Adaptive and Array Signal Processing
Homework 08

1. Implement the standard *ESPRIT* algorithm in *MATLAB*. To this end assume a linear uniform array and choose the selection matrices for the subarrays to get maximum overlap. Start with the line

   ```
   mu = Esprit(X,d)
   ```

   where `X` is the $M \times N$ matrix of the array measurements and `d` is the number of signal directions to look for. The return value `mu` is a vector containing the estimated spatial frequencies of the `d` wavefronts, sorted in ascending order.

   Hints:

   - To estimate the signal subspace *don't* use the SVD of `X` but rather the EVD of `X*X'/N` . In order to get the eigenvectors sorted use the following matlab routine

     ```
     function [V,D] = SortedEig(M)
     %[V,D] = SortedEig(M)
     %Computes eigenvectors and eigenvalues, sorted by
     %the magnitude of the eigenvalues. (Largest first)

     [V,D] = eig(M);
     d = diag(D);
     [ds,idx] = sort(abs(d));
     idx = idx(end:-1:1);
     V = V(:,idx);
     D = diag(d(idx));
     ```

   - Solve the invariance equation in the least-squares sense. To do this, utilize the pseudoinverse (`pinv`).

2. Let's see if it does compute.

   (a) First write a *MATLAB* function that will compute the array output vector of a uniform linear array. The function shall take as input the following parameters

   ```
   M   : number of antennas in the array
   mu  : Vector of spatial frequencies of the impinging wavefronts
   SNR : signal to noise ratio (uncorrelated white gaussian noise)
   N   : Number of snapshots to compute
   ```

   and return as output the matrix of array measurements. The signals are mutually uncorrelated and should be drawn from a zero mean, unity power complex Gaussian distribution. Start with the following line:

   ```
   function X = GetArrayOutput(M,mu,SNR,N)
   ...
   ```

   (b) Check your implementation of ESPRIT by first generating $N = 100$ snapshots of a $M = 12$ element ULA impinged by $d = 4$ uncorrelated, equally powered wavefronts from the directions $\mu \in \{-1, -0.5, 0.6, 1.5\} \cdot \mu_b$ with $\mu_b = \frac{2\pi}{M}$ beeing the standard beamwidth. (Use `GetArrayOutput` from the previous subtask). Assume a SNR of 20 dB. Does your algorithm seem to work? Increase the SNR to 80dB. The estimates should now be quite accurate (at least up to four digits or so). Can you confirm this?

(c) Now let's analyze the resolution capabilities of ESPRIT under practical conditions. Assume a ULA with $M = 3$ antenna elements beeing impinged by $d = 2$ uncorrelated, equally powered wavefronts from the directions $\mu \in \{0;\ \mu_k\}$, with a SNR $= 20$dB. For different values of μ_k the relative estimation error should be computed for different numbers of available snapshots. The μ_k shall be varied from $0.1 \cdot \mu_b$ to $1.0 \cdot \mu_b$ in 10 logarithmically equally spaced steps:

$$\mu_k = \mu_b \cdot \mathrm{e}^{\ln(0.1)\cdot(1-k/10)}; \quad 0 \leq k \leq 10$$

If these spatial directions are collected into a vector $\mathbf{m}_k = [\ 0\ \mu_k\ ]^T$ and the corresponding estimation results as returned from ESPRIT are put into the vector $\mathbf{m}_k^{(\mathrm{est})}$ the relative root mean squared error can be estimated as

$$\epsilon_k = \frac{1}{\mu_k}\sqrt{\frac{1}{R \cdot d}\sum_{r=1}^{R}\left|\left|\mathbf{m}_k - \mathbf{m}_k^{(\mathrm{est})}\right|\right|_2^2}$$

where R is the number of estimations that are performed for each value of k.

i. Write a *MATLAB* program that will plot the relative root mean squared error as a function of spatial separation μ_k for the number of available snapshots being $N \in \{100, 200, 500, 1000\}$. Assume SNR = 20dB.

Hints:

- Use the function `GetArrayOutput` to compute the snapshot matrix `X`
- Use `semilogy` to make the plot
- All four curves should be plotted into a single diagram
- Scale the μ-axis to μ_b such that it runs from 0.1 to 1.
- To get reliable results, the number of runs R should be 100 or greater. For $R = 100$, the time required for generating the plot may vary. It can take as few as a couple of seconds to up to a couple of minutes depending on the speed of the computer you are using.

ii. Suppose you have $N = 1000$ snapshots available (SNR=20dB) and must keep the relative root mean squared error below 2 percent. Using the graphs from above, to what spatial resolution does this lead?

Albert Iepure Matrikel_Nr: 03650065

Homework 08

1. The function called Esprit is listed below:

```
function mu = Esprit(X,d)

    [M,N] = size(X);
    [V,D] = SortedEig(X*X'/N);
    Us = V(:,1:d);
    J1 = [eye(M-1),zeros(M-1,1)];
    J2 = [zeros(M-1,1),eye(M-1)];
    Y = pinv(J1*Us)*J2*Us;
    [T,D] = eig(Y);
    mu = sort(angle(diag(D)));

end
```

It calls the functions SortedEig, which is listed below:

```
function [V,D] = SortedEig(M)

    [V,D] = eig(M);
    ds = diag(D);
    [ds,idx] = sort(abs(ds));
    idx = idx(end:-1:1);
    V = V(:,idx);
    D = diag(ds(idx));

end
```

2. A)The function GetArrayOutput is listed below:

```
function X = GetArrayOutput(M,mu,SNR,N)

    d = length(mu);
    X = exp(i*([0:M-1]')*mu')*randn(d,N) + sqrt(d)*(randn(M,N))*10^(-SNR/20);

end
```

The function called ESPRIT is shown below:

```
function mu = Esprit(X,d)

    [M,N] = size(X);
    [V,D] = SortedEig(X*X'/N);
    Us = V(:,1:d);
    J1 = [eye(M-1),zeros(M-1,1)];
    J2 = [zeros(M-1,1),eye(M-1)];
    Y = pinv(J1*Us)*J2*Us;
    [T,D] = eig(Y);
```

```
    mu = sort(angle(diag(D)));

end
```

b) The program that checks the implementation of ESPRIT is listed below:

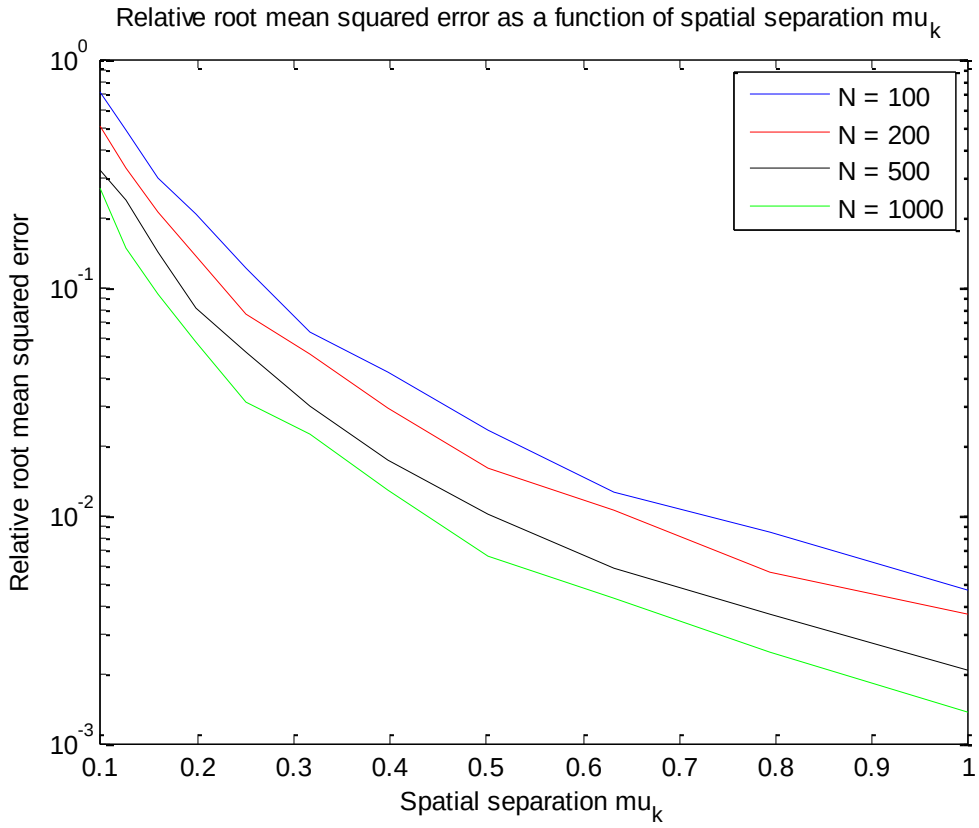
```
clear all

N = 100;
M = 12;
mu_b = 2*pi/M;
mu = [-1; -0.5; 0.6; 1.5]*mu_b

SNR = 20;
X = GetArrayOutput(M,mu,SNR,N);
mu_est1 = Esprit(X,length(mu))

SNR = 80;
X = GetArrayOutput(M,mu,SNR,N);
mu_est2 = Esprit(X,length(mu))
```

The real spatial frequencies and the estimated ones (for SNR = 20 dB and SNR = 80 dB) are listed below:

```
mu =

   -0.5236
   -0.2618
    0.3142
    0.7854


mu_est1 =

   -0.5339
   -0.2638
    0.3115
    0.7891


mu_est2 =

   -0.5236
   -0.2618
    0.3142
    0.7854
```

c) i. The program that plots the relative root mean squared error as a function of spatial separation mu_k for N ∈ {100, 200, 500, 1000} is listed below:

```
clear all

N = [100 200 500 1000];
M = 3;
mu_b = 2*pi/M;
SNR = 20;
R = 100;
d = 2;
mu_k = mu_b*0.1*logspace(0, 1, 11);

figure
c = ['b','r','k','g'];
for i = 1:length(N)
    for k = 1:length(mu_k)
        for r = 1:R
            m_k = [0; mu_k(k)];
            X = GetArrayOutput(M,m_k,SNR,N(i));
            m_k_est = Esprit(X,d);
            err(r) = norm(m_k-m_k_est)^2;
        end
        k
        m_k
        m_k_est
        e_k(k) = sqrt(mean(err)/d)/mu_k(k);
    end
    semilogy(mu_k/mu_b,e_k,c(i))
    hold on
end
legend('N = 100','N = 200','N = 500','N = 1000')
title('Relative root mean squared error as a function of spatial separation
mu_k ')
xlabel('Spatial separation mu_k')
ylabel('Relative root mean squared error')
```



ii) According to the figure below, using the data cursor, we can see that for a spatial separation of 0.3165 we get a relative root mean squared error of below two percent. So we can consider a spatial separation of at least 0.31 in order to obtain a root mean squared error below 2%.

Albert Iepure Matrikel_Nr: 03650065