Paper: Validated exponential analysis for harmonic sounds.pdf

Matteo Briani, Annie Cuyt, and Wen-shin Lee

Validated exponential analysis for harmonic sounds

Example Figure 1

Contents

Script environment

This script does not depend on the random number generator state.

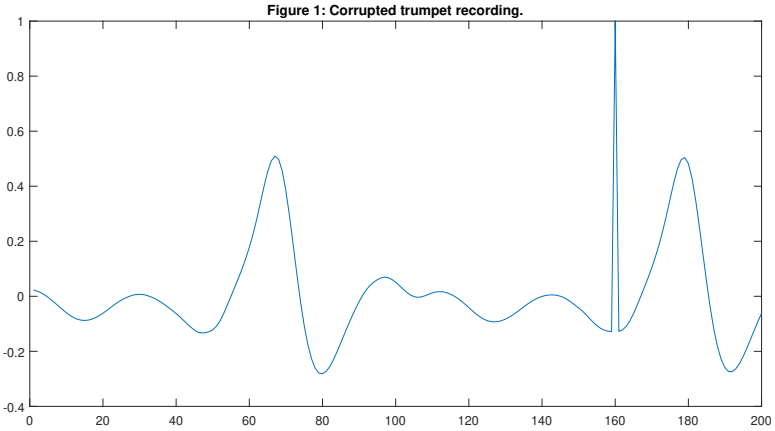
```
clear
close all
```

Example Figure 1

To illustrate the validation aspect and how it is robust in the presence of outliers, we consider 400 audio samples of the sustained part of an A4 note played by a trumpet, corrupted by an outlier as shown in Figure 1.

```
[piece_total,Fs] = audioread('trumpet-a4.wav');
piece = piece_total(18003:18402);
piece(160) = 1;
signal = Signal(Fs,piece);

figure
plot(signal.samples(1:200))
title('Figure 1: Corrupted trumpet recording.')
set(gcf,'Position',[50 100 1000 500])
```



We put $k = 4$ and $\kappa = 3$ and compare the validation to a standard implementation of ESPRIT. As can be seen in the ESPRIT reconstruction in Figure 1 (top), it suffers from the presence of the outlier, as any parametric method would. The new method, illustrated in Figure 1 (bottom), deals with k decimated signals instead of the full signal and is more robust. In both approaches we choose $n = 20$. While the ESPRIT algorithm deals with a Hankel matrix of size 260×141, the cluster analysis only needs Hankel matrices of size 70×30. When the recording is corrupted by an outlier, here only one of the k decimated signals is affected. The decimated sample set that contains the outlier does not contribute to the formed clusters. But the cluster algorithm still detects clusters composed of at least $k?1$ eigenvalues at the correct locations λ_i^k. Since one easily identifies the decimated signal that did not contribute to all clusters, the equations coming from that set of samples and contributing to (4) for the computation of the α_i, are best removed from the Vandermonde system.

```
bsolver_esprit = BSolverEsprit('--nsamples',400,'--nrows',260,...
                        '--ncols',141,'--nterms',20);

bsolver_vexpa  = BSolverVexpa( '--bsolver',   BSolverEsprit( ...
                                              '--ncols', 30, ...
                                              '--nterms',20) ...
                       , '--csolver',   CSolverVandermondeLS ...
                       , '--rate'   ,   4 ...
                       , '--shift'  ,   3 ...
                       , '--M'      ,   15 ...
                       , '--u-epsilon', 0.01 ...
                       , '--u-minpts' , 3 ...
                       , '--s-epsilon', 0.05 ...
                       , '--s-minpts' , 2 ...
                       , '--plot'   , false ...
                       , '--time'   , false ...
                       );
```

```
csolver = CSolverVandermondeLS('--nrows',400,'--delta',1);


B_esprit = bsolver_esprit.solve(signal);
C_esprit = csolver.solve(signal,B_esprit);
parms_esprit = MultiExponentialParameters(Fs,...
               {B_esprit,C_esprit},'normalized');
signal_esprit = parms_esprit.construct(400);

figure
plot(signal.samples(1:200))
hold on
plot(real(signal_esprit.samples(1:200)),'o')
title(['Figure 1 (top,circles): Corrupted trumpet recording ',...
       'reconstructed by ESPRIT.'])
set(gcf,'Position',[50 100 1000 500])


B_vexpa = bsolver_vexpa.solve(signal);
C_vexpa = csolver.solve(signal,B_vexpa);
parms_vexpa = MultiExponentialParameters(Fs,...
               {B_vexpa,C_vexpa},'normalized');
signal_vexpa = parms_vexpa.construct(400);

figure
plot(signal.samples(1:200))
hold on
plot(real(signal_vexpa.samples(1:200)),'+')
title(['Figure 1 (bottom, crosses): Corrupted trumpet ',...
       'recording reconstructed by the new method.'])
set(gcf,'Position',[50 100 1000 500])
```

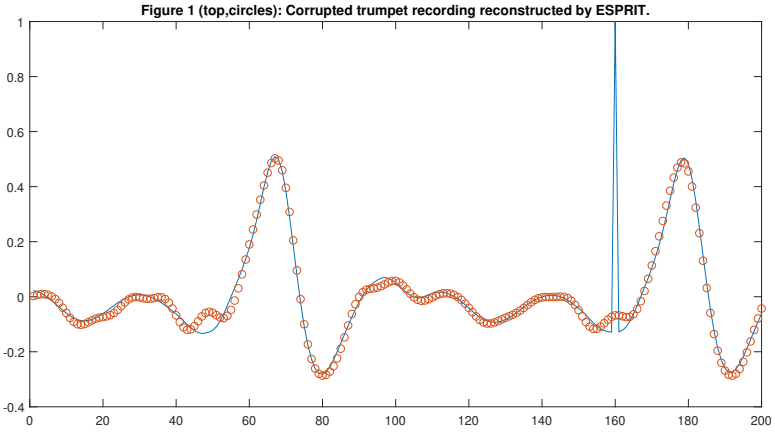


Figure 1 (bottom, crosses): Corrupted trumpet recording reconstructed by the new method.