Antenna Array Processing

HW1

Mohammadreza Arani :::::::::::: 810100511

1401/07/26

```
clear; clc; close all;
```

Question 1: EEG Problem

```
Num_of_channel = 16;
K_timewin = 11;
T_win_sample = 100 ;


Num_of_sources_k  = 2+ abs((1:K_timewin)-6) ;

A = cell(1,K_timewin)  ;
S =A;  RS = A ; RL =A; RN = A; Noise = A; X=A;
```

```
    % Part 1:
for i=1:K_timewin
    A{i}  = randn(Num_of_channel , Num_of_sources_k(i)) ;
end
```

```
    % Part 2:

fs  = 20e3; % Sampling Freq
Ts = 1/fs;

for i=1:K_timewin
    f = (1:Num_of_sources_k(i))*1e3 ;
    t = ((i-1)*T_win_sample:i*T_win_sample-1)*Ts;
    S{i} = cos(2*pi*f'*t) ;
    RS{i}  = S{i}*S{i}'/T_win_sample ;


end

% To see RS:
disp(RS{randi(K_timewin)})
```

```
    0.5000   -0.0000    0.0000   -0.0000    0.0000
   -0.0000    0.5000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.5000    0.0000   -0.0000
   -0.0000    0.0000    0.0000    0.5000    0.0000
    0.0000    0.0000   -0.0000    0.0000    0.5000
```

We can see that as expected RS includes diagonal Matrix!

```
    % Part 3:

for  i = 1:K_timewin
    RL{i} = A{i}*RS{i}*A{i}.' ; % Quadratic form to formulate RL
    RL_rank = rank(RL{i});
    disp("Rank of RL{"+i+"}"+" = "+RL_rank+...
                " (M("+i+") = "+Num_of_sources_k(i)+" )")
end
```

```
Rank of RL{1} = 7 (M(1) = 7 )
Rank of RL{2} = 6 (M(2) = 6 )
Rank of RL{3} = 5 (M(3) = 5 )
Rank of RL{4} = 4 (M(4) = 4 )
Rank of RL{5} = 3 (M(5) = 3 )
Rank of RL{6} = 2 (M(6) = 2 )
Rank of RL{7} = 3 (M(7) = 3 )
Rank of RL{8} = 4 (M(8) = 4 )
Rank of RL{9} = 5 (M(9) = 5 )
Rank of RL{10} = 6 (M(10) = 6 )
Rank of RL{11} = 7 (M(11) = 7 )
```

As it apears, Rank of RL is the same as Num_of_sources used to produce the signal at interval K!

```
    % Part 4:

 Sigma_2 = rand()+1;
 rho = 0.2*Sigma_2;
 Rn = eye(Num_of_channel)*Sigma_2;
 Rn(1:end-1 , 2:end)  = Rn(1:end-1 , 2:end) + eye(Num_of_channel-1)*rho ;
 Rn(2:end , 1:end-1)  = Rn(2:end , 1:end-1) + eye(Num_of_channel-1)*rho ;
```

```
    % Part 5:
[W , Lambda]  = eig(Rn);

for i=1:K_timewin
    Noise{i} = repmat(sqrt( diag(Lambda) ), 1,T_win_sample).*randn(Num_of_channel , T_win_sampl
    Noise{i} = W*Noise{i};
    RN{i} = Noise{i}*Noise{i}.'/T_win_sample;
end
disp(RN{randi(max(Num_of_sources_k))})
```

```
  Columns 1 through 10

    1.6889    0.1801   -0.2608   -0.1725    0.1978   -0.0235   -0.1619   -0.0959   -0.0937    0.0336
    0.1801    1.1695    0.2838    0.0618   -0.0486   -0.0061   -0.0621    0.1000    0.0130    0.1510
   -0.2608    0.2838    1.1429    0.3825   -0.1802   -0.0952   -0.1284   -0.1032   -0.0297   -0.1236
   -0.1725    0.0618    0.3825    1.3825    0.3333    0.0556   -0.0971    0.0732   -0.1679   -0.0541
    0.1978   -0.0486   -0.1802    0.3333    1.4137    0.3458    0.0623   -0.0111   -0.0260   -0.0447
   -0.0235   -0.0061   -0.0952    0.0556    0.3458    1.3338    0.3687    0.1688    0.0125    0.1667
   -0.1619   -0.0621   -0.1284   -0.0971    0.0623    0.3687    1.1760    0.1710   -0.0875   -0.0042
```

```
  -0.0959    0.1000   -0.1032    0.0732   -0.0111    0.1688    0.1710    1.3697    0.2527    0.0440
  -0.0937    0.0130   -0.0297   -0.1679   -0.0260    0.0125   -0.0875    0.2527    1.4461    0.1009
   0.0336    0.1510   -0.1236   -0.0541   -0.0447    0.1667   -0.0042    0.0440    0.1009    1.1654
  -0.1164    0.1392    0.0198   -0.1345   -0.0143   -0.0732    0.2487   -0.0329    0.1390    0.2222
  -0.0419    0.1047    0.1104   -0.0487   -0.0020   -0.2179   -0.0985   -0.1072   -0.0920    0.1193
  -0.0230    0.0970    0.1034    0.0469   -0.0269   -0.0546   -0.1053    0.2106   -0.0506   -0.0684
   0.0896   -0.2300    0.0670    0.0465   -0.0798   -0.0273   -0.3523    0.0511    0.0528    0.1094
   0.1191   -0.1966    0.0303   -0.0164    0.1318    0.0512   -0.1892   -0.1714   -0.0684   -0.0904
  -0.0437    0.2013    0.1450   -0.1066   -0.1217    0.0569   -0.0790    0.1683   -0.2228    0.0192

  Columns 11 through 16

  -0.1164   -0.0419   -0.0230    0.0896    0.1191   -0.0437
   0.1392    0.1047    0.0970   -0.2300   -0.1966    0.2013
   0.0198    0.1104    0.1034    0.0670    0.0303    0.1450
  -0.1345   -0.0487    0.0469    0.0465   -0.0164   -0.1066
  -0.0143   -0.0020   -0.0269   -0.0798    0.1318   -0.1217
  -0.0732   -0.2179   -0.0546   -0.0273    0.0512    0.0569
   0.2487   -0.0985   -0.1053   -0.3523   -0.1892   -0.0790
  -0.0329   -0.1072    0.2106    0.0511   -0.1714    0.1683
   0.1390   -0.0920   -0.0506    0.0528   -0.0684   -0.2228
   0.2222    0.1193   -0.0684    0.1094   -0.0904    0.0192
   1.5561    0.1911   -0.1151   -0.2689    0.0854   -0.0325
   0.1911    1.4022    0.4477   -0.0866    0.1596    0.1556
  -0.1151    0.4477    1.3557    0.4048    0.0025    0.1117
  -0.2689   -0.0866    0.4048    1.5859    0.4122    0.0759
   0.0854    0.1596    0.0025    0.4122    1.3889    0.0800
  -0.0325    0.1556    0.1117    0.0759    0.0800    1.0995
```

```
% Try previous Part with a bigger T_Win:

T_win_sample_2  =1e6;

% Part 5:
[W , Lambda]  = eig(Rn);
Noise2 = Noise;
RN2 = RN;

for i=1:K_timewin
    Noise2{i} = repmat(sqrt( diag(Lambda) ), 1,T_win_sample_2).*randn(Num_of_channel , T_win_sa
    Noise2{i} = W*Noise2{i};
    RN2{i} = Noise2{i}*Noise2{i}.'/T_win_sample_2;
end
disp(RN{randi(max(Num_of_sources_k))})
```

```
  Columns 1 through 10

   1.3580    0.2190   -0.0930    0.0136   -0.0994   -0.1621   -0.1450   -0.0556    0.0331   -0.0006
   0.2190    1.3731    0.2923   -0.0056   -0.1175    0.0954    0.1227   -0.1847   -0.1794   -0.0683
  -0.0930    0.2923    1.5339    0.4065   -0.0640    0.0569   -0.0769   -0.3293   -0.4331   -0.2314
   0.0136   -0.0056    0.4065    1.2986    0.2251    0.1024    0.0151    0.0776   -0.1114   -0.1526
  -0.0994   -0.1175   -0.0640    0.2251    1.1478    0.3337   -0.1381    0.0526    0.0761   -0.0140
  -0.1621    0.0954    0.0569    0.1024    0.3337    1.2212    0.1232    0.0342    0.0696    0.1814
  -0.1450    0.1227   -0.0769    0.0151   -0.1381    0.1232    1.4009    0.4502   -0.0522    0.1658
  -0.0556   -0.1847   -0.3293    0.0776    0.0526    0.0342    0.4502    1.3729    0.1039    0.0771
   0.0331   -0.1794   -0.4331   -0.1114    0.0761    0.0696   -0.0522    0.1039    1.3945    0.4705
  -0.0006   -0.0683   -0.2314   -0.1526   -0.0140    0.1814    0.1658    0.0771    0.4705    1.5626
   0.1022   -0.0674   -0.2417    0.0733   -0.0660    0.1218    0.1486    0.0157    0.0132    0.4133
```

```
    0.2497    0.2302    0.0911    0.2345   -0.0463    0.0500   -0.1145   -0.0272   -0.1564    0.0334
   -0.1900   -0.0670    0.2362    0.0702   -0.1389   -0.0599   -0.1635   -0.1876   -0.1196    0.1360
   -0.1130   -0.0683   -0.0571    0.0143    0.0973    0.1019   -0.1451    0.0186    0.1145   -0.1588
   -0.1313   -0.2632   -0.0860    0.0274    0.0422   -0.1121   -0.0625    0.1643    0.0841   -0.3409
   -0.0714   -0.1102   -0.1573   -0.0755   -0.1085   -0.1635    0.0598    0.1210    0.0453   -0.0403

  Columns 11 through 16

    0.1022    0.2497   -0.1900   -0.1130   -0.1313   -0.0714
   -0.0674    0.2302   -0.0670   -0.0683   -0.2632   -0.1102
   -0.2417    0.0911    0.2362   -0.0571   -0.0860   -0.1573
    0.0733    0.2345    0.0702    0.0143    0.0274   -0.0755
   -0.0660   -0.0463   -0.1389    0.0973    0.0422   -0.1085
    0.1218    0.0500   -0.0599    0.1019   -0.1121   -0.1635
    0.1486   -0.1145   -0.1635   -0.1451   -0.0625    0.0598
    0.0157   -0.0272   -0.1876    0.0186    0.1643    0.1210
    0.0132   -0.1564   -0.1196    0.1145    0.0841    0.0453
    0.4133    0.0334    0.1360   -0.1588   -0.3409   -0.0403
    1.3394    0.5536   -0.0003   -0.1801   -0.0963    0.1416
    0.5536    1.5500    0.2051   -0.1645   -0.2083   -0.1655
   -0.0003    0.2051    1.4397    0.3364   -0.1074   -0.0941
   -0.1801   -0.1645    0.3364    1.3217    0.2891   -0.0067
   -0.0963   -0.2083   -0.1074    0.2891    1.5321    0.4132
    0.1416   -0.1655   -0.0941   -0.0067    0.4132    1.3150
```

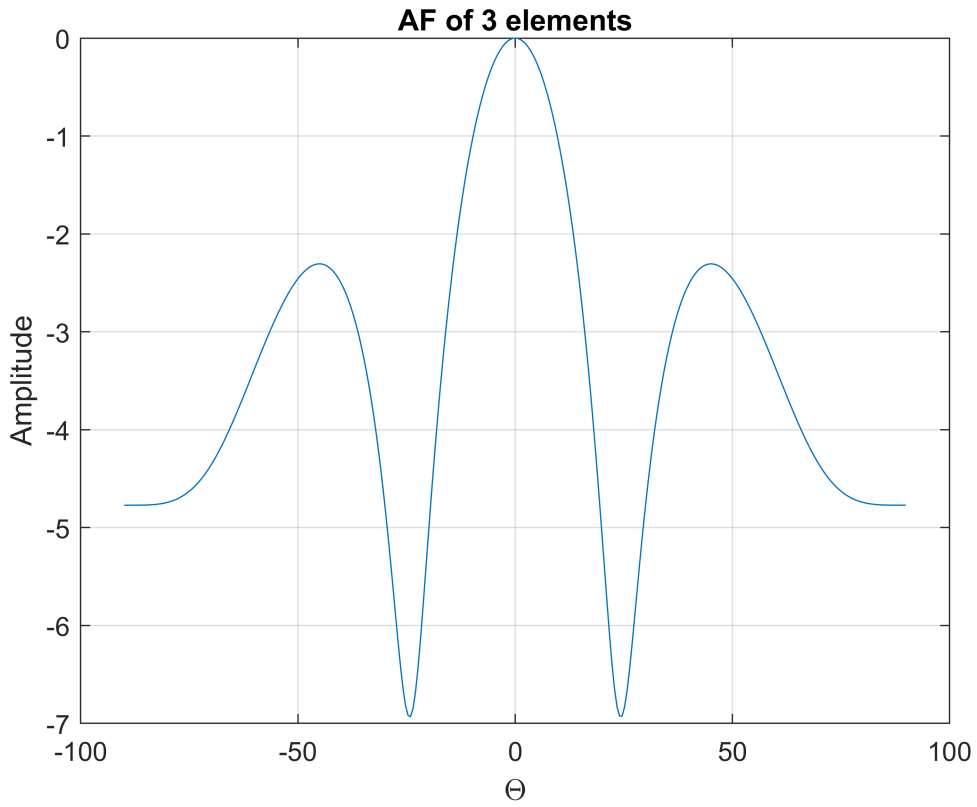
```
    % Part 6:

for i=1:K_timewin
        X{i} =  A{i}*S{i} + Noise{i} ;
end
```

Question 2: Antenna Array Problem

Part 1:

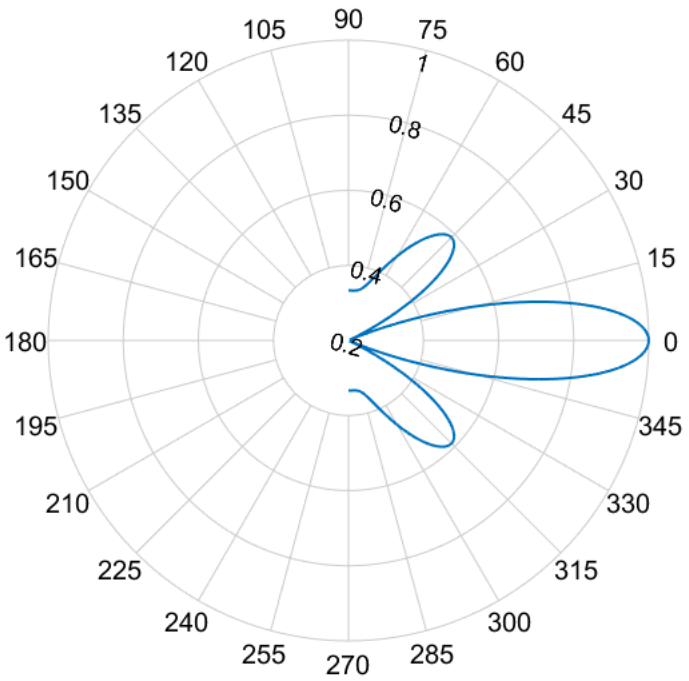
```
clear;

N = 3;  % a horizontal Array

d1 = 2; d2 = 4; d3 =5; % Height of each element:::meter
D=[d1,d2,d3];
W = ones(1,length(D)) ;

fc = 150e6; % 150MHz
c = 3e8;
k = 2*pi*fc/c ;

theta = -pi/2:0.01:pi/2 ;

AF  = abs(W*exp(1j*k*D'*sin(theta))  );

figure()
plot(180/pi*theta , db(sqrt(AF/max(AF) ) )    ) % AF with Normaliztion
```

```
grid on
title("AF of "+length(W)+" elements")
xlabel("\Theta")
ylabel("Amplitude")
```



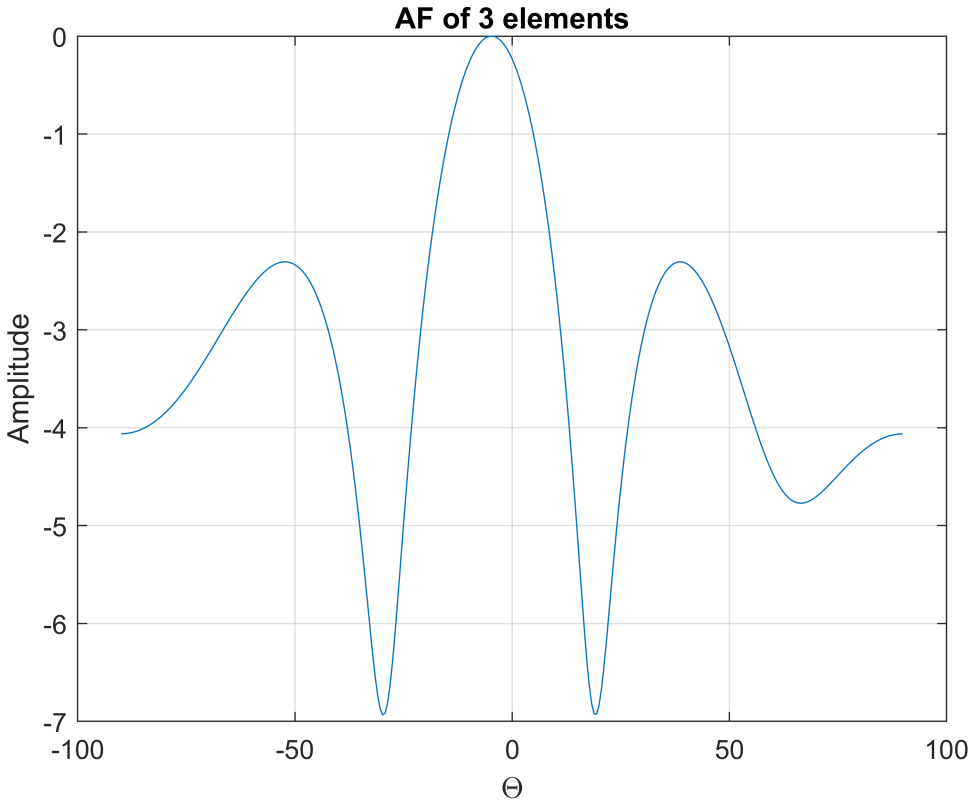

```
figure()
polarpattern(theta*180/pi,AF/max(AF));
```

Right click to interact with the plot



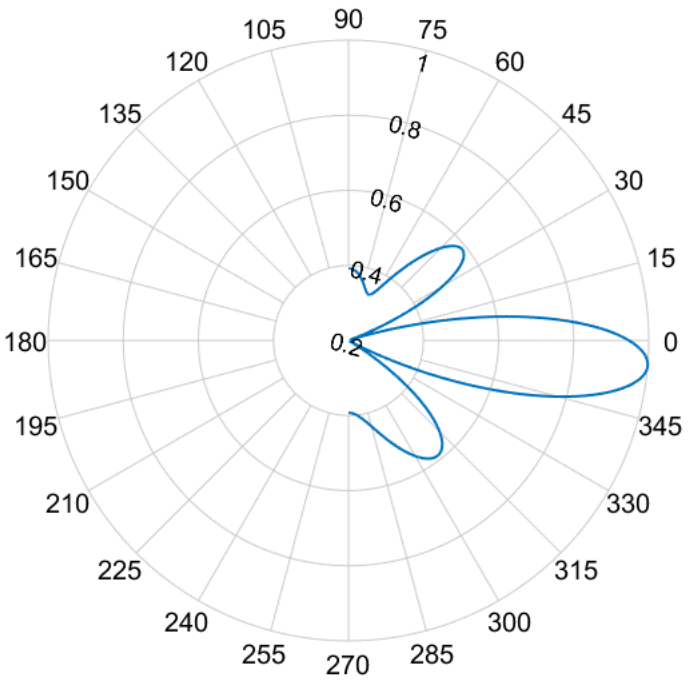
Part 2:

```
phase_diff = [0,30,45];

AF_2  = abs(exp(1j*pi*phase_diff/180).*W*exp(1j*k*D'*sin(theta))  );

figure()
plot(180/pi*theta , db(sqrt(AF_2/max(AF_2) ) )    ) % AF with Normaliztion
grid on
title("AF of "+length(W)+" elements")
xlabel("\Theta")
ylabel("Amplitude")
```




```
figure()
polarpattern(theta*180/pi,AF_2/max(AF_2));
```

Right click to interact with the plot



Part 3:

```
phi_2 = -90:1:90 ;
phi_3 = phi_2;

[PHI_2 , PHI_3] = meshgrid(phi_2,phi_3);

fun  = @(phi1,phi2) Array_factor(phi1,phi2)
```

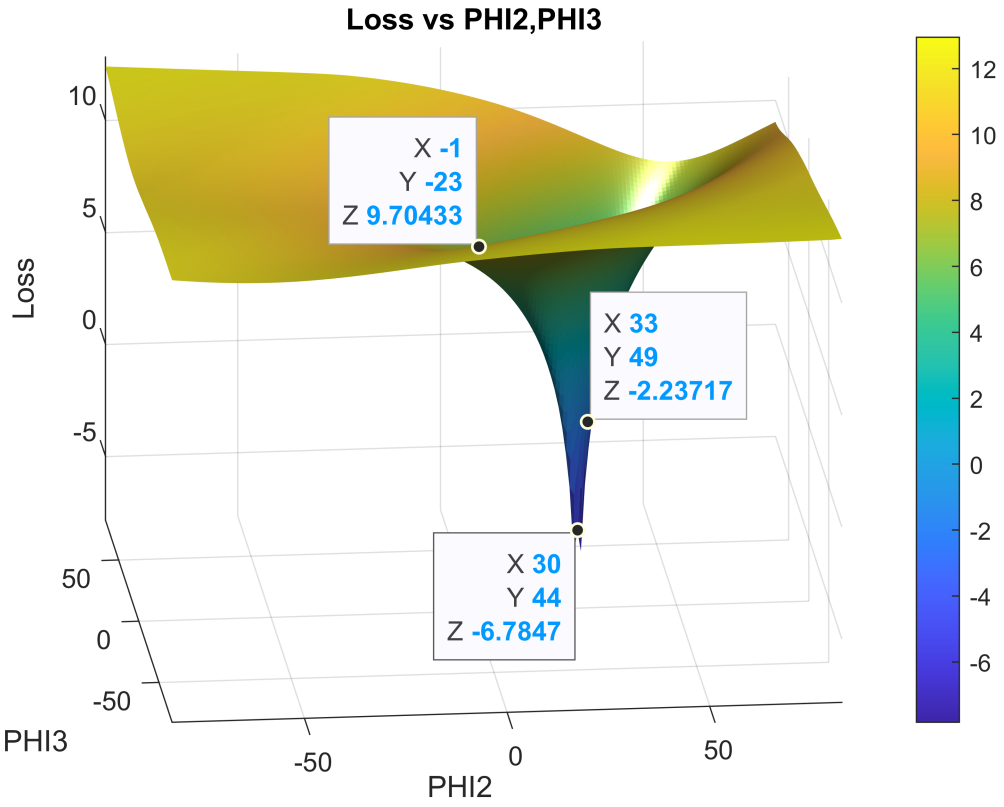
```
fun = function_handle with value:
    @(phi1,phi2)Array_factor(phi1,phi2)
```

```
figure()
h=surf( PHI_2  ,  PHI_3  ,  arrayfun( @(x,y)fun(x,y)  ,  PHI_2 ,PHI_3)      )
```

```
h =
  Surface with properties:

      EdgeColor: [0 0 0]
      LineStyle: '-'
      FaceColor: 'flat'
    FaceLighting: 'flat'
      FaceAlpha: 1
          XData: [181×181 double]
          YData: [181×181 double]
          ZData: [181×181 double]
```

```
            CData: [181×181 double]

   Show all properties
```

```
xlabel('PHI2');
ylabel('PHI3');
zlabel('Loss');
title("Loss vs PHI2,PHI3")
set(h,'LineStyle','none')
camlight
colorbar
```




```
fun(30,45)
```

```
ans = -Inf
```

It can be seen that the answer is unique for this problem, but it will not be assured to have a unique answer for other problems as well!

a limited 3 element array, produced an array factor and without changing the design and weights, we have steered the array so to change the direction of the pattern! =>

There will be just 1 answer for this problem due to stability of all other variables and changing just the array factor phase of 2 elements simulating a beam-steering!

```
function Loss = Array_factor(phi1,phi2)

    N = 3;  % a horizontal Array

    d1 = 2; d2 = 4; d3 =5; % Height of each element:::meter
    D=[d1,d2,d3];
    W = ones(1,length(D)) ;

    fc = 150e6; % 150MHz
    c = 3e8;
    k = 2*pi*fc/c ;
    theta = -pi/2:0.01:pi/2 ;
    phase_diff = [0,30,45];

    AF_2  = abs(exp(1j*pi*phase_diff/180).*W*exp(1j*k*D'*sin(theta))  );

   AF = abs(exp(1j*pi*[0,phi1,phi2]/180).*W*exp(1j*k*D'*sin(theta))  );
   Loss = db(sqrt(norm(AF - AF_2 ))); % L2 Norm
   %obj = x(1)*a + x(2)^2*b + c;
end

%%
```