Paper: Regular Sparse Array Direction of Arrival in One Dimension.

Ferre Knaepkens, Annie Cuyt, Wen-shin Lee, Dirk de Villiers

Regular Sparse Array Direction of Arrival in One Dimension

V.A. Example 1

Contents

Script environment

This script does depends on the random number generator state.

```
clear
close all
warning off;
```

V.A. Example 1

First we demonstrate the behaviour under influence of noise, mainly to illustrate the effect of the validation step. Therefore, we consider $n = 10$ received signals, given in Table I.. We use both standard ESPRIT and the new method with ESPRIT underlying to retrieve the 10 corresponding angles for increasing levels of noise. The additive noise is expressed in terms of SNR, which is defined by $20 \log_{10}(||f||_2/||\epsilon||_2)$, where $||f||_2$ and $||\epsilon||_2$ denote the 2-norm of respectively the sample and additive noise vector.

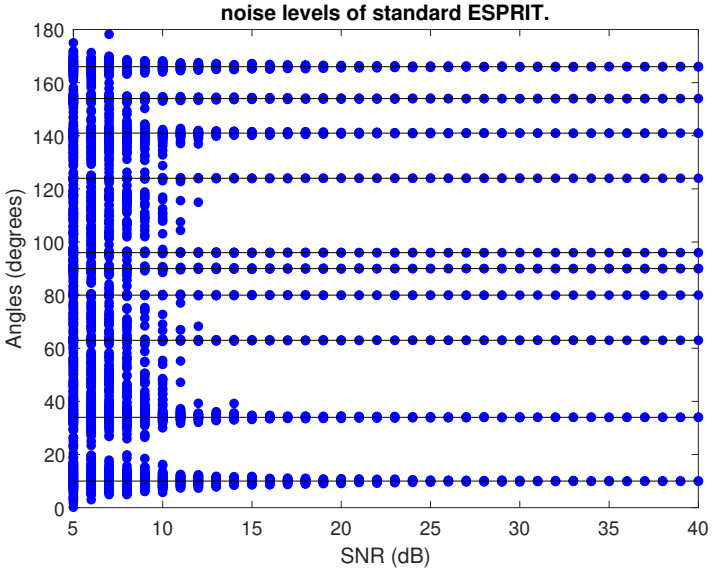
```
signal.ampl = [0.3, 0.2, 0.4, 0.5, 0.3,...
               0.4, 0.7, 0.2, 0.5, 0.4];
signal.phase = [0.9, 1.2, 0.8, 0.7, 1.1,...
                0.7, 1.3, 1.2, 1.0, 1.1]*pi;
signal.angles = deg2rad([10, 34, 63, 80, 90,...
                         96, 124, 141, 154, 166]);
```

For the standard ESPRIT approach, a ULA of 60 antennas with a distance of 0.48λ between the elements is considered. We also tell ESPRIT the correct number of signals, i.e. $n = 10$. We solve the DOA problem using ESPRIT on 256 snapshots. In Fig. 4 at the top, the results of all 256 snapshots are shown together. We observe that ESPRIT works well for a high SNR, however for

higher noise levels (approaching 10dB) the standard ESPRIT approach delivers unreliable results.

```
signal.freq = 1.5*1e6;
c = physconst('LightSpeed');
signal.dist = 0.48*(2*pi*c/signal.freq);
signal.nrelems = [60,0];
signal.rate = 1;
signal.shift = 0;
t = linspace(0,1,256);
K = numel(t);

bsolver = BSolverEsprit('--nsamples',60,'--nrows',41,...
                        '--ncols',20,'--nterms',10);

SNRvec = 5:40;
angles_ESPRIT = cell(numel(SNRvec),1);
SNR_ESPRIT = repmat(5:40,10*K,1);

wtbr = waitbar(0,'Please wait...');
for j = 1:numel(SNRvec)
    SNR = SNRvec(j);
    waitbar(j/numel(SNRvec),wtbr,sprintf('ESPRIT: SNR = %i',SNR));
    [signal.samples,~] = create_DOA_signal(signal,t,SNR);

    angles_ESPRIT{j} = zeros(K,10);
    for k = 1:K
        sgnl = Signal(1/signal.dist,signal.samples(k,:));
        PHI = bsolver.solve(sgnl);
        angles_ESPRIT{j}(k,:) = rad2deg(real(acos(1i*log(PHI)*c...
                                    /signal.dist/signal.freq)));
    end
    angles_ESPRIT{j} = angles_ESPRIT{j}(:);
end
close(wtbr)

fig_ESPRIT = figure;
plot(SNR_ESPRIT(:),cell2mat(angles_ESPRIT),'b.','MarkerSize',15)
hold on
plot(repmat([5;40],1,10),repmat(rad2deg(signal.angles),2,1),'k');
xlabel('SNR (dB)')
ylabel('Angles (degrees)')
title({['Fig. 4. (top) The solution of the DOA problem given by ',...
        'Table I for increasing'],['noise levels of standard ESPRIT.']})
```

Fig. 4. (top) The solution of the DOA problem given by Table I for increasing noise levels of standard ESPRIT.



At the same time, our new approach (with ESPRIT as underlying method) also uses 60 antennas in total: a first sparse ULA of 30 antennas and a shifted ULA of 30 antennas, with a scale and shift parameter of respectively $\sigma = 25$ and $\rho = 14$. The distance between the virtual dense array elements is also chosen as 0.48λ resulting in a total array size of more than 350λ. This might be an unrealistically large system for many applications, but the example serves to illustrate the efficacy of the proposed method even under such demanding conditions where both σ and ρ are large. As stated before, increasing σ results in a more difficult root intersection (validation) problem. Since the method detects the number of signals n automatically, n need not be passed to the algorithm. For the clusters C_σ, we choose the DBSCAN parameters $\mu = 0.8 \times 256 = 205$ with increasing equidistant δ values, namely $\delta = 0.01, 0.0825, 0.155, 0.2275, 0.3$. For the C_ρ clusters we take $\mu = 0.6 \times 256 = 154$ with $\delta = 0.5$, because they are usually less accurate, as already pointed out. At the bottom of Fig. 4 we find the results from 256 snapshots. For the lower noise levels, we clearly see that our method performs comparably. When the signals are perturbed with a lot of noise, we observe that the new method does not return all the angles, however, it also does not return unreliable results such as the stand-alone ESPRIT method. It detects that the signal is heavily perturbed, since the results are not validated by the cluster analysis and hence not all angles are recovered.

```
signal.freq = 1.5*1e6;
c = physconst('LightSpeed');
signal.dist = 0.48*(2*pi*c/signal.freq);
signal.nrelems = [30,30];
signal.rate = 25;
signal.shift = 14;
```

```
t = linspace(0,1,256);

cluster_spec.MinPts1 = 205;
cluster_spec.MinPts2 = 154;
cluster_spec.epsvec = [0.01,0.085,0.155,0.2275,0.3];
cluster_spec.eps2 = 0.5;

angles_new = cell(numel(SNRvec),1);
SNR_new = cell(numel(SNRvec),1);

wtbr = waitbar(0,'Please wait...');
for j = 1:numel(SNRvec)
    SNR = SNRvec(j);
    waitbar(j/numel(SNRvec),wtbr,sprintf('NEW: SNR = %i',SNR));
    [signal.samples1,signal.samples2] = create_DOA_signal(signal,t,SNR);
    angles_new{j} = DOAsolver(signal,cluster_spec,15);
    angles_new{j} = angles_new{j}(:);
    SNR_new{j} = repmat(SNR,numel(angles_new{j}),1);
end
close(wtbr)

fig_new = figure;
plot(cell2mat(SNR_new),rad2deg(cell2mat(angles_new)),'r.','MarkerSize',15)
hold on
plot(repmat([5;40],1,10),repmat(rad2deg(signal.angles),2,1),'k');
xlabel('SNR (dB)')
ylabel('Angles (degrees)')
title({['Fig. 4. (bottom) The solution of the DOA problem given by ',...
        'Table I for increasing'],['noise levels of the new method.']})
```

Fig. 4. (bottom) The solution of the DOA problem given by Table I for increasing noise levels of the new method.