1, radiation pattern of linear array and n element uniform array

Program 1

```
clear all;
lambda=0.03;
An = 1;
d=0.5*lambda;
k=(2*pi)/lambda;
N=30;
j = sqrt(-1);
A=zeros(1,360);
for theta=1:360
    deg2rad(theta) = (theta*pi)/(180);
    for n=0:N-1
        AF(theta) = AF(theta)+An*exp(j*k*n*d*cos(theta)+beta);
    end
end
AF(theta)=abs(AF(theta));
y=10*log(AF(theta));
plot(beta,A)
```

Program 2

```
clear all;
lambda = 0.03;
An = 1;
d = 0.5 * lambda;
k = (2 * pi) / lambda;
N = 30;
j = sqrt(-1);
AF = zeros(1, 360);
theta_deg = 1:360;
for theta = theta_deg
    theta_rad = (theta * pi) / 180;
    for n = 0:N-1
        AF(theta) = AF(theta) + An * exp(j * k * n * d * cos(theta_rad));
    end
    AF(theta) = abs(AF(theta));
end
AF_dB = 10 * log10(AF);
figure;
plot(theta_deg, AF_dB);
```

```
xlabel('Angle (degrees)');
ylabel('Array Factor (dB)');
title('Radiation Pattern of Uniform Linear Array');
grid on;
```

2.Aoa estimation using music and espritt algorithm

Program 1

```
%% Parameters
N = 8;
M = 2;
SNR = 20;
Fs = 1000;
T = 1;
f = [0.1, 0.2];
angles = [30, 60];

%% Generate signals
t = 0:1/Fs:T-1/Fs;
S = zeros(M, length(t));
for k = 1:M
   S(k,:) = cos(2*pi*f(k)*t);
end

%% Simulate sensor array
d = 0.5;
theta = angles;
steeringVectors = zeros(N, M);

for m = 1:M
    for n = 1:N
        steeringVectors(n, m) = exp(1j*2*pi*(n-1)*d*sind(theta(m)));
    end
end

A = steeringVectors;
X = A*S + (randn(N, length(t)) + 1j*randn(N, length(t)))/sqrt(2);
```

```
%% MUSIC Algorithm
Rxx = X*X'/length(t);
[eigenVecs, eigenVals] = eig(Rxx);
[~, ind] = sort(diag(eigenVals), 'descend');
eigenVecs = eigenVecs(:, ind);

Us = eigenVecs(:, 1:M);
Un = eigenVecs(:, M+1:end);

angles_range = -90:0.1:90;
P_music = zeros(size(angles_range));

for i = 1:length(angles_range)
   a = exp(1j*2*pi*(0:N-1)'*d*sind(angles_range(i)));
   P_music(i) = 1 / (a' * (Un * Un') * a);
end

figure;
plot(angles_range, 10*log10(abs(P_music)/max(abs(P_music))));
xlabel('Angle (Degrees)');
ylabel('MUSIC Spectrum (dB)');
title('MUSIC AOA Estimation');
grid on;

%% ESPRIT Algorithm
[eigenVecs, eigenVals] = eig(Rxx);
[~, ind] = sort(diag(eigenVals), 'descend');
eigenVecs = eigenVecs(:, ind);

Us = eigenVecs(:, 1:M);
Phi = Us(2:N, :) \ Us(1:N-1, :);

[eigenVecs, eigenVals] = eig(Phi);
angles_esprit = asin(angle(diag(eigenVals)) / (2*pi*d)) * (180/pi);

figure;
stem(angles_esprit, 'filled');
xlabel('Source Index');
ylabel('Estimated AOA (Degrees)');
title('ESPRIT AOA Estimation');
grid on;
```

Program 2

```
clc;
clear;
close all;

% Parameters
N = 8;
M = 2;
SNR = 10;
theta = [-30, 30];
lambda = 1;
d = lambda/2;


theta_rad = theta * pi / 180;
A = exp(1j * (0:N-1)' * 2 * pi * d / lambda * sin(theta_rad));


S = randn(M, 1000) + 1j * randn(M, 1000);
X = A * S;


noise = (randn(N, 1000) + 1j * randn(N, 1000)) * 10^(-SNR/20);
X_noisy = X + noise;


R = (X_noisy * X_noisy') / size(X_noisy, 2);


[EV, D] = eig(R);
[~, idx] = sort(diag(D), 'descend');
En = EV(:, idx(M+1:end));
theta_scan = -90:0.1:90;
P_music = zeros(size(theta_scan));
for i = 1:length(theta_scan)
   a = exp(1j * (0:N-1)' * 2 * pi * d / lambda * sin(theta_scan(i) * pi / 180));
   P_music(i) = 1 / (a' * (En * En') * a);
end


[EV, D] = eig(R);
[~, idx] = sort(diag(D), 'descend');
E = EV(:, idx(1:M));
E1 = E(1:end-1,:);
```

```
E2 = E(2:end,:);
phi = pinv(E1) * E2;
[~, D_esp] = eig(phi);
angles_esp = angle(diag(D_esp));
theta_esp = asin(angles_esp) * (180/pi) * (lambda / (2 * pi * d));

figure;
subplot(2,1,1);
plot(theta_scan, 10*log10(P_music/max(P_music)), 'LineWidth', 2);
title('MUSIC Spectrum');
xlabel('Angle (degrees)');
ylabel('Normalized Power (dB)');
grid on;

subplot(2,1,2);
stem(theta_esp, ones(size(theta_esp)), 'r', 'filled');
title('ESPRIT Estimated Angles');
xlabel('Angle (degrees)');
ylabel('Magnitude');
grid on;
```

3.estimates weight of an array using final weight estimate array factor and means square error

Program 1

```
N = 8;
M = 3;
theta = [30, 60, 90];
SNR = 20;
f = [0.1, 0.2, 0.3];
T = 1;
Fs = 1000;
d = 0.5;

t = 0:1/Fs:T-1/Fs;
S = zeros(M, length(t));
for k = 1:M
   S(k, :) = cos(2*pi*f(k)*t);
end

A = zeros(N, M);
```

```
for m = 1:M
    for n = 1:N
        A(n, m) = exp(1j*2*pi*(n-1)*d*sind(theta(m)));
    end
end

X = A*S + (randn(N, length(t)) + 1j*randn(N, length(t)))/sqrt(2);

Rxx = (X * X') / length(t);

desired_signal = ones(N, 1);
weights = (A' * A) \ A' * desired_signal;

angles_range = -90:0.1:90;
AF = zeros(size(angles_range));
for i = 1:length(angles_range)
    a = exp(1j*2*pi*(0:N-1)'*d*sind(angles_range(i)));
    AF(i) = abs(weights' * a);
end

AF = AF / max(AF);

figure;
plot(angles_range, 20*log10(AF));
xlabel('Angle (Degrees)');
ylabel('Array Factor (dB)');
title('Array Factor (AF)');
grid on;

error = (A * weights - desired_signal);
MSE = mean(abs(error).^2);

disp(['Mean Square Error (MSE): ', num2str(MSE)]);
```

Program 2

```
clc
clear

N = 10
d = 0.5
theta = linspace(0, pi, 180)
desired_pattern = sin(theta)
```

```
weights = ones(N, 1)

A = zeros(length(theta), N)
for i = 1:N
    A(:, i) = exp(1j * 2 * pi * (i-1) * d * sin(theta))
end

weights = (A' * A) \ (A' * desired_pattern')

array_factor = abs(A * weights)

mse = mean((desired_pattern' - array_factor).^2)

disp('Estimated Weights:')
disp(weights)
disp('Mean Square Error:')
disp(mse)
```

4.dynamixally alter the main lobe direction based on the information of aoa

Program

```
N = 8
M = 1
SNR = 20
f = 0.1
T = 1
Fs = 1000
d = 0.5
theta_actual = 45

t = 0:1/Fs:T-1/Fs
S = cos(2*pi*f*t)

theta = theta_actual
A = zeros(N, 1)
for n = 1:N
A(n) = exp(1j*2*pi*(n-1)*d*sind(theta))
end
```

```
X = A*S + (randn(N, length(t)) + 1j*randn(N, length(t)))/sqrt(2)

Rxx = (X * X') / length(t)
[eigenVecs, eigenVals] = eig(Rxx)
[~, ind] = sort(diag(eigenVals), 'descend')
eigenVecs = eigenVecs(:, ind)

Us = eigenVecs(:, 1:M)
Un = eigenVecs(:, M+1:end)

angles_range = -90:0.1:90
P_music = zeros(size(angles_range))
for i = 1:length(angles_range)
a = exp(1j*2*pi*(0:N-1)'*d*sind(angles_range(i)))
P_music(i) = 1 / (a' * (Un * Un') * a)
end

[~, peak_idx] = max(abs(P_music))
estimated_theta = angles_range(peak_idx)

disp(['Estimated AOA:', num2str(estimated_theta), 'degrees'])

A_dynamic = zeros(N, 1)
for n = 1:N
A_dynamic(n) = exp(1j*2*pi*(n-1)*d*sind(estimated_theta))
end

angles_range_AF = -90:0.1:90
AF_dynamic = zeros(size(angles_range_AF))
for i = 1:length(angles_range_AF)
a_dynamic = exp(1j*2*pi*(0:N-1)'*d*sind(angles_range_AF(i)))
AF_dynamic(i) = abs(A_dynamic'*a_dynamic)
end

AF_dynamic = AF_dynamic / max(AF_dynamic)

figure
plot(angles_range_AF, 20*log10(AF_dynamic))
xlabel('Angle (Degrees)')
ylabel('Array Factor (dB)')
title('Dynamic Array Factor with Steered Main Lobe')
grid on
```