB:

Listing 2: **broadband.m**

```
clear all;
close all;

%% Before STFT
load("..\data\Observation_wb.mat");
% load("data\Observations_nb.mat");
[Frame, ~] = size(X);

%% STFT
len = 1024;
inc = 1024;
nfft = len; % The smallest 2^n \ge len, to optimize FFT
[st_idx, ed_idx, fn] = separate(len, inc, Frame);

% STFT -> 4 sensors, value after FFT,
STFT = zeros([fn nfft 4]);
for i=1:fn
  STFT(i, :, :) = fft(X(st_idx(i):ed_idx(i), :), nfft);
end

% Perform MUSIC algorithm

% Initialize data
J = 4;
dx = 3.4*10^-2;
dy = 0;
c = 340; % Velocity of sound
Index = linspace(0,J-1,J);
p = (-(J-1)/2 + Index.') * [dx dy]; % Position vector
stride = 0.5;
theta = -90:stride:90;
v = [sin(theta*pi/180); -cos(theta*pi/180)];

P = zeros([180/stride+1 1]); % -90:stride:90
for i=1:ceil(nfft/2)

% P: index -> f
% $$\frac{(k - 1)f_s}{n}$$
f_c = (i - 1)*fs/nfft;
X_ = squeeze(STFT(:, i, :));

[Frame_, ~] = size(X_);

R_x = X_'*X_/Frame_;
a_theta = exp(-1i*2*pi*f_c*(p*v)./c); % steering vector

[V, D] = eig(R_x);
eig_val = diag(D);
[~, Idx] = sort(eig_val);
```

```
Un = V(:, Idx(1:J-2)); % noise subspace
P_sm = diag(a_theta'*(Un*Un')*a_theta);
P = P + P_sm;

end

P = 1./P;

% Find the local maximum;
P_middle = abs(P(2:end-1));
P_front = abs(P(1:end-2));
P_back = abs(P(3:end));
logic_front = (P_middle - P_front)>0;
logic_back = (P_middle - P_back)>0;
logic = logic_front & logic_back;
P_middle(~logic) = min(P_middle);
P_local = [abs(P(1));P_middle;abs(P(end))];
[~,doa_Idx] = maxk(P_local, 2);
doa = theta(doa_Idx);
[~,minIdx] = min(abs(doa));
source_1 = doa(minIdx);
[~,maxIdx] = max(abs(doa));
source_2 = doa(maxIdx);

disp(['The first source with MUSIC is: ',num2str(source_1),' deg']);
disp(['The second source with MUSIC is: ',num2str(source_2),' deg']);

% figure;
% linspec = {'b-','LineWidth',2};
% plot(theta, 10*log10(abs(P)), linspec{:});
% title('MUSIC pseudo power spectrum')
% xlabel('Angle in [degrees]');
% ylabel('Power spectrum in [dB]');
% xlim([-90,90])
%% Functions

function [ st_index, ed_index, fn ] = separate(len, inc, Frame)
  fn = floor((Frame-len)/inc + 1);
  st_index = (0:(fn-1))*inc + 1;
  ed_index = (0:(fn-1))*inc + len;
end
```