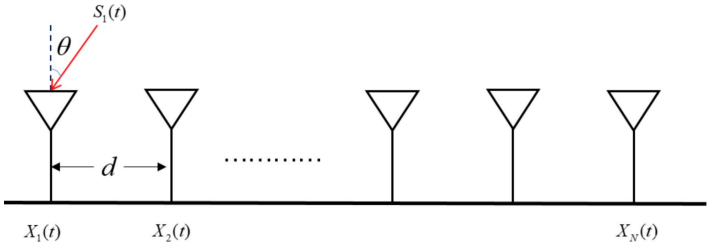
Music (multiple signal classification) algorithm is to find the DOA (direction of Arrival)



MUSIC is a signal-subspace type technique.

MUSIC uses the orthogonality of the signal subspace and the noise subspace to construct a spatical spectrum function. and then estimates the parameters of the signal by searching for the peak.

MuSIC algorithm:

step 0: assumption: M and array geometry are known.

step 1: receive analogue signal vector $\underline{x}(t) \in C^{N\times 1}$ or its discrete version $\underline{x}(t_l)$ for $l = 1, 2, \ldots, L$

step 2: find covariance matrix

$$R_{xx} = \begin{cases} \mathcal{E}\{\underline{x}(t)\cdot\underline{x}(t)^H\} \in C^{N\times N} & \text{in theory} \\ \frac{1}{L}\sum_{l=1}^{L}\underline{x}(t_l)\cdot\underline{x}(t_l)^H \in C^{N\times N} & \text{in practice} \end{cases}$$

step 3. find the Eigenvectors and Eigenvalues of R_{xx}

step 4. form the matrix E_s with columns the eigevectors which correspond to the M largest eigenvalues of R_{xx}

step 5. find the arg of the M minima of the function

$$f_s(p) = \underline{s}(p)^H \cdot P_n \cdot \underline{s}(p) \quad \forall p.$$

where $P_n = I_n - P_{E_s}$ and P_{E_s} is the projection operator onto $\{E_s\}$

* MUSIC estimates the intersection $\{E_s\}$ and the array manifold

for $S(p)$ is the array manifold vector.

project $S(p)$ on to the subspace $\{E_n\}$.

$$Z(p) = P_{E_s} \cdot S(p)$$

$Z(p)$: projection of $S(p)$ onto the subspace $\{E_n\}$.

it should be 0 since they are orthogonal

* The aim of MUSIC is to find the source directions.

Experimental part:

task 1 & 2:

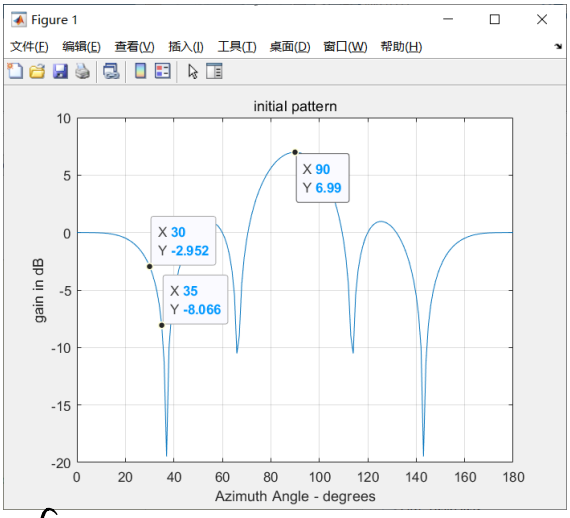
```
array=[-2 0 0;-1 0 0;0 0 0;1 0 0;2 0 0];
direction=[30,0;35,0;90,0];
```

```
%%
% Task 1: Form the  of the above array.
% jay=sqrt(-1);
% theta=direction(:,1);
% r_x=array(:,1)';
% Z_direction=exp(-jay*pi*r_x*cos(30*pi/180))';
% array(:,1)=Z_direction;                          SPV in x direction
Z=pattern(array);
figure(1)
plot2d3d(Z,[0:180],0,'gain in dB','initial pattern');
% for 30, gain is -2.95dB; for 35, gain is -8.066dB; for 90. gain is 6.99dB

% Task 2: Theoretical Covariance Matrix Formation:
S=spv(array, direction);    % we can get a 5*3 array and it is just the Source Position Vectors:
Rmm=eye(length(direction));
sigma2=0.0001;
Rxx_theoretical=S*Rmm*S'+sigma2*eye(5,5);
```

for the three sources: gain in 30° is −2.952 dB

35° is −8.066 dB. 90° is 6.99 dB.



and above are all azimuth.

and we can get manifold S:

5x3 complex double

	1	2	3
1	0.6661 - 0.7458i	0.4210 - 0.9071i	1.0000 + 0.0000i
2	-0.9127 + 0.4086i	-0.8429 + 0.5381i	1.0000 + 0.0000i
3	1.0000 + 0.0000i	1.0000 + 0.0000i	1.0000 + 0.0000i
4	-0.9127 - 0.4086i	-0.8429 - 0.5381i	1.0000 - 0.0000i
5	0.6661 + 0.7458i	0.4210 + 0.9071i	1.0000 - 0.0000i

for Rmm. we can get a 3x3 matrix, and the diagonals of the matrix are all 1s, and the rest are all zeros.

for Rxx_theoretical:

	1	2	3	4	5
1	3.0001 + 0.0000i	-0.7556 + 0.9467i	2.0871 - 1.6529i	0.8300 + 1.9440i	0.2419 - 1.7573i
2	-0.7556 - 0.9467i	3.0001 + 0.0000i	-0.7556 + 0.9467i	2.0871 - 1.6529i	0.8300 + 1.9440i
3	2.0871 + 1.6529i	-0.7556 - 0.9467i	3.0001 + 0.0000i	-0.7556 + 0.9467i	2.0871 - 1.6529i
4	0.8300 - 1.9440i	2.0871 + 1.6529i	-0.7556 - 0.9467i	3.0001 + 0.0000i	-0.7556 + 0.9467i
5	0.2419 + 1.7573i	0.8300 - 1.9440i	2.0871 + 1.6529i	-0.7556 - 0.9467i	3.0001 + 0.0000i
6					

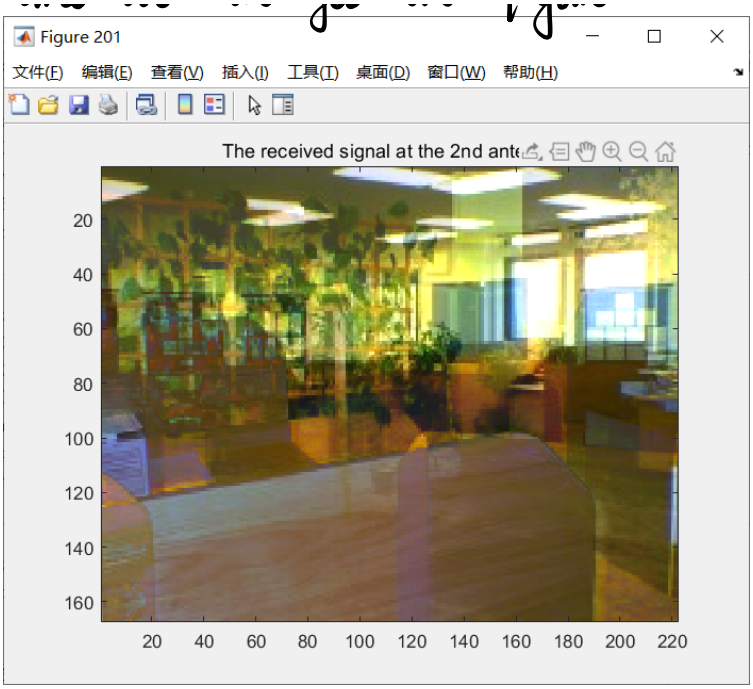
Task 3:

```
%%
% Task 3: Practical Covariance Matrix:
load Xaudio.mat;
load Ximage.mat;
sound(abs(X_au(2,:)), 11025);
displayimage(X_im(2,:),image_size, 201,'The received signal at the 2nd antenna');
Rxx_au= X_au* X_au'/length(X_au(1,:));
Rxx_im= X_im* X_im'/length(X_im(1,:));
```

when we run this part, we can't hear the music.

we need to use "soundsc" rather than sound to play the music in this task.

And we can get the figure:



in fact, this picture is formed by superimposing three different picture.

For Rxx_au:

5x5 complex double

	1	2	3	4	5	6
	1.1997e+05 + 0.0000e+00i	-5.1814e+04 - 1.0120e+04i	1.0504e+05 - 8.2967e+04i	1.4680e+03 + 5.5453e+04i	2.6946e+04 - 1.1453e+05i	
	-5.1814e+04 + 1.0120e+04i	2.9137e+04 + 0.0000e+00i	-3.7674e+04 + 4.7515e+04i	-1.3849e+03 - 2.5954e+04i	1.4680e+03 + 5.5453e+04i	
	1.0504e+05 + 8.2967e+04i	-3.7674e+04 - 4.7515e+04i	1.5086e+05 + 0.0000e+00i	-3.7674e+04 + 4.7515e+04i	1.0504e+05 - 8.2967e+04i	
	1.4680e+03 - 5.5453e+04i	-1.3849e+03 + 2.5954e+04i	-3.7674e+04 - 4.7515e+04i	2.9137e+04 + 0.0000e+00i	-5.1814e+04 - 1.0120e+04i	
	2.6946e+04 + 1.1453e+05i	1.4680e+03 - 5.5453e+04i	1.0504e+05 + 8.2967e+04i	-5.1814e+04 + 1.0120e+04i	1.1997e+05 + 0.0000e+00i	

For Rxx_im:

5x5 complex double

	1	2	3	4	5
1	6.4023e+04 + 0.0000e+00i	-1.4333e+04 - 9.0323e+03i	5.9094e+04 - 3.8583e+04i	8.7597e+03 + 2.2303e+04i	2.3832e+04 - 5.4652e+04i
2	-1.4333e+04 + 9.0323e+03i	1.4890e+04 + 0.0000e+00i	-6.5491e+03 + 2.2141e+04i	1.6189e+03 - 7.6468e+03i	8.7597e+03 + 2.2303e+04i
3	5.9094e+04 + 3.8583e+04i	-6.5491e+03 - 2.2141e+04i	8.0591e+04 + 0.0000e+00i	-6.5491e+03 + 2.2141e+04i	5.9094e+04 - 3.8583e+04i
4	8.7597e+03 - 2.2303e+04i	1.6189e+03 + 7.6468e+03i	-6.5491e+03 - 2.2141e+04i	1.4890e+04 + 0.0000e+00i	-1.4333e+04 - 9.0323e+03i
5	2.3832e+04 + 5.4652e+04i	8.7597e+03 - 2.2303e+04i	5.9094e+04 + 3.8583e+04i	-1.4333e+04 + 9.0323e+03i	6.4023e+04 + 0.0000e+00i
6					

Task 4 & 5:

```
%%
% Task 4
directions=[];
Rmm=[];
S=[];
sigma2=[];

% Task 5: Detection Problem
Eig=eig(Rxx_theoretical);
Eig_au=eig(Rxx_au);
Eig_im=eig(Rxx_im);
```

we can get eigenvalue for Rxx_theoretical:

5x1 double

	1
1	1.0000e-04
2	1.0000e-04
3	0.0953
4	4.9681
5	9.9369

for Eig_au:

	1
1	0.0100
2	0.0101
3	173.3260
4	1.0720e+04
5	4.3818e+05
6	

for Eig_im:

5x1 double

	1
1	-9.4922e-11
2	7.0614e-11
3	190.8705
4	2.0391e+04
5	2.1784e+05

Question. 5.1.

we can get the noise power and the number of sources by observing the eigenvalues of Rxx_theoretical.

Question 5.2.

① for the number of source:

- Therefore, theoretically, the number of emitting sources M can be determined by the eigenvalues of the covariance matrix $\mathbb{R}_{xx}$ of the Rx signal-vector $\underline{x}(t)$, and more specifically by the following expression

$$M = N - (\text{multiplicity of } \textbf{minimum} \text{ eigenvalue of } \mathbb{R}_{xx}) \quad (35)$$

Hence $M = N - 2 = 5 - 2 = 3$.

there are 3 source.

② for the noise power:

- if M is **known** then .

$$\begin{aligned} \widehat{\sigma}_n^2 &= \text{the average of the } N - M \text{ smallest eigenvalues} \\ &= \frac{1}{N-M}\left(\sigma_{M+1}^2 + \sigma_{M+2}^2 + ... + \sigma_N^2\right) \end{aligned} \quad (41)$$

- If M is **unknown** then its estimation is not an easy task and a naive approach is likely to fail. Solution: see next section (Decision Theory).

we have known. M is 3. N is 5.

Hence. the noise power is $\frac{1}{2} \times (0.0001 + 0.0001) = 0.0001$

What's more, in fact. $\sigma_n^2 \Rightarrow$ the minimum eigenvalues of Rxx is just the noise power.

Question 5.3.

repeat the instruction for Rxx_cm and Rxx_im.

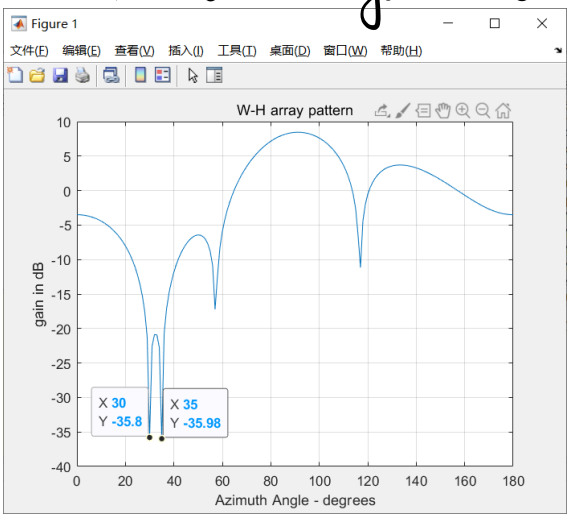
we can get their eigenvalues as above

and we can see that the conclusions in 5.2 is not valid. in the eigenvalue matrix, the 5 values are totally different, which means in the practical situation, the conclusion in 5.2 is not correct. to find the number of source in reality. we need to use MDL and AIC

Task 6.

```
%%
% Task 6: Estimation Problem
Sd=spv(array,[90,0]);
a=6.99;
wopt= a*inv(Rxx_theoretical)*Sd;                    % optimum Wiener-Hopf solution
Z=pattern(array, wopt);
plot2d3d(Z,[0:180], 0, 'gain in dB','W-H array pattern');
```

then we can get the figure:



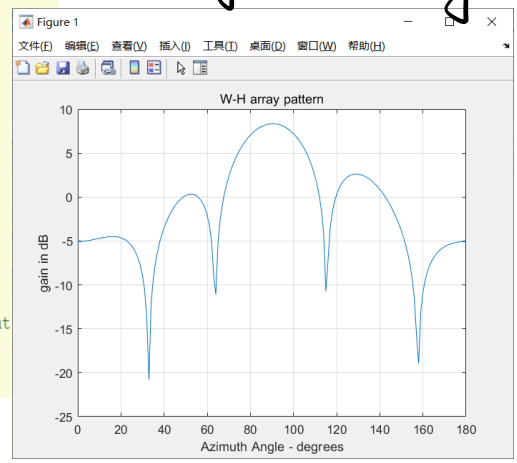
and we can find the lowest point. 30° and 35° in azimuth.
Hence. we can distinguish the directions of the two interferences is 30° and 35°.

Task 7.

Repeat 2. 4. 5.1 and 6.
with noise power $\sigma^2 = 10dB$,
Hence $\sigma^2 = 0.1$

```
%%
% Task 7:repeat 2 4 5.1 6
S=spv(array, direction);
Rmm=eye(length(direction));
sigma2=0.1;      % here the noise is 10dB
Rxx_theoretical=S*Rmm*S'+sigma2*eye(5,5);
directions=[];
Rmm=[];
S=[];
sigma2=[];
Eig=eig(Rxx_theoretical);
Sd=spv(array,[90,0]);
a=6.99;
wopt= a*inv(Rxx_theoretical)*Sd;                    % optimum Wiener-Hopf solut
Z=pattern(array, wopt);
plot2d3d(Z,[0:180], 0, 'gain in dB','W-H array pattern');
```

we can get the figure:



Task 8.

compare the figures between 6 and 7.
we can see that when the noise power is -40 dB, we can easily distinguish the direction, but when -10 dB, we can't distinguish them.
which means when SNR is not huge, we can't distinguish the directions of interference successfully.

Task 9.

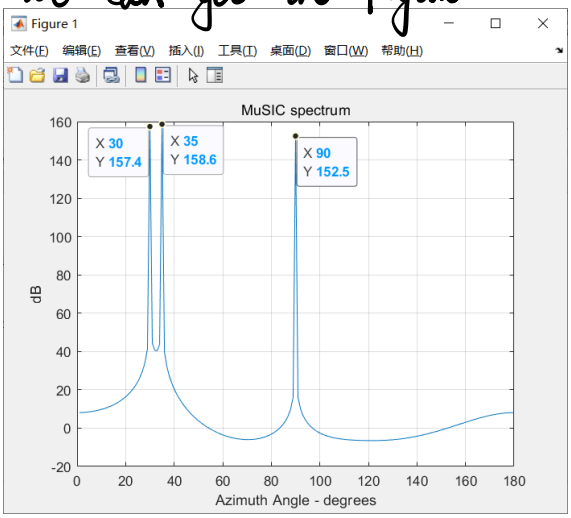
study and implement the Music algorithm:

```
function Z=music(array,Rxx_theoretical,sources_num)
[eigvector,eigvalue]=eig(Rxx_theoretical);     %find the eigvector and eigvalue
[eigvalue_s,I]=sort(sum(eigvalue),'descend');  %sort the eigvalue
eigvector_s=eigvector(:,I);
source=eigvector_s(:,1:sources_num);
for theta=1:180
    Sp=spv(array,[theta,0]);
    Z(theta)=Sp'*(eye(5)-fpo(source))*Sp;
end
Z=-10*log10(Z);
end
```

then use this function:

```
Z = music(array, Rxx_theoretical)
plot2d3d(Z,[0:180],0,'dB', 'MuSIC spectrum');
```

we can get the figure:

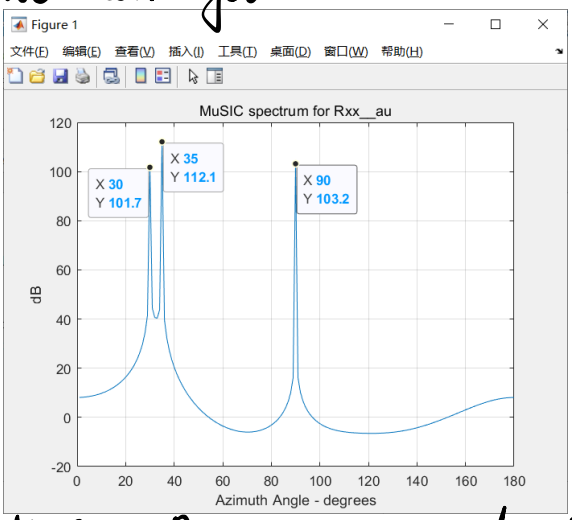


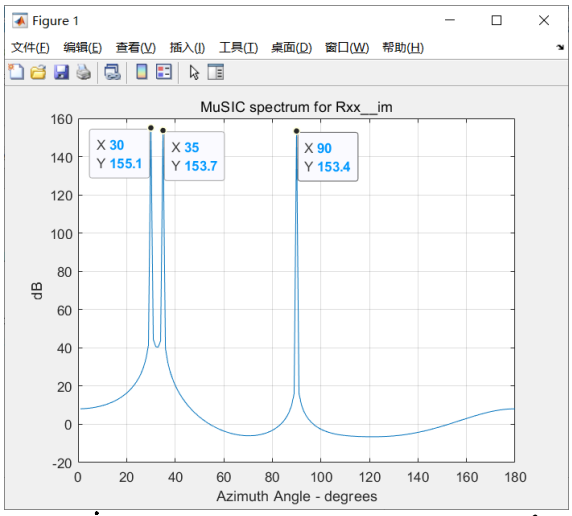
we can get the conclusion from the figure. that the direction of the source is in the 30°, 35° and 90° in azimuth. because in these direction. the gain is very high.

Task 10:

Repeat instruction 9 by using Rxx_au and Rxx_im:

we can get:





use Rxx_au and Rxx_im. we can get the same conclusion. the direction is the same as we use Rxx_theoretical.

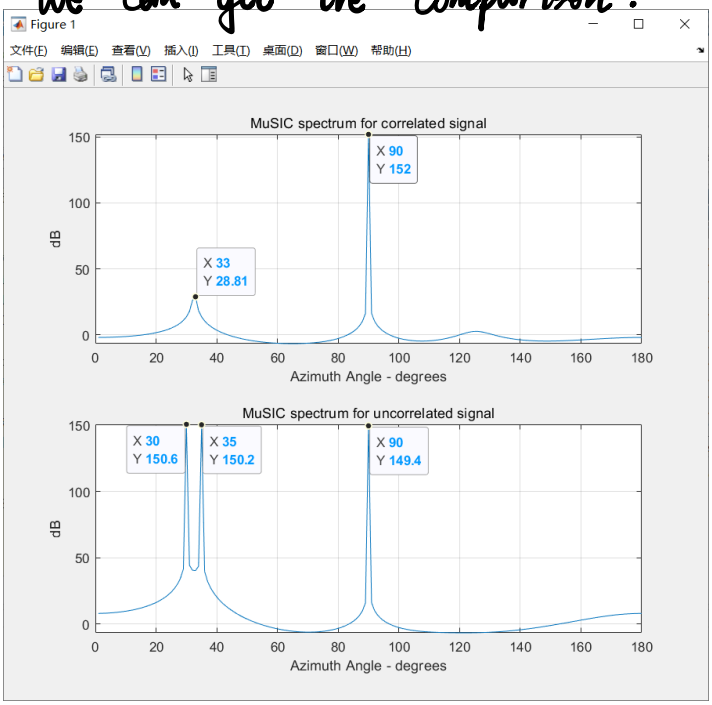
Task 11.

we mainly learn the usage of spatial smoothing technique in this task.

for correlated signals and uncorrelated signals, we can adjust Rmm to change the results.

```
%%
% Task 11
array=[-2 0 0; -1 0 0; 0 0 0; 1 0 0; 2 0 0];
directions = [30, 0; 35, 0 ; 90, 0];
Num_receiver = 5;
Num_source = 3;
SS = spv(array, directions);                    %get 5 * 3 Source Position Vector
Rmm_uncorr = [1 0 0;0 1 0;0 0 1];               %uncorrelated signal
Rmm_corr = [1 1 0;1 1 0;0 0 1];                 %correlated signal
sigma2=0.0001;
Rxx_uncorr = SS*Rmm_uncorr*SS'+sigma2*eye(Num_receiver);
Rxx_corr = SS*Rmm_corr*SS'+sigma2*eye(Num_receiver);
subplot(2,1,1)
Z_corr = music(array, Rxx_corr,3);
plot2d3d(Z_corr,[1:180],0,'dB', 'MuSIC spectrum for correlated signal');
subplot(2,1,2)
Z_uncorr = music(array, Rxx_uncorr,3);
plot2d3d(Z_uncorr,[1:180],0,'dB', 'MuSIC spectrum for uncorrelated signal');
```

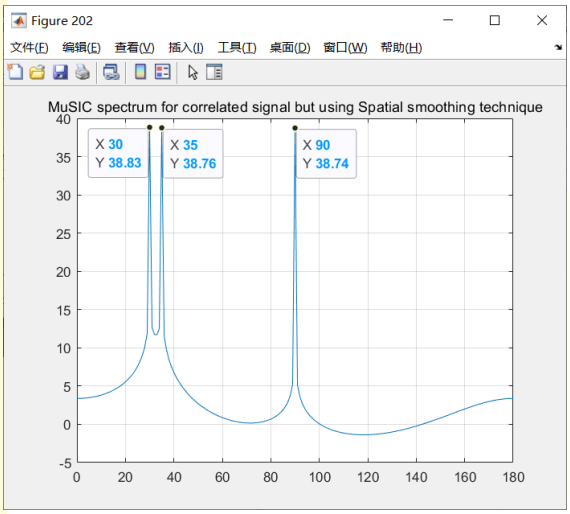
we can get the comparison:



when the signals are correlated, we can't distinguish the signal from 30° and 35°.

to overcome the problem, we can use spatial smoothing technique

```
%spatial smoothing technique
[M,MM] = size(Rxx_corr);
Num_element = 4;                          %The number of element in the subarrray
Num_subarray = M - Num_element + 1;                          %The number of subarray
Rxx_window=zeros(Num_element,Num_element);
for i=1:Num_subarray%第1个子阵列到第N个子阵列
Rxx_window=Rxx_window+Rxx_corr(i:i+Num_element-1,i:i+Num_element-1);
end
Rxx_spa=Rxx_window/Num_subarray;
[EV,D]=eig(Rxx_spa);
EVA=diag(D)';
[EVA,I]=sort(EVA);
EVA=fliplr(EVA);
EV=fliplr(EV(:,I));
Es = EV(:,[1 2 3]);
PEs = fpo(Es);
Pn = eye(Num_element) - PEs;
for i = 0:180
    SS = spv([-2 0 0;-1 0 0;0 0 0;1,0,0],[1,0]);
    kesai_spa(i+1) = 1/(SS'*Pn*SS);
    kesai_spa_dB(i+1) = 10*log10(kesai_spa(i+1))/Num_element;
end
plot([0:180],kesai_spa_dB);
grid on;
title('MuSIC spectrum for correlated signal but using Spatial smoothing technique')
```



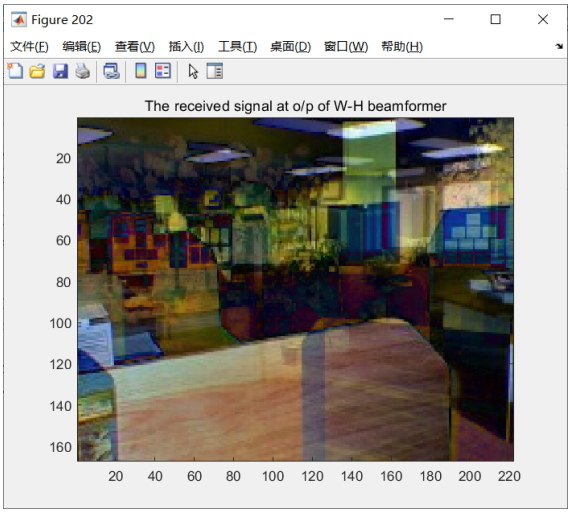
we can see that after using spatial smoothing, the correlated signals can be distinguished again.

Task 12

```
% Task 12
Sd_au=spv(array,[90,0]);
wopt_au=pinv(Rxx_au)*Sd_au;
yt_au=wopt'* X_au;
sound(real(yt_au), 11025);

Sd_im=spv(array,[90,0]);
wopt=pinv(Rxx_im)*Sd_im;
yt_im=wopt'* X_im;
yt_im_nor = mapminmax(yt_im, 0, 255);
displayimage(yt_im_nor, image_size, 202,'The received signal at o/p of W-H beamformer');
```

we can hear a boy sings a song, but it is clearer than played in task 3.

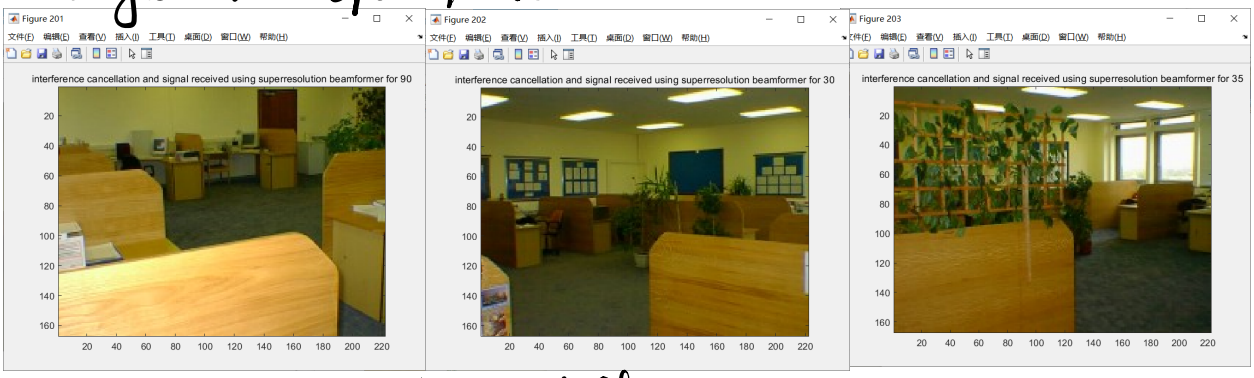


and we can get a picture 202. as left.

12.3: we can see that in 12.2, the picture is formed by 3 different picture, now we

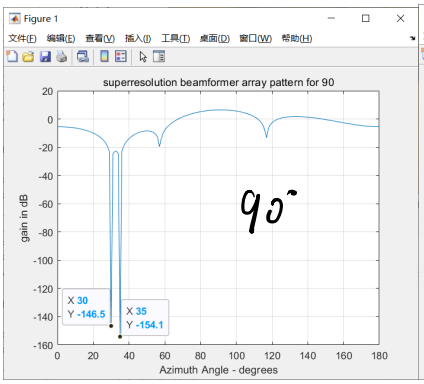
```
% Task 12_3
S_d = spv(array,[90,0]);
S_j = spv(array,[30,0;35,0]);
P = fpoc(S_j);
w = P * S_d;
yt = w'*X_im;
yt_im_nor = mapminmax(yt, 0, 255);
displayimage(yt_im_nor, image_size, 201,'interference cancellation and signal received using superresolution beamformer');
Z=pattern(array, w);
plot2d3d(Z, [0:180], 0, 'gain in dB','superresolution beamformer array pattern');
%%
S_d_30 = spv(array,[30,0]);
S_j_30 = spv(array,[35,0;90,0]);
P_30 = fpoc(S_j_30);
w_30 = P_30 * S_d_30;
yt_30 = w_30'*X_im;
yt_im_nor_30 = mapminmax(yt_30, 0, 255);
displayimage(yt_im_nor_30, image_size, 202,'interference cancellation and signal received using superresolution beamformer');
Z_30=pattern(array, w_30);
plot2d3d(Z_30, [0:180], 0, 'gain in dB','superresolution beamformer array pattern');
%%
S_d_35 = spv(array,[35,0]);
S_j_35 = spv(array,[30,0;90,0]);
P_35 = fpoc(S_j_35);
w_35 = P_35 * S_d_35;
yt_35 = w_35'*X_im;
yt_im_nor_35 = mapminmax(yt_35, 0, 255);
```

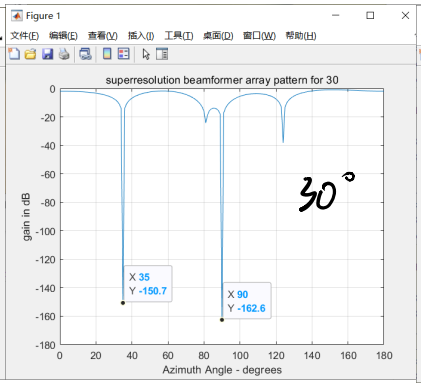
to get the three picture, we can just change the angle in spv function.

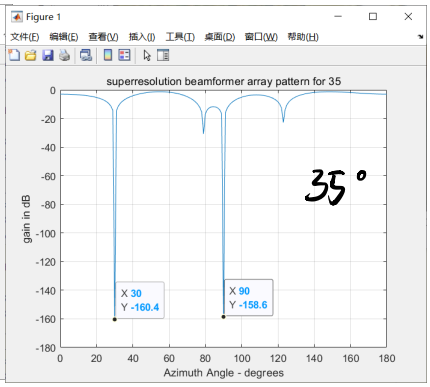


we can get three different picture.

and for superresolution beamformer array pattern:





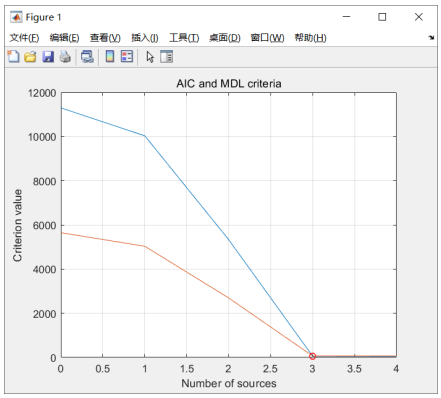


Task 13.

in this task. we use AIC and MDL to detect the three source:

```
% Task 13
array=[-2 0 0; -1 0 0; 0 0 0; 1 0 0; 2 0 0];
directions = [30, 0; 35, 0 ; 90, 0];
Num_receiver = 5;
Num_source = 3;
SS = spv(array, directions);                    %Known signal source and manifold vector
sigma2=0.0001;
Rmm = eye(Num_source);
L = 250;                                        %Generate L snapshots of x(t)
Rxx_theoretical = SS*Rmm*SS'+sigma2*eye(Num_receiver);
[EV,D]=eig(Rxx_theoretical);
z_t1 = zeros(Num_receiver,L);
x_t1 = zeros(Num_receiver,L);
for i = 1 : L
    z_t1(:,i) = (randn(Num_receiver,1)+1i*randn(Num_receiver,1))/sqrt(2);
    x_t1(:,i) = EV * sqrt(D) * z_t1(:,i);
end

Rxx_practical = x_t1 * x_t1'/L;
[EV_p,D_p]=eig(Rxx_practical);
eigenvalues=sort(diag(D_p),'descend');
 % criterion
for k = 0 : Num_receiver-1
    index = 1/(Num_receiver-k);
    %get geometric mean and arithmetic mean
    geometric = prod(eigenvalues(k+1:Num_receiver)).^(index);
    arithmetic = index * sum(eigenvalues(k+1:Num_receiver));
    %using AIC equation and MDL equation from paper
    AIC(k+1) = -2*L*log(((geometric/arithmetic)^((Num_receiver-k))))+2*k*(2*Num_receiver-k);
    MDL(k+1) = -L*log(((geometric/arithmetic)^((Num_receiver-k))))+0.5*k*(2*Num_receiver-k)*log(L);
end
[AIC_value_min,num_sources]=min(AIC);
[MDL_value_min,num_sources]=min(MDL);
num_sources = num_sources-1;       %retrieve 1 because the value k=0 is stocked in the index 1.
```



we can get the picture.
so. the both 'min is 3.
so we can detect that.
the number of sources is 3.