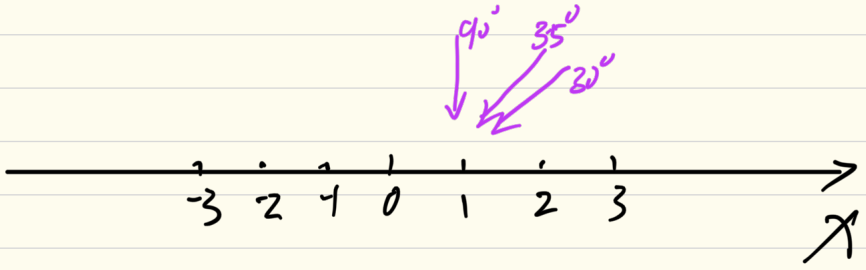
# Lab: AM1: Array communication processing

5 sensor location,

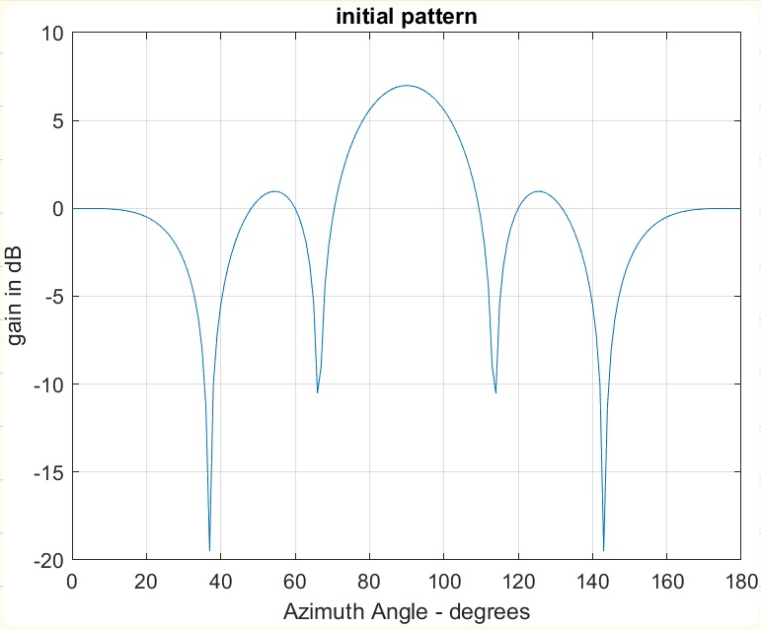
① 读music算法之前状.

$$array = \begin{bmatrix} -2 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

$$Direction: = \begin{bmatrix} 30 & 0 \\ 35 & 0 \\ 90 & 0 \end{bmatrix}$$




1. Pattern of the array,




The gain is max at 90° azimuth.

## 2. Theoretical Covariance matrix formation,

### 2.1 Manifold vector:

$$s(\theta,\phi) = \begin{bmatrix} e^{-jkr_1} \\ e^{-jkr_2} \\ \vdots \\ e^{-jkr_N} \end{bmatrix}, \quad k = \frac{2\pi}{\lambda}\begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi \end{bmatrix}$$

$r_N$: position of element

s：这是输出的**信号源位置向量**，又称为**阵列流形向量**（Array Manifold Vector）。矩阵 s 的每一列对应一个信号源的到达方向，每行对应一个阵元在该方向上的相位响应。

Array pattern:

gain ← $g(\theta,\phi) = w^H s(\theta,\phi)$

### 2.2 Covariance matrix for source signal

$R_{mm} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ (uncorrelated) power=1

### 2.3 Form of the theoretical covariance matrix — $R_{xx,theoretical}$

$x(t) = s\,m(t) + n(t)$

- $x(t)$: received signal
- $s$: manifold vector
- $m(t)$: source signal
- $n(t)$: noise

AC54.

$$R_{xx} = E[x(t) \cdot x^H(t)]$$

↓

expectation

$$= E \cdot [(sm(t) + n(t)) \cdot (sm(t) + n(t))^H]$$

$$= E \cdot [sm(t) s^H m^H(t)] + E \cdot [sm(t) \cdot n^H(t)] + E[n(t) \cdot s^H m^H(t)] + E[n(t) \cdot n^H(t)]$$

(the two middle terms $\to 0$)

Assume, signal and noise are uncorrelated.

$$= E[sm(t) s^H m^H(t)] + E[n(t) \cdot n^H(t)]$$

$$= R_m \cdot S \cdot S^H + \sigma^2 \cdot I_N$$

$R_m$: $E[m(t) \cdot m^H(t)]$

$I_N$: unit matrix

**信号特征分析：**

- 协方差矩阵包含了信号和噪声之间的统计关系，描述了阵列接收到的信号在不同传感器之间的相关性。
- 通过协方差矩阵，可以确定信号的特征，例如信号方向、功率等。

## 3. Practical covariance matrix

The practical received signal:

$$x = [x_1(t), x_2(t) \cdots x_L(t)]$$

$$R_{xx} = \frac{x \cdot x^H}{L}$$

## 5. Direction problem

$$R_{xx} = S \cdot R_{mm} \cdot S^H + \sigma^2 \cdot I_N$$

$$= R_{signal} + \sigma^2 \cdot I_N$$

$$R_{xx,theo} = V \cdot \Lambda \cdot V^H$$

$V$ → eigenvector matrix

$\Lambda$ → eigenvalue

$$\Lambda = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \ddots & & \\ & & & \lambda_i & \\ & & & & \ddots \\ & & & & & 0 \\ & & & & & & 0 \end{bmatrix}_{N \times N}$$

$N - M$ = number (noise)

↓

number of source signal

In Matlab:

Eigenvalue : $R_{xx\_inv}$ :

$$\begin{bmatrix} -0.0001 \\ 0.0001 \\ 0.0953 \\ 4.9681 \\ 4.9369 \end{bmatrix}$$

} noise: 0.0001 $\sigma^2 =$

3 sources

$R_{xx\_an}$

$$\begin{bmatrix} -0.0000 \\ 0.0000 \\ 0.0017 \\ 0.1072 \\ 4.3818 \end{bmatrix}$$

} noise: 0 $\sigma^2 =$

$R_{xx\_im}$

$$\begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.0019 \\ 0.2039 \\ 2.1784 \end{bmatrix}$$

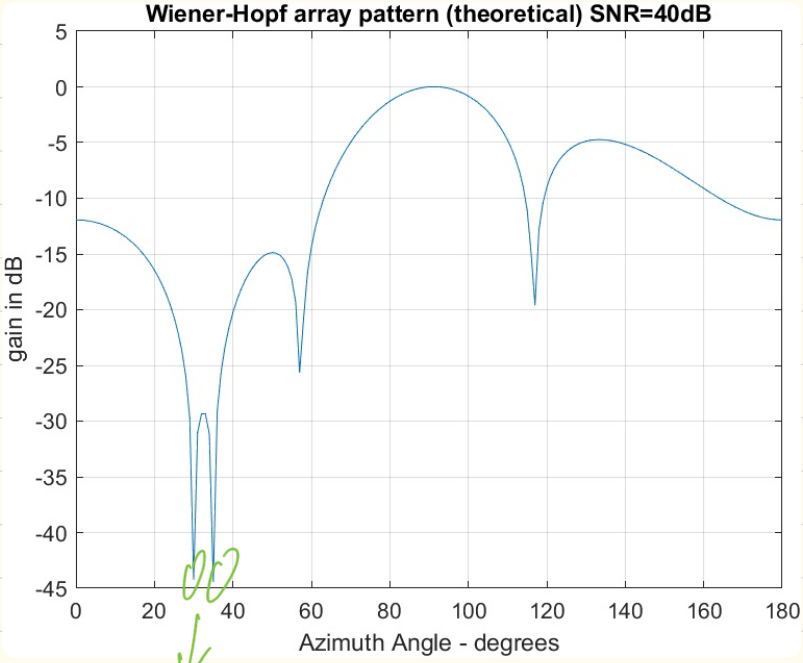
} noise: 0 $\sigma^2 =$

# 6. Estimation problem

The weight:

$$W = a \cdot R_{xx\text{-theo}}^{-1} \cdot S_d$$

$$= SPV(array, [90,0])$$




interferences.

**公式的物理含义**

最优权重向量的计算公式：

$$w_{\text{opt}} = aR_{xx,\text{theoretical}}^{-1}S_d$$

这实际上是**Wiener-Hopf解**，是用于实现最优波束成形的一种方式，目的是最大化目标方向的信号增益，同时抑制来自其他方向的干扰和噪声。其物理含义包括以下几点：

1. **信号增强**：
   - `Sd` 是目标方向信号的方向矢量，通过对阵列加权使得阵列对该方向上的信号具有最强的响应。
2. **干扰和噪声抑制**：
   - 协方差矩阵 `Rxx_theoretical` 包含了噪声和干扰的统计特性，对其求逆的目的是为了降低干扰和噪声的影响。通过这种方式，可以使加权向量在目标信号方向上达到最大增益，同时在干扰方向上最小化响应。
3. **增益因子** `a`：
   - 增益因子 `a` 用于调整整体的波束成形增益，它的值可以选择合适的常数，以确保输出功率达到预期。

7. Repeat 2, 4, 5.1, 6 with SNR = 10dB

$R_2$:

$R_4$:

$R_{5.1}$:

Eigenvalue; $R_{xx}$_theo:
0.1
0.1
0.1952
5.0680
10.0368
} noise
closed / hard to determine M.

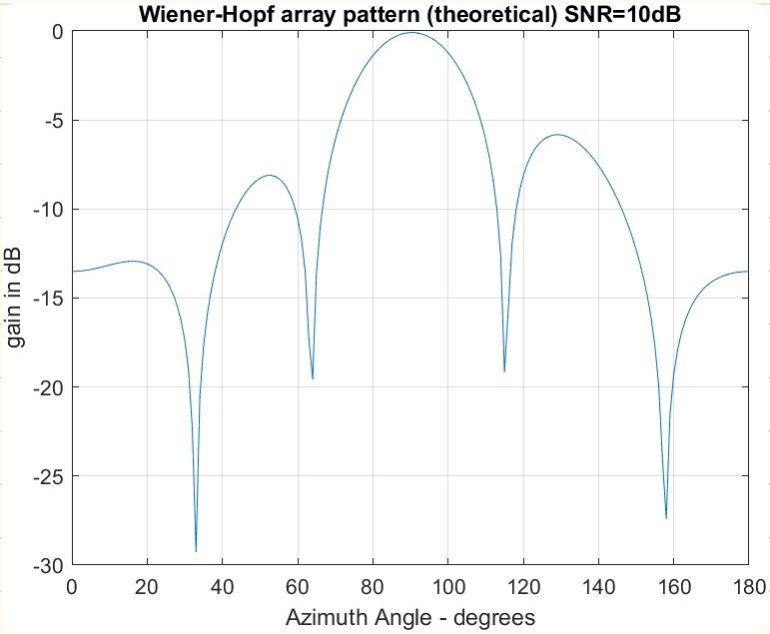
Akaike (AIC)

MDL

$R_{xx}$_an:
0
0
0.0017
0.1072
4.3818
} noise

$R_{xx}$_im:
0
0
0.0019
0.2039
2.1784
} noise

Q6:




Hard to find two interferences.
due to the low SNR

# Q9: Estimation problem — Music algorithm.

$$P_{MUSIC}(\theta) = \frac{1}{a^*(\theta)\cdot E_N \cdot E_N^* \cdot a(\theta)}$$

$a(\theta)$ : Array manifold vector

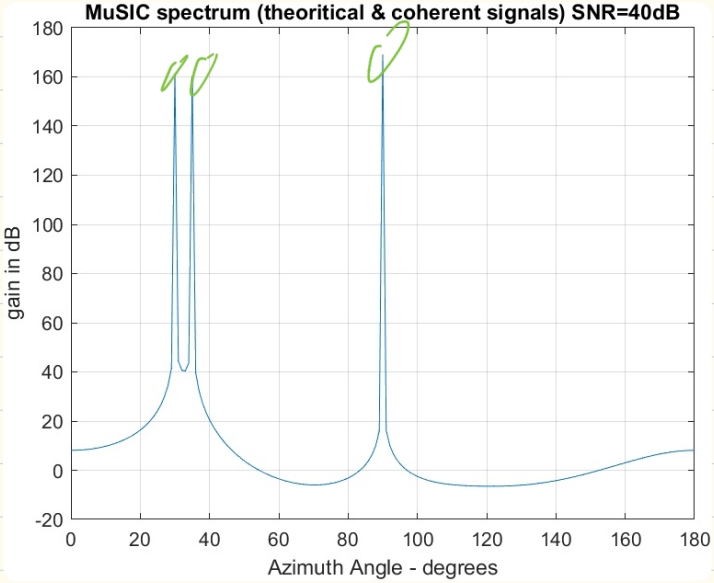
$a^*(\theta)$ → SV of each $\theta$

$E_N$ → noise eigenvector

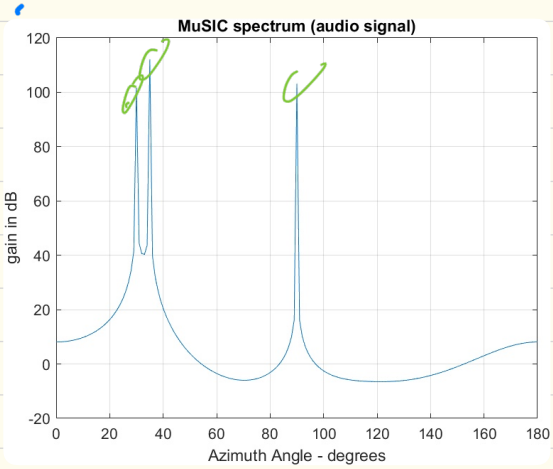
$E_N = E_n \cdot E_n^H$ → space of noise

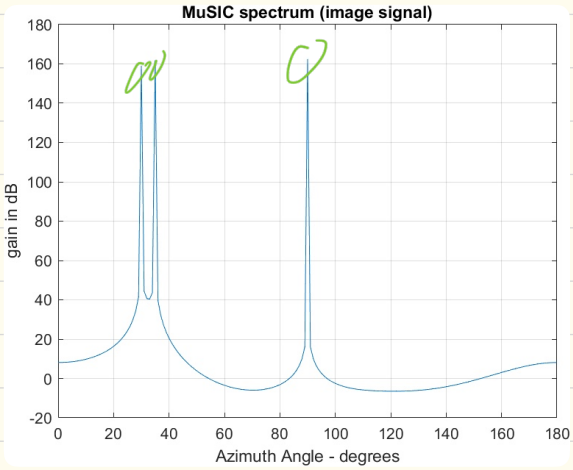
$[E, D] = \text{eig}(R_{xx})$

$$\therefore \xi(\varphi) = \frac{1}{S(\varphi)^H \cdot S(\varphi) \cdot P_{in}}$$



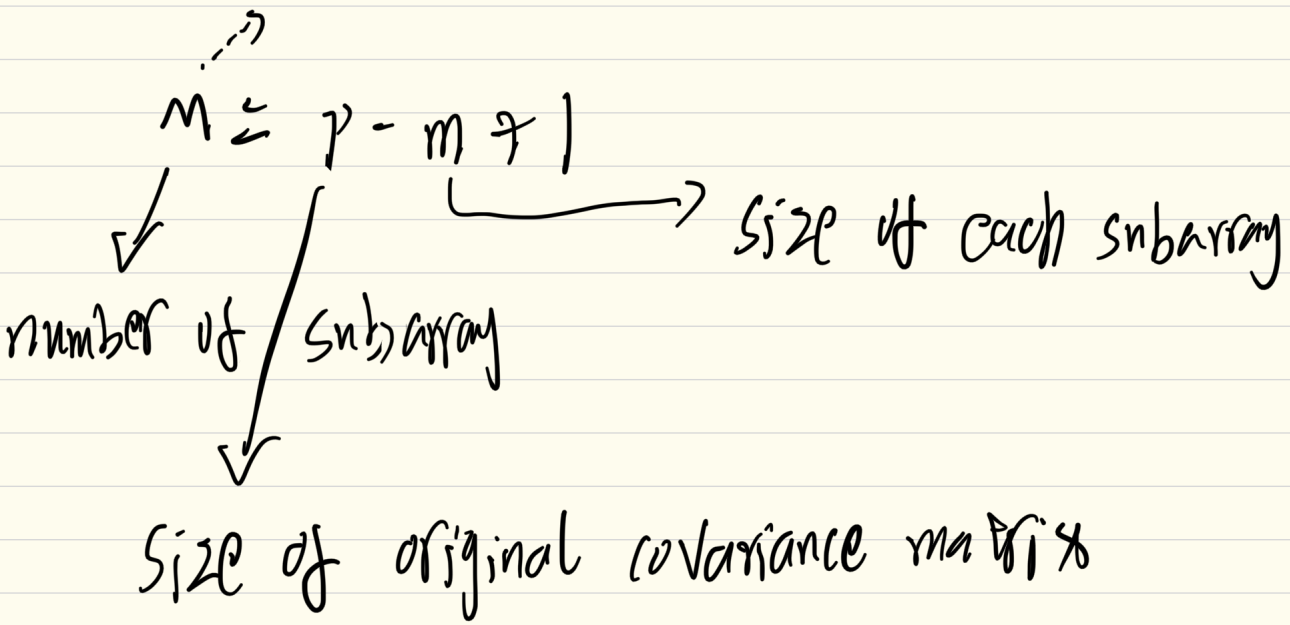

# Q10:



If S is orthogonal to the noise space, the value of $E$ would be small, which means there is a greater probability of a signal at the azimuth.

Q11. Multipaths - Coherents sources.




$m \geq q + 1$

↓

the number of source.

$m \geq 3 + 1 = 4$

$n = p - m + 1$

$= 5 - m + 1 > 0$

$6 > m$

↓

$4 \leq m < 6$

formula:

$$R_k = A D^{k-1} S D^{(k-1)^*} \cdot A^* + \sigma^2 I$$

$$\bar{R} = \frac{1}{n} \cdot \sum_{k=1}^{n} R_k$$

↓

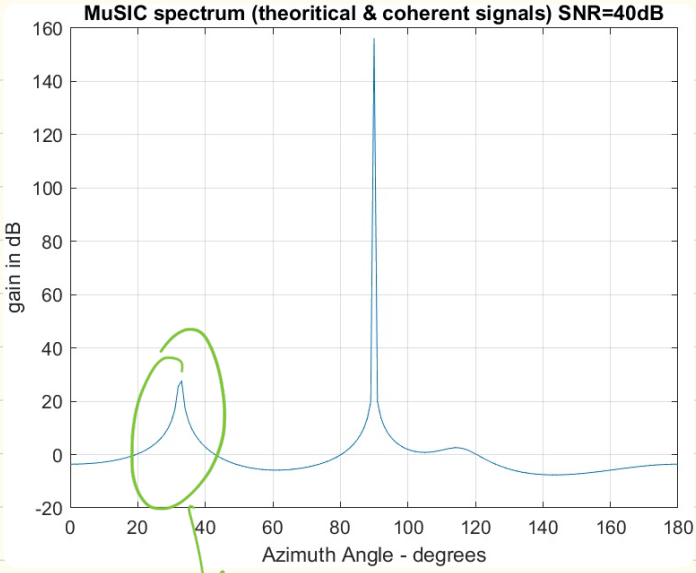
music ( array, $\bar{R}$, $n$, $m$ )

↓ ↓

3 = 4

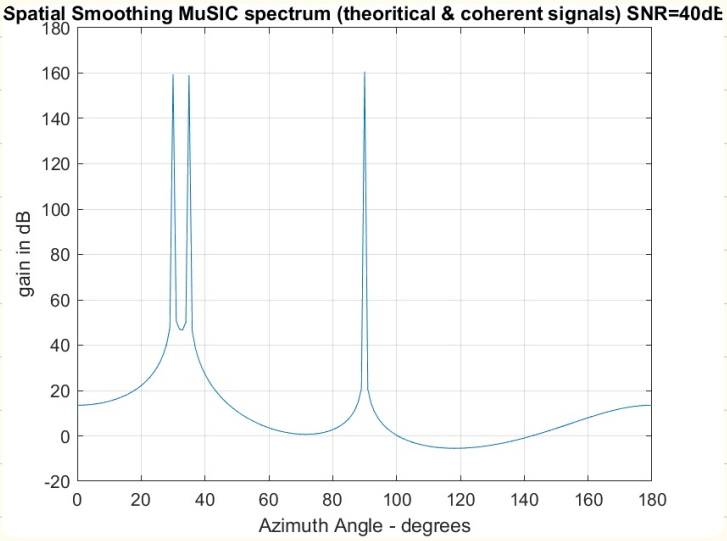
Reference 2.

coherent →




cannot define the interference

2mu th_ music:

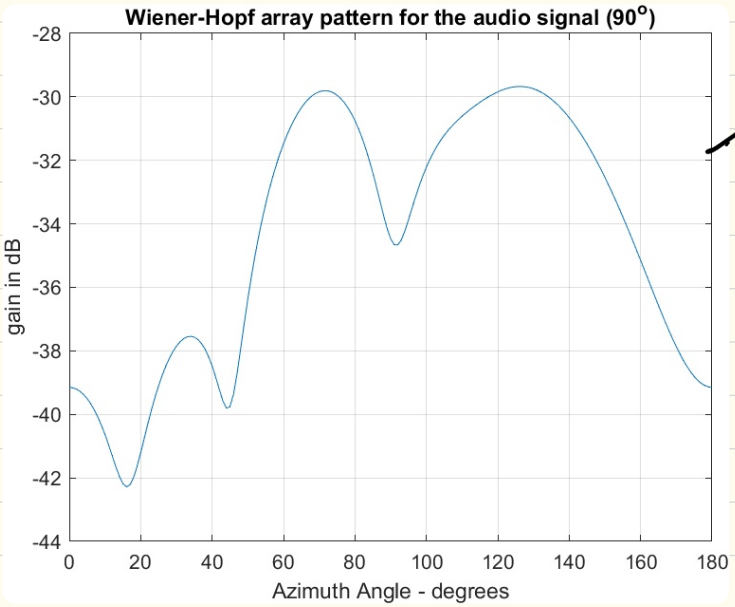



**为什么空间平滑技术可以优化 MUSIC 算法**

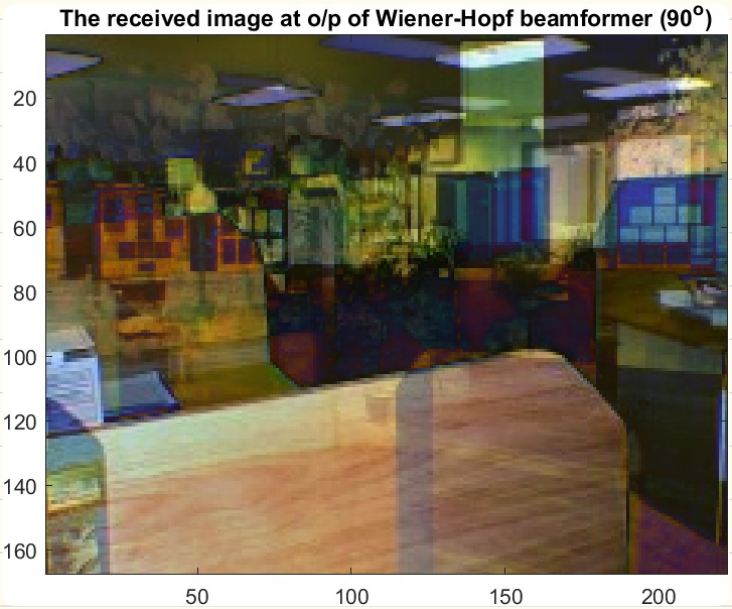
1. **解决信号源相关性问题：**
   - 原始 MUSIC 算法假设信号源之间是非相关的。当存在多个高度相关（或完全相关）的信号源时，原始协方差矩阵的秩会退化，导致算法无法准确分辨多个信号源的位置。
   - 空间平滑技术通过将阵列分割成多个重叠的子阵列，并对每个子阵列的协方差进行平均，有效地减弱了相关信号源之间的耦合，从而恢复协方差矩阵的满秩特性。

# Q12. Reception problem.

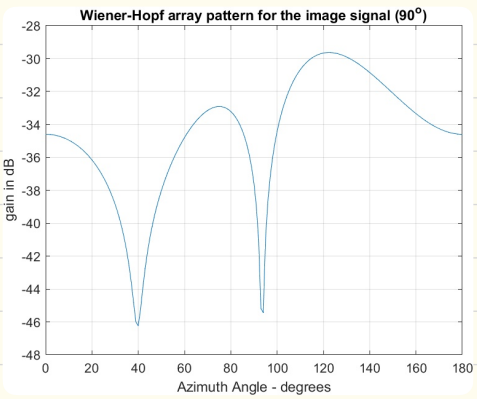
Q12.1, 12.2




→ lots of interference




} Both are not clear




W-H beamformer is not good to receive the 2D signal

Q 12.3 Superresolution beamformer:

- A Superresolution Beamformer based on DOA estimation:

$$\underline{w} = \mathbb{P}^{\perp}_{S_J} \underline{S}_{\substack{\text{desired} \\ \text{signal}}} \quad (94)$$

$$\text{where } \mathbf{S} = [\underline{S}_{\substack{\text{desired} \\ \text{signal}}}, \mathbf{S}_J]$$

  - Provides complete (asymptotically) interference cancellation.
  - Maximizes the SIR at the array output.
  - It is optimum wrt SIR criterion
  - It is a superresolution beamformer (i.e. resolution is not a function of the $\text{SNR}_{\text{in}}$)
  - Needs an estimation algorithm to provide the DOAs of all incident signals.

$$P^{\perp}_{S_J} = \underline{I} - P_{S_J}$$

$$P_{S_J} = S_J \cdot S_J^H \cdot (S_J^H \cdot S_J)^{-1}$$

- A Superresolution Beamformer not based on DOA estimation of interfering sources

$$\underline{w} = \mathbb{P}^{\perp}_{\mathbb{E}_{n_j}} \cdot \underline{S}_{\substack{\text{desired} \\ \text{signal}}} \quad (95)$$

where $\mathbb{E}_{n_j}$ = noise subspace of $\mathbb{R}_{n+J}$
Note: $\mathbb{R}_{n+J}$ = covariance matrix where the effects of the desired signal have been removed

- Maximum Likelihood (ML) Beamformer

$$\underline{w} = \text{col}_{\text{des}}\left(\mathbf{S}^{\#}\right) \quad (96)$$

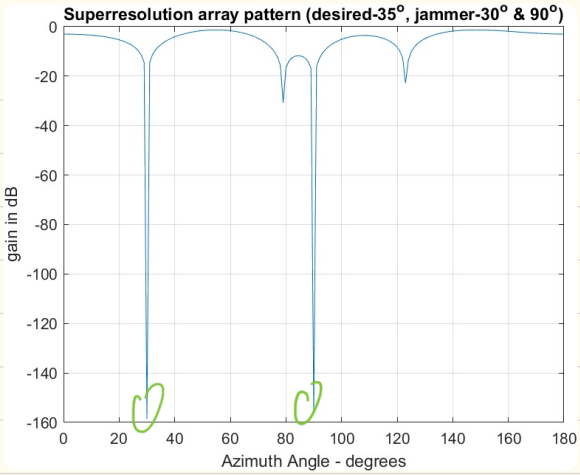
$$\text{where } \mathbf{S}^{\#} = \mathbf{S}.(\mathbf{S}^H\mathbf{S})^{-1} \quad (97)$$



$$P_{E_{nj}} = E_j \cdot E_j^H \cdot (E_j \cdot E_j^H)^{-1}$$

↓

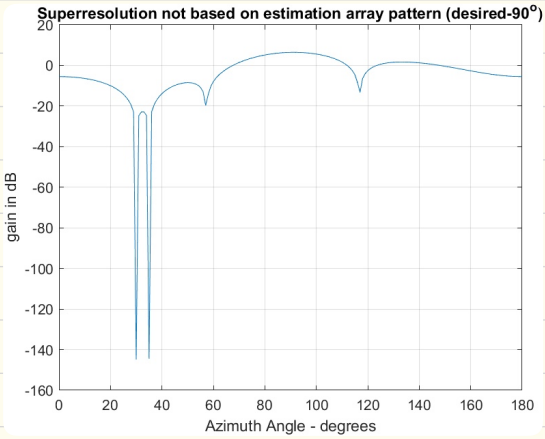
eigenvectors of interference subspace

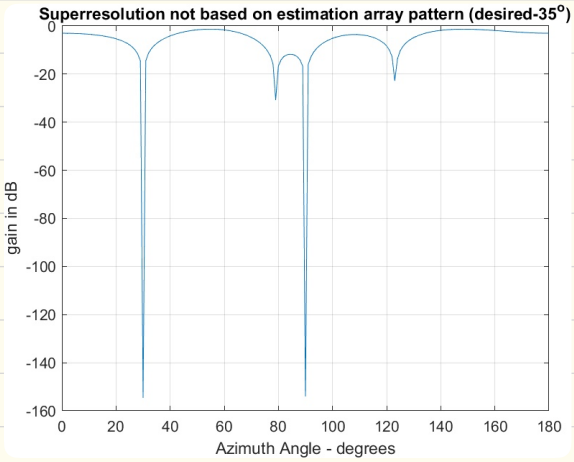



35° is desired
30°, 90° are
suppressed


Based on DoA

not based on DoA.





R 13.

$$x(t) = s(p) \cdot m(t) + n(t)$$

Akaike Information Criterion

The Detection Problem: Basic Detection Theory | Information Theoretic Criteria

**Solution (AIC)**

*Select the model which gives the minimum of AIC[a], i.e.*

$$\min_k \{AIC(k)\} \tag{42}$$

*where*

$$AIC(k) \triangleq -2 \ln \underbrace{\left( \max_k LF^{(k)} \right)}_{ML\ estimator} + 2k \tag{43}$$

*with k denoting the number of free adjusted parameters in the model.*

[a] AIC - Akaike Information Criterion

$$= -2L(N-k)\log\left(\frac{a}{b}\right) + 2k(2N-k)$$

aggregation of eigenvalues

model complexity penalty

MDL:

The Detection Problem: Basic Detection Theory | Information Theoretic Criteria

**Solution (MDL)**

*Select the model which gives the minimum of MDL, i.e.*

$$\min_k \{MDL(k)\} \tag{44}$$

*where*

$$MDL(k) \triangleq -\ln \underbrace{\left( \max_k LF^{(k)} \right)}_{ML\ estimator} + \frac{1}{2} k \ln L \tag{45}$$

*with k denoting the number of free adjusted parameters in the model.*

**N.B.:**

- Apart from a factor of 2, the **first term** in Equation 45 is identical to the corresponding one in the AIC, while the **second term** has an extra factor of $\frac{1}{2} \ln L$.

$$= -2(N-k)\log\left(\frac{a}{b}\right) + 0.5k(2N-k)\log L$$

Result:

$N\_AIC = 3$

$N\_MDL = 3$

Code:

Q1 - Q10

```
%1
array=[-2 0 0; -1 0 0; 0 0 0; 1 0 0; 2 0 0]; % locations of array sensors
directions = [30, 0; 35, 0 ; 90, 0]; % DOAs of sources
Z = my_pattern(array);  % array pattern: Weighted response of the array to incident signals from different directions

plot2d3d(Z,[0:180],0,'gain in dB','initial pattern');

%2
S=spv(array, directions) % Manifold Vector: The phase response received on each element when the signal arrives at the array from different directions

sigma2 = 0.0001; % noise power 40dB
SNR_dB = 10*log10(1 / sigma2);
Rmm = eye(3); %Correlation between individual signal sources
Rxx_theoretical = S*Rmm*S' + sigma2*eye(5,5); % covariance matirx of received signals / The noise is the independent uniformly distributed white noise on each sensor

%3
load Xaudio.mat
load Ximage.mat

soundsc(abs(X_au(2,:)), 11025)
displayimage(X_im(2,:),image_size, 201,'The received signal at the 2nd antenna')

% covariance matrix in practice
Rxx_au = X_au*X_au' / length(X_au(1,:));
Rxx_im = X_im*X_im' / length(X_im(1,:));
%4

%estimated parameters
directions = [];
Rmm = [];
S = [];
sigma2 = [];

%5 Detection problem
eig_theoretical = eig(Rxx_theoretical);
eig_au = eig(Rxx_au);
eig_im = eig(Rxx_im);
M = 3;
%
%6 Estimated problem
%Wiener-Hopf algorithm
Sd = spv(array, [90,0]); % Steering vector: Used to generate array response vectors for specific directions
a=1 % gain factor of WIENER-HOPF Beamformer
wopt = a*inv(Rxx_theoretical) * Sd;
Z = my_pattern(array, wopt);
plot2d3d(Z,[0:180],0,'gain in dB',['Wiener-Hopf theoretical array pattern SNR=',num2str(SNR_dB),'dB']);

%9 Music problem
% Theoritical
Z = music(array, Rxx_theoretical, M);
plot2d3d(Z,[0:180],0,'gain in dB',['MuSIC spectrum (theoritical & coherent signals) SNR=' num2str(SNR_dB) 'dB']);

% audio signal
Z = music(array, Rxx_au, M);
plot2d3d(Z,[0:180],0,'gain in dB',['MuSIC spectrum (audio signal)']);
%
% image signal
Z = music(array, Rxx_im, M);
plot2d3d(Z,[0:180],0,'gain in dB',['MuSIC spectrum (image signal)']);
```

Ry:

```
% Change 40dB to 10dB
%1
array=[-2 0 0; -1 0 0; 0 0 0; 1 0 0; 2 0 0]; % locations of array sensors
directions = [30, 0; 35, 0 ; 90, 0]; % DOAs of sources
Z = my_pattern(array);  % array pattern

plot2d3d(Z,[0:180],0,'gain in dB','initial pattern');

%2
S=spv(array, directions)

sigma2 = 0.1; % noise power 10dB
SNR_dB = 10*log10(1 / sigma2);
Rmm = eye(3);
Rxx_theoretical = S*Rmm*S' + sigma2*eye(5,5); % covariance matirx of received signals

%3
load Xaudio.mat
load Ximage.mat

soundsc(abs(X_au(2,:)), 11025)
displayimage(X_im(2,:),image_size, 201,'The received signal at the 2nd antenna')

% covariance matrix in practice
Rxx_au = X_au*X_au' / length(X_au(1,:));
Rxx_im = X_im*X_im' / length(X_im(1,:));

%4

%estimated parameters
directions = [];
Rmm = [];
S = [];
sigma2 = [];

%5 Detection problem
eig_theoretical = eig(Rxx_theoretical);
eig_au = eig(Rxx_au);
eig_im = eig(Rxx_im);
M = 3;

%6 Estimated problem
Sd = spv(array, [90,0]);
wopt = inv(Rxx_theoretical) * Sd;
Z = my_pattern(array, wopt);
plot2d3d(Z,[0:180],0,'gain in dB',['Wiener-Hopf theoretical array pattern SNR=',num2str(SNR_dB),'dB']);

%9 Music problem
% Theoritical
Z = music(array, Rxx_theoretical, M);
plot2d3d(Z,[0:180],0,'gain in dB',['MuSIC spectrum (theoritical & coherent signals) SNR=' num2str(SNR_dB) 'dB']);
```

```
%Coherent sources
%1
array=[-2 0 0; -1 0 0; 0 0 0; 1 0 0; 2 0 0]; % locations of array sensors
directions = [30, 0; 35, 0 ; 90, 0]; % DOAs of sources
Z = my_pattern(array);  % array pattern: Weighted response of the array to incident signals from different directions

plot2d3d(Z,[0:180],0,'gain in dB','initial pattern');

%2
S=spv(array, directions) % Manifold Vector: The phase response received on each element when the signal arrives at the array from different direc

sigma2 = 0.0001; % noise power 40dB
SNR_dB = 10*log10(1 / sigma2);
Rmm = [1 1 0; 1 1 0; 0 0 1]; %Correlation between individual signal sources
Rxx_theoretical = S*Rmm*S' + sigma2*eye(5,5); % covariance matirx of received signals / The noise is the independent uniformly distributed white
M=3

%9 Music problem
% Theoritical
Z = music(array, Rxx_theoretical, M);
plot2d3d(Z,[0:180],0,'gain in dB',['MuSIC spectrum (theoritical & coherent signals) SNR=' num2str(SNR_dB) 'dB']);

% Smooth MUSIC
Z = smooth_music(array, Rxx_theoretical, M, 4);
plot2d3d(Z,[0:180],0,'gain in dB',['Spatial Smoothing MuSIC spectrum (theoritical & coherent signals) SNR=' num2str(SNR_dB) 'dB']);
```

Q

12.1, 12.2

```
clc
clear all

array=[-2 0 0; -1 0 0; 0 0 0; 1 0 0; 2 0 0];
load Xaudio.mat
load Ximage.mat

%12.1.a WH beamformer
Rxx_au = X_au*X_au' / length(X_au(1,:));
Sd = spv(array, [90,0]);
wopt = inv(Rxx_au) * Sd;
Z = my_pattern(array, wopt);
plot2d3d(Z,[0:180],0,'gain in dB','Wiener-Hopf array pattern for the audio signal (90^o)');
%12.1.b
yt_au=wopt'*X_au;
soundsc(real(yt_au), 11025);

%12.2
Rxx_im = X_im*X_im' / length(X_im(1,:));
Sd = spv(array, [90,0]);
wopt = inv(Rxx_im) * Sd;
yt_im=wopt'*X_im;
yt_im = abs(yt_im);
yt_im = (yt_im-min(yt_im)) * 255/(max(yt_im)-min(yt_im)); % normalize the image to [0,255]
displayimage(yt_im, image_size, 202,'The received image at o/p of Wiener-Hopf beamformer (90^o)');
Z = my_pattern(array, wopt);
plot2d3d(Z,[0:180],0,'gain in dB','Wiener-Hopf array pattern for the image signal (90^o)');


```

12.3

```
load Xaudio.mat
load Ximage.mat
% superresolution Beamformer based on DOA estimation
array=[-2 0 0; -1 0 0; 0 0 0; 1 0 0; 2 0 0];
directions = [30, 0; 35, 0 ; 90, 0];
% DOAs - 30, 35 and 90;
S = spv(array,directions);
Sd = S(:,2); % desired signal is 35

S_if = [S(:,1) S(:,3)]; % DOA of interference signal
%Projection matrix for the interference subspace
P_if = S_if*inv(S_if'*S_if)*S_if';
P_if_orth = eye(size(P_if))-P_if; % Orthogonal projection matrix

wsuper = P_if_orth*Sd;
Z = my_pattern(array, wsuper);
plot2d3d(Z,[0:180],0,'gain in dB','Superresolution array pattern (desired-35^o, interference-30^o & 90^o)');

%Apply the weight on image signal
yt=wsuper'*X_im;
yt = abs(yt);
yt = (yt-min(yt)) * 255/(max(yt)-min(yt)); % normalize the image to [0,255]
displayimage(yt, image_size, 202,'The received signal at o/p of Superresolution Beamformer (35^o)');
```

```
% not based on DOA estimation (Theoretical Signal)
array=[-2 0 0; -1 0 0; 0 0 0; 1 0 0; 2 0 0];
directions = [30, 0; 35, 0 ; 90, 0];
S = spv(array,directions);
sigma2 = 0.1; % noise power SNR=10dB

Rmm = eye(3);
Rxx_theoretical = S*Rmm*S' + sigma2*eye(5,5);
Sd = S(:,2); % desired signale - 35

R_nJ = Rxx_theoretical - Sd*Sd';

[E, D] = eig(R_nJ);
D = diag(D);
[D,I] = sort(D,'descend');
E = E(:,I);
Ej = []; % eigenvectors of interference subspace
Ej = E(:,1:2);

P_nJ = Ej*inv(Ej'*Ej)*Ej';
P_nJ_orth = eye(size(P_nJ))-P_nJ;
wsuper = P_nJ_orth*Sd;

Z = my_pattern(array, wsuper);
plot2d3d(Z,[0:180],0,'gain in dB','Superresolution not based on estimation array pattern (desired-35^o)');
%Apply the weight on image signal
yt=wsuper'*X_im;
yt = abs(yt);
yt = (yt-min(yt)) * 255/(max(yt)-min(yt)); % normalize the image to [0,255]
displayimage(yt, image_size, 202,'The received signal at o/p of Superresolution Beamformer (35^o)');
```

Q 13.

```m
array=[-2 0 0; -1 0 0; 0 0 0; 1 0 0; 2 0 0];
directions = [30, 0; 35, 0 ; 90, 0];

% random source signal from 3 directions
L = 250;
N = size(array, 1);
M = size(directions,1);
m = (randn(M,L) + 1i*randn(M,L)) / sqrt(2); % Create random source

% random noise
sigma2 = 0.1;
noise = sqrt(sigma2) * (randn(N,L) + 1i*randn(N,L)) / sqrt(2);% Create noise
S = spv(array,directions);

% received signal
X = S * m + noise;

% covariance matrix
Rxx = X*X' / length(X(1,:));

eigenv = sort(real(eig(Rxx)), 'descend');

% AIC and MDL criterion
AIC = zeros(1,N);
MDL = zeros(1,N);
for k = 0:N-1
    a = vpa(prod(eigenv(k+1:N) .^(1/(N-k))),6);% The geometric mean of the eigenvalues
    b = (1/(N-k)) * sum(eigenv(k+1:N));% Arithmetic mean of the eigenvalue

    AIC(k + 1) = -2 * L * (N - k) * log(a / b) + 2 * k * (2 * N - k);
    MDL(k + 1) = -L * (N - k) * log(a / b) + 0.5 * k * (2 * N - k) * log(L);
end

[~,M_AIC] = min(AIC);
[~,M_MDL] = min(MDL);

M_AIC = M_AIC - 1;
M_MDL = M_MDL - 1;
```

Music alg.:

```
function gain = music(array, Rxx, M_source, m)
% MUSIC algorithm
p = size(Rxx,1);
if nargin<=3
    m = p; % for spatital smoothing techniques
end

% The eigenvalue decomposition of covariance matrix
[E, D] = eig(Rxx); % eigenvectors and eigenvalues
D = diag(D);
[~,i] = sort(D); % Sort the eigenvalue vector D in ascending order and return the sorted index
E = E(:,i); % The eigenvector matrix E is rearranged by index I

% Construct noise subspace
En = []; % eigenvectors of nosie subspace
for i=1:p-M_source
    En = [En E(:,p-M_source-i+1)];
end

gain = []; % gain according to different directions
for azimuth = 0:180
    S = spv(array,[azimuth,0]);
    S = S(1:m);
    gain = [gain 1/(S'*(En*En')*S)];
end

gain = 10*log10(gain);
end
```

# Smooth music code

```
function gain = smooth_music(array, Rxx, M_source, m)
% MUSCI algprithm with spatitial smoothing technique

p = size(Rxx, 1);
M = p-m+1;
Rxxs = zeros(m,m);
for i=1:M
    Rxxs=Rxxs+Rxx(i:i+m-1,i:i+m-1);
end
Rxxs = Rxxs/M; % smoothed covariance matrix

gain = music(array, Rxxs, M_source, m);
end
```