# Paper: Sparse multidimensional exponential analysis with an application to radar imaging.

Annie Cuyt, Yuan Hou, Ferre Knaepkens, Wen-shin Lee

Sparse multidimensional exponential analysis with an application to radar imaging

4.1. 3-dimensional illustration of the new algorithm

## Contents


- Script environment
- 4.1. 3-dimensional illustration of the new algorithm


## Script environment

This script depends on the random number generator state.

```
clear
close all
```

## 4.1. 3-dimensional illustration of the new algorithm

To illustrate the method on a synthetic small-scale 3-D example, we consider the 29-term exponential expression (4.1) with $(x_j, y_j, z_j)$ and $\beta_j$ given in Table 1. We further set the following radar parameters:

$f_0 = 7.9$ GHz, $\quad \delta_f = 0.0015$ GHz,

$\theta_0 = \phi_0 = -0.024, \quad \delta_\theta = \delta_\phi = 3.75 \times 10^{-4}.$

We choose (for no specific reason, except that (2.2) and (2.8) need to be satisfied): $\Delta_1 = (1.17, 0.7, 1.87), \Delta_2 = (-1.00, -1.00, -1.00), \Delta_3 = (-2.10, 1.20, 3.29), N = 450, \nu = 150, \eta = 100, \mathrm{n} = 450, \kappa = 10, \mathrm{p} = 0.1$ and start the collection of the required samples. To each evaluation of (4.1) we add a Gaussian noise term of a fixed prechosen signal-to-noise ratio SNR (in dB). So

$F_s = f(s\Delta_1) + \epsilon_s, \quad s = 0, \ldots, M-1 := N + (\kappa - 1)\lfloor \mathrm{p}N \rfloor - 1$

$F_{s2} = f(s\Delta_1 + \Delta_2) + \epsilon_{M+s}, \quad s = 0, \ldots, \mathrm{m}-1 := \mathrm{n} + (\kappa - 1)\lfloor \mathrm{p}N \rfloor - 1$

$F_{s2} = f(s\Delta_1 + \Delta_2) + \epsilon_{M+s}, \quad s = 0, \ldots, \mathrm{m}-1 := \mathrm{n} + (\kappa - 1)\lfloor \mathrm{p}N \rfloor - 1$

With our choices for the parameters, we sample at 855 points in the direction of $\Delta_1$ and another 855 at each of the shifted locations $s\Delta_1 + \Delta_i, i = 2, 3$, or a total of 2565 points. This number is in sharp contrast with even the simplest grid structured data set of $35 \times 35 \times 35 = 42875$ points, where we choose 35 as a very mild overestimate of $n = 29$. In addition, a $d$-dimensional algorithm departing from a grid structured data set [16] does not offer any of the advantages we have discussed so far, among which:

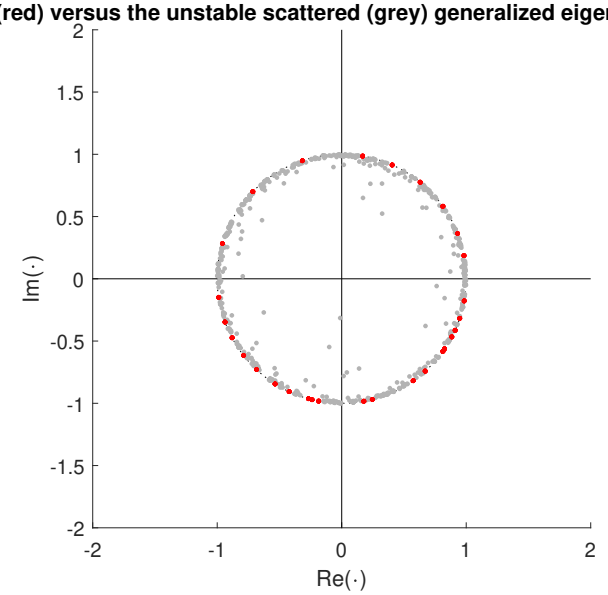
- the natural pairing of $\Phi_{ji}, i = 1, \ldots, d$ to $\Phi_j, j = 1, \ldots, n$,
- the automatic detection of the sparsity $n$, and
- the validation of the computed locations $(x_j, y_j, z_j)$.

In Figure 1 we show the DBSCAN result for SNR=10 dB with $m_\delta = \kappa - 2$ and $\delta$ varying over $5^\ell \times 10^{-4}, \ell = 0, \ldots, 4$: among the 1000 computed generalized eigenvalues ($\eta = 100, \kappa = 10$) 29 clusters are indicated in colour. They identify the stable generalized eigenvalues, detected and confirmed by the algorithm outlined in Section 2. None of the groups of $m_\delta$ associated values $\exp(\Phi_{ji}), j = 1, \ldots, 29, i = 2, 3$ exhibits a standard deviation larger than 0.25.

```
load('flighter29.mat')
nterms = numel(s_i);
nsamples = 450;
overlap = 0.9;
nwindow = 10;
total_samples = nsamples + ...
                   (nwindow-1)*ceil((1-overlap)*nsamples);

c = physconst('LightSpeed');


object_parms.loc = [flighter_x; flighter_y; flighter_z];
object_parms.ampl = s_i;


ISAR_parms.df = 0.0015e9;
ISAR_parms.dtheta = 3.75e-04;
ISAR_parms.dphi = 3.75e-04;
ISAR_parms.f0 = 7.9e9;
ISAR_parms.theta0 =  -0.024;
ISAR_parms.phi0 = -0.024;
ISAR_parms.fc = 8.4e9;

delta = [ 5,  3,  8 ;
       -1,  1, -1 ;
       -7,  4, 11];
```

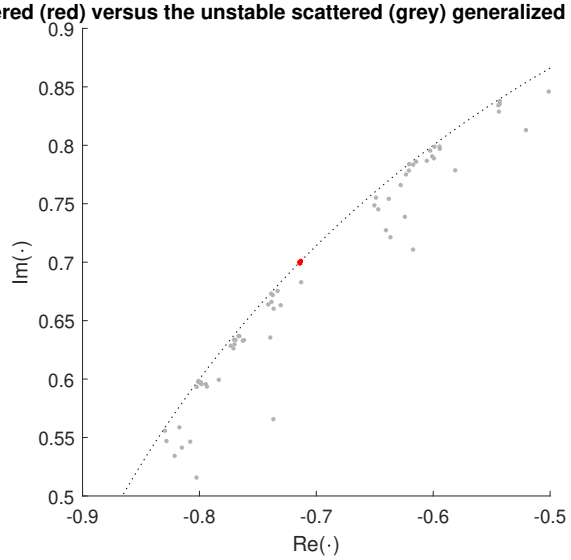
```
for k = 1:3
    bound = (1.05+0*rand)*max(4/c*abs(ISAR_parms.df*...
             object_parms.loc(1,:)*delta(k,1)...
            + ISAR_parms.fc*ISAR_parms.dtheta*...
             object_parms.loc(2,:)*delta(k,2)...
            + ISAR_parms.fc*ISAR_parms.dphi*...
             object_parms.loc(3,:)*delta(k,3)));
    delta(k,:) = delta(k,:)/bound;
end

ISAR_parms.delta = delta;


signal_parms.nsamples = nsamples;
signal_parms.SNR = 10;
signal_parms.overlap = overlap;
signal_parms.nwindow = nwindow;


data = create_signal(object_parms, ISAR_parms, signal_parms);


ESPRIT_parms.ncols = 150;
ESPRIT_parms.nrows = 301;
ESPRIT_parms.nterms = 100;

clust_parms.MinPts = ceil(0.9*signal_parms.nwindow);
clust_parms.epsvec = [0.0001,0.0005,0.0025];
clust_parms.disc_eps = 0.25;
clust_parms.lb = 0.7;
clust_parms.ub = 1.3;


loc = ISARsolver(data,ISAR_parms,ESPRIT_parms,clust_parms,true);


figure(1);
title({['Fig 1. The 29 clusters identified by DBSCAN (left) ',...
       'on the stable clustered'],['(red) versus the ',...
       'unstable scattered (grey) generalized eigenvalues']})

close(2); close(3);
```

**Fig 1. The 29 clusters identified by DBSCAN (left) on the stable clustered (red) versus the unstable scattered (grey) generalized eigenvalues**




```
figure(1)
xlim([-0.9,-0.5])
ylim([0.5,0.9])
title({['Fig 1. The 29 clusters identified by DBSCAN ',...
        'and a zoom (right) ',...
       'on the stable'],['clustered (red) versus the ',...
       'unstable scattered (grey) generalized eigenvalues']})
```

**Fig 1. The 29 clusters identified by DBSCAN and a zoom(right) on the stable clustered (red) versus the unstable scattered (grey) generalized eigenvalues**



We also run the above example for varying noise levels, from 40 dB SNR to 5 dB SNR, now with $\kappa = 20$ and each experiment repeated 100 times as the noise is randomly generated. In Figure 2 we show the average true cluster radius over the 29 scattering locations, for the generalized eigenvalues $\exp(\Phi_j)$ with $m_\delta = \kappa - 2$. This true radius is computed a posteriori with the exact $\exp(\Phi_j)$ in the center. In Figure 3 we respectively show at the left and the right for $i = 2, 3$ the average cluster radius over the 29 scattering locations, for the $\kappa - 5$ estimates closest to the true associated $\exp(\Phi_{ji})$. With $\kappa = 20$, a ratio of $\kappa - 5$ over the maximum number $m_\delta$ of associated elements still represents 83.3% of the associated values. In each of the Figures 2 and 3 we also show the smallest and largest cluster radius: they differ by a factor of about 2. It is quite clear that the computation of the $\exp(\Phi_j)$ is more accurate than that of the $\exp(\Phi_{ji})$. The estimates of the latter can be tightened but this is not really important at this point.

```
close(1)
SNRvec = 5:40;
nexp = 100;

b1 = exp( -4*pi*1i/c*( ISAR_parms.df*...
             object_parms.loc(1,:)*delta(1,1)...
         + ISAR_parms.fc*ISAR_parms.dtheta*...
             object_parms.loc(2,:)*delta(1,2)...
         + ISAR_parms.fc*ISAR_parms.dphi*...
             object_parms.loc(3,:)*delta(1,3) ) );
b2 = exp( -4*pi*1i/c*( ISAR_parms.df*...
             object_parms.loc(1,:)*delta(2,1)...
         + ISAR_parms.fc*ISAR_parms.dtheta*...
             object_parms.loc(2,:)*delta(2,2)...
         + ISAR_parms.fc*ISAR_parms.dphi*...
             object_parms.loc(3,:)*delta(2,3) ) );
b3 = exp( -4*pi*1i/c*( ISAR_parms.df*...
             object_parms.loc(1,:)*delta(3,1)...
         + ISAR_parms.fc*ISAR_parms.dtheta*...
             object_parms.loc(2,:)*delta(3,2)...
         + ISAR_parms.fc*ISAR_parms.dphi*...
             object_parms.loc(3,:)*delta(3,3) ) );

signal_parms.nwindow = 20;
signal_parms.SNR = [];

data = create_signal(object_parms, ISAR_parms, signal_parms);

bsolver = BSolverEsprit('--nsamples',nsamples,...
                                     '--nrows',301,...
```

```
                                    '--ncols',150,...
                                    '--nterms',100);
csolver = CSolverVandermondeLS('--nrows',nsamples,'--delta',1);

wtbr = waitbar(0,'Please wait...'); %waitbar
cntr = 0;
total_exp = numel(SNRvec)*nexp*signal_parms.nwindow;
results = struct;
for SNR = SNRvec
    B1_exp = cell(1,nexp);
    B2_exp = cell(1,nexp);
    B3_exp = cell(1,nexp);
    for m = 1:nexp
        for k = 1:signal_parms.nwindow
            cntr = cntr +1;
            waitbar(cntr/total_exp,wtbr,sprintf(['SNR = %d: ',...
                'experiment %d of %d \nwindow %d of %d'],...
                    SNR,m,nexp,k,signal_parms.nwindow));

            %exponential analysis
            signal1 = data{k}(1,:);
            signal2 = data{k}(2,:);
            signal3 = data{k}(3,:);
            noise = rand(3,nsamples) + 1i*rand(3,nsamples);
            signal1 = signal1 + (norm(signal1,'fro')...
                        /(10^(SNR/20)*norm(noise(1,:),'fro')))...
                        *noise(1,:);
            signal2 = signal2 + (norm(signal2,'fro')...
                        /(10^(SNR/20)*norm(noise(2,:),'fro')))...
                        *noise(2,:);
            signal3 = signal3 + (norm(signal3,'fro')...
                        /(10^(SNR/20)*norm(noise(3,:),'fro')))...
                        *noise(3,:);
            signal1 = Signal(1,signal1);
            signal2 = Signal(1,signal2);
            signal3 = Signal(1,signal3);

            B1=bsolver.solve(signal1);

            A1 = csolver.solve(signal1,B1);
            A2 = csolver.solve(signal2,B1);
            A3 = csolver.solve(signal3,B1);

            B2 = (A2./A1).';
            B3 = (A3./A1).';
```

```
            distmat = pdist2([real(b1(:)),imag(b1(:))],...
                             [real(B1(:)),imag(B1(:))]);
            [~,I] = min(distmat,[],2);
            B1_exp{m}(k,:) = B1(I);
            B2_exp{m}(k,:) = B2(I);
            B3_exp{m}(k,:) = B3(I);
        end
    end

    s = sprintf('SNR%i',SNR);
    S.B1_exp = B1_exp;
    S.B2_exp = B2_exp;
    S.B3_exp = B3_exp;
    results = setfield(results,s,S);
end
close(wtbr)


SNRnames = fieldnames(results);
K = numel(SNRnames);
SNRvalue = zeros(1,K);

radius = cell(1,3);
radius{1} = zeros(K,29);
radius{2} = zeros(K,29);
radius{3} = zeros(K,29);
for k = 1:K
    SNRvalue(k) = str2double(SNRnames{k}(4:end));
    S = getfield(results,SNRnames{k});
    M = numel(S.B1_exp);
    radius1 = zeros(M,29);
    radius2 = zeros(M,29);
    radius3 = zeros(M,29);
    for m = 1:M
        B1 = S.B1_exp{m};
        B2 = S.B2_exp{m};
        B3 = S.B3_exp{m};
        dist1 = abs(B1-repmat(b1,20,1));

        [val1,I1] = mink(dist1,18,1);
        radius1(m,:) = val1(end,:);

        dist2 = zeros(18,29);
        dist3 = zeros(18,29);
        for j = 1:29
```

```
            dist2(:,j) = abs(B2(I1(:,j),j)-b2(j));
            dist3(:,j) = abs(B3(I1(:,j),j)-b3(j));
        end

        [val2,~] = mink(dist2,15,1);
        radius2(m,:) = val2(end,:);
        [val3,~] = mink(dist3,15,1);
        radius3(m,:) = val3(end,:);
    end
    radius{1}(k,:) = mean(radius1);
    radius{2}(k,:) = mean(radius2);
    radius{3}(k,:) = mean(radius3);
end

[SNRvalue,I] = sort(SNRvalue);

figure
plot(SNRvalue,mean(radius{1}(I,:),2),'b-o')
hold on
plot(SNRvalue,min(radius{1}(I,:),[],2),'k--')
plot(SNRvalue,max(radius{1}(I,:),[],2),'k--')
title(['Fig. 2. Average radius of the exp(\Phi_j), ',...
       'j=1,...,29 clusters'])
xlabel('SNR')
ylabel('radius')
xlim([5,40])

figure
plot(SNRvalue,mean(radius{2}(I,:),2),'b-o')
hold on
plot(SNRvalue,min(radius{2}(I,:),[],2),'k--')
plot(SNRvalue,max(radius{2}(I,:),[],2),'k--')
title(['Fig. 3. Average radius of the exp(\Phi_{j2}), ',...
       'j=1,...,29 clusters (left)'])
xlabel('SNR')
ylabel('radius')
xlim([5,40])

figure
plot(SNRvalue,mean(radius{3}(I,:),2),'b-o')
hold on
plot(SNRvalue,min(radius{3}(I,:),[],2),'k--')
plot(SNRvalue,max(radius{3}(I,:),[],2),'k--')
title(['Fig. 3. Average radius of the exp(\Phi_{j3}), ',...
       'j=1,...,29 clusters (right)'])
xlabel('SNR')
```

```
ylabel('radius')
xlim([5,40])
```

**Fig. 2. Average radius of the exp($\Phi_j$), j=1,...,29 clusters**

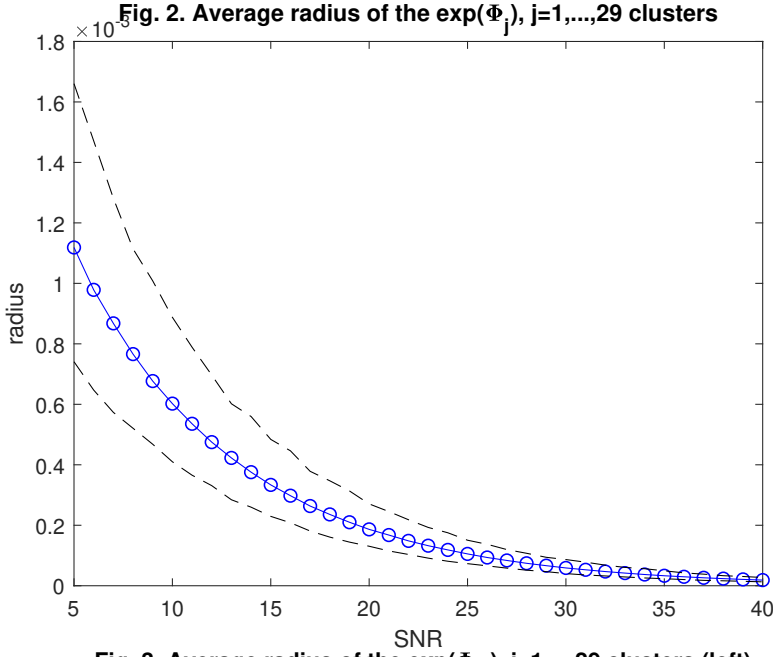



**Fig. 3. Average radius of the exp($\Phi_{j2}$), j=1,...,29 clusters (left)**

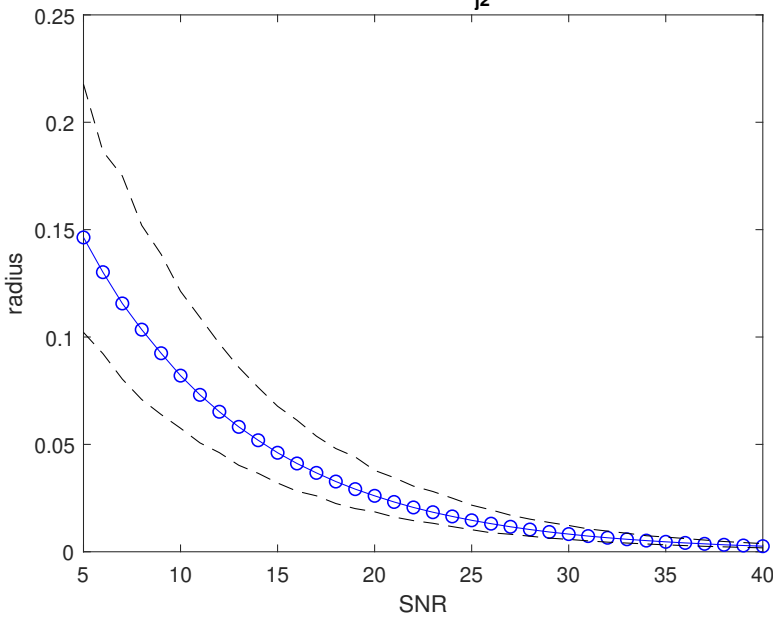
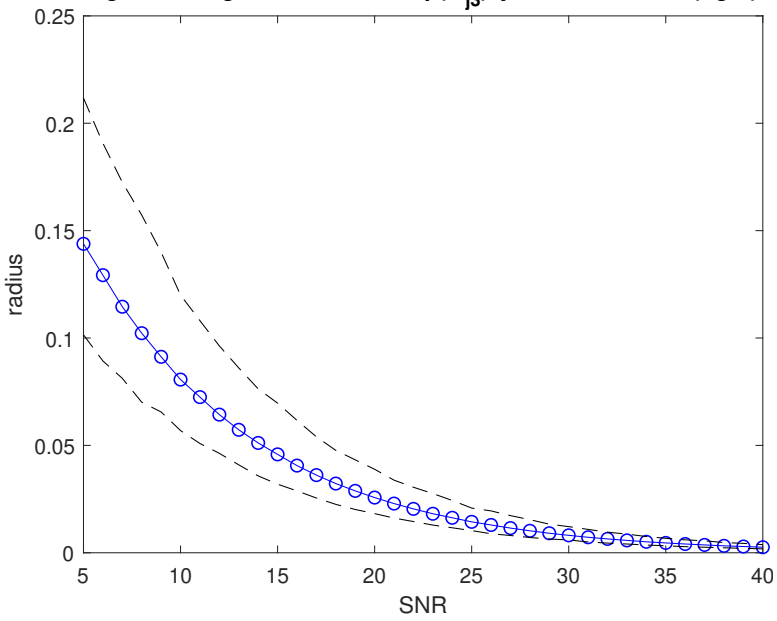



**Fig. 3. Average radius of the exp($\Phi_{j3}$), j=1,...,29 clusters (right)**