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

4.2. 2-dimensional illustration of the validation aspect

Contents

Script environment

This script depends on the random number generator state.

```
clear
close all
```

4.2. 2-dimensional illustration of the validation aspect

In another experiment we consider the 2-D example with 12 scattering centers (x_j, y_j) from Table 2. The dimension is reduced from three to two for the sole reason that in our figures we want to use the third dimension to graph the impact of the SNR. The radar parameters f_0, θ_0 and δ_f, δ_θ are as in Section 4.1. We further take $N = 150, \nu = \eta = 50, \mathrm{n} = 100, \kappa = 11, \mathrm{p} = 0.054$ and $\Delta_1 = (1.38, 4.14), \Delta_2 = (-7.56, 5.67)$.

In order to reduce the cluster radius in the shift direction Δ_2 we perform the shift a number of times, over $\Delta_2, 2\Delta_2, \ldots, 8\Delta_2$ and combine the results, since for s fixed,

$f(s\Delta_1 + m\Delta_2) = \sum_{j=1}^{n} \alpha_j \exp(\langle \phi_j, m\Delta_2 \rangle) \exp(\langle \phi_j, s\Delta_1 \rangle)$.

So the coefficients extracted from the different shifts are

$\alpha_j \exp(\langle \phi_j, \Delta_2 \rangle), \alpha_j \exp(\langle \phi_j, 2\Delta_2 \rangle), \ldots, \alpha_j \exp(\langle \phi_j, 8\Delta_2 \rangle)$.

The total number of collected samples then adds up to $(\kappa - 1)\lfloor \mathrm{p}N \rfloor + N$ in the Δ_1 direction and $8 \times ((\kappa - 1)\lfloor \mathrm{p}N \rfloor + \mathrm{n})$ in the Δ_2 shifts, or 1670 samples altogether.

In Figure 4 (right) we show the result of the computations, after applying `DBSCAN` with $m_\delta = \kappa - 1$ and $\delta = 0.00001, 0.002505, 0.005$ to each SNR result for the Φ_j and discarding cluster results when the standard deviation of the Φ_{j2} exceeds 0.2. We let the SNR vary from 40 dB to 5 dB, top to bottom. The SNR=10 dB slice is presented in Figure 5 (right) and a separate coordinate view is found in Figure 6 (right) where the SNR varies from right to left. For each SNR the experiment is repeated 250 times.

We compare these results to the output in the Figures 4 (left), 5 (left) and 6 (left) of the 2-D Prony-like algorithm `MEMP` [16] using the same number of samples but now laid out in a grid of size 40×42. We remark the main differences with the new algorithm:

- the matching in the MEMP algorithm between results computed in separate dimensions is definitely not flawless, and as the noise increases erroneous combinations give rise to inexistent locations;
- the matching through the indexing of the variables in (2.6) and (2.9) leaves no room for error and so does not introduce matching errors;
- for increasing noise, meaning decreasing SNR, the unvalidated MEMP algorithm may return a few erroneous (x_j, y_j), despite the fact that the sparsity $n = 12$ was passed to the algorithm as well;
- the correct sparsity $n = 12$ need not be passed to the new algorithm, which detects it automatically as the number of identified and confirmed clusters;
- in the new algorithm the results for very small SNR are either somewhat less accurate or absent because of the high validation requirement, which can of course be relaxed by the user.

```
flighter_x = [0,1,2,2,2,2.5,2.5,3,3,3,4,5];
flighter_y = [4,3.5,5,4,3,2,1,5,4,3,3.5,4];
s_i = repmat(50,1,12);

c = physconst('LightSpeed');

nterms = numel(s_i);
object_parms.loc = [flighter_x; flighter_y];
object_parms.ampl = s_i;

ISAR_parms.df = 0.0015e9;
ISAR_parms.dtheta = 3.75e-04;

ISAR_parms.f0 = 7.9e9;
ISAR_parms.theta0 =  -0.024;
ISAR_parms.fc = 8.4e9;

delta = [ 3,  9;
```

```
       -4,  3];

for k = 1:2
    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)));
    delta(k,:) = delta(k,:)/bound;
end

ISAR_parms.delta = delta;

signal_parms.nsamples = [150,100];
signal_parms.SNR = 50;
signal_parms.overlap = 0.95;
signal_parms.nwindow = 11;
signal_parms.nshift = 8;

ESPRIT_parms.ncols = 50;
ESPRIT_parms.nrows = 101;
ESPRIT_parms.nterms = 50;

clust_parms.MinPts = 10;
clust_parms.epsvec = linspace(0.00001,0.005,3);
clust_parms.disc_eps = 0.2;
clust_parms.lb = 0;
clust_parms.ub = inf;

wtbr = waitbar(0,'Please wait...');
SNRvec = 5:40;
T = 0;

nrexp = [repmat(250,1,16),repmat(100,1,10),repmat(10,1,10)];
count = 0;
total_count = sum(nrexp);

results = struct;
for k = 1:numel(SNRvec)
    SNR = SNRvec(k);


    signal_parms.SNR = SNR;
    loc = cell(1,nrexp(k));
    for j = 1:nrexp(k)
        tic
        waitbar(count/total_count,wtbr,...
```

```
            sprintf(['SNR %d of %d: experiment %d of %d\n',...
                     'Previous run time: %fs'],SNR, ...
                      max(SNRvec),j,nrexp(k),T));
        data = create_signal_2D(object_parms, ISAR_parms,...
                                signal_parms);

        loc{j} = ISARsolver_2D(data,ISAR_parms,ESPRIT_parms,...
                               clust_parms,false);
        T = toc;
        count = count+1;
    end

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



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

loc = [];
Z = [];

for k = 1:K
    SNRvalue(k) = str2double(SNRnames{k}(4:end));
    S = getfield(results,SNRnames{k});
    loc = [loc, cell2mat(S.loc)];
    Z = [Z,repmat(SNRvalue(k),1,size(cell2mat(S.loc),2))];
end

figure
lw = 1;
plot3([0,0],[4,4],[5,40],'k','LineWidth',lw)
hold on
plot3([1,1],[3.5,3.5],[5,40],'k','LineWidth',lw)
plot3([2,2],[5,5],[5,40],'k','LineWidth',lw)
plot3([2,2],[4,4],[5,40],'k','LineWidth',lw)
plot3([2,2],[3,3],[5,40],'k','LineWidth',lw)
plot3([2.5,2.5],[2,2],[5,40],'k','LineWidth',lw)
plot3([2.5,2.5],[1,1],[5,40],'k','LineWidth',lw)
plot3([3,3],[5,5],[5,40],'k','LineWidth',lw)
plot3([3,3],[4,4],[5,40],'k','LineWidth',lw)
```

```
plot3([3,3],[3,3],[5,40],'k','LineWidth',lw)
plot3([4,4],[3.5,3.5],[5,40],'k','LineWidth',lw)
plot3([5,5],[4,4],[5,40],'k','LineWidth',lw)
scatter3(loc(1,:),loc(2,:),Z,15,'filled','MarkerFaceAlpha',...
        .3,'MarkerFaceColor','r')
axis([-1 6 0 6 5 40])
view([33,40])
title(['Fig. 4: Validated (x_j,y_j) locations from the new ',...
       'algorithm (right).'])
grid on

figure
scatter(loc(1,Z==10),loc(2,Z==10),15,'filled',...
        'MarkerFaceAlpha',.3,'MarkerFaceColor','r')
grid on
xlim([-0.5,5.5])
ylim([0,6])
xticks(-1:6)
yticks(0:6)
pbaspect([1 1 1])
title({['Fig. 5: Slice of Figure 4 (SNR=10) with the ',...
       'intensity indicating the frequency'],['of detection ',...
       'in the 250 runs.']})

figure
subplot(2,1,1)
plot([5,40],[0,0],'k')
hold on
plot([5,40],[1,1],'k')
plot([5,40],[2,2],'k')
plot([5,40],[2.5,2.5],'k')
plot([5,40],[3,3],'k')
plot([5,40],[4,4],'k')
plot([5,40],[5,5],'k')
scatter(Z,loc(1,:),30,'filled','MarkerFaceAlpha',.3,...
        'MarkerFaceColor','r')
ylim([-1,6])
title(['Fig. 6: Validated x_j and y_j coordinates from ',...
       'the new algorithm (right).'])
subplot(2,1,2)
plot([5,40],[1,1],'k')
hold on
plot([5,40],[2,2],'k')
plot([5,40],[3,3],'k')
plot([5,40],[3.5,3.5],'k')
plot([5,40],[4,4],'k')
```

```
plot([5,40],[5,5],'k')
scatter(Z,loc(2,:),30,'filled','MarkerFaceAlpha',.3,...
        'MarkerFaceColor','r')
ylim([0,6])
```

Fig. 4: Validated (x_j,y_j) locations from the new algorithm (right).

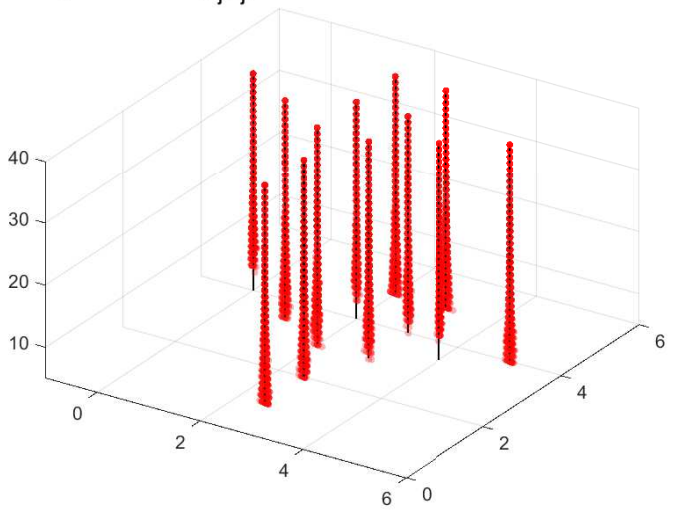

Fig. 5: Slice of Figure 4 (SNR=10) with the intensity indicating the frequency of detection in the 250 runs.

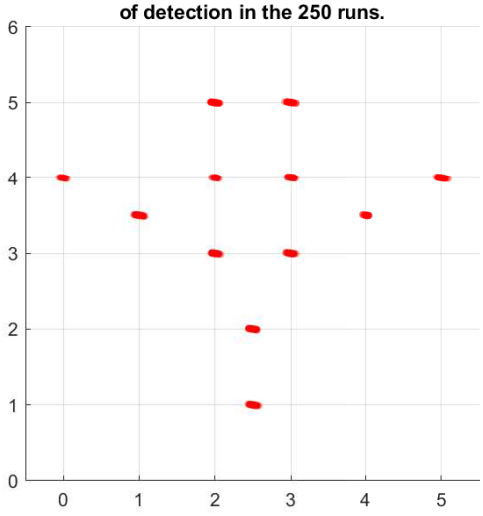

Fig. 6: Validated x_j and y_j coordinates from the new algorithm (right).