Paper: Validated exponential analysis for harmonic sounds.pdf

Matteo Briani, Annie Cuyt, and Wen-shin Lee

Validated exponential analysis for harmonic sounds

Illustration on a harmonic sound

Contents

Script environment

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

```
clear
close all
```

Illustration on a harmonic sound

We consider a recorded sound of a guitar playing a D3 note corrupted by electrical humming [32], downloaded from the website freesound.org. The samples are collected at a rate of 48 kHz, for a duration of about 9 seconds in total (454 071 sample points t_j). We apply the method described above to audio windows of 1024 samples, with an overlap of 75% between the windows. The goal is to extract the sinusoidal tracks [33] that form the guitar partials. We choose the downsampling factor $k = 5$ and take $\kappa = 7$ (other combinations work as well, of course). So in each window the downsampled set contains 204 samples, which we use to extract $\nu = 61$ generalized eigenvalues, leaving us to Hankel matrices of size (at most) 143×61. For the solution of (7) we use the ESPRIT algorithm [31]. After superimposing the $k = 5$ analyses of the downsampled audio windows, a cluster analysis using DBSCAN is performed for each window, thus retrieving the generalized eigenvalues λ_i^k most accurately.

To illustrate the regularization effect on the rectangular 143×61 analogon of (2) from choosing $k > 1$, we show in Figure 2 the distribution of the generalized eigenvalues λ_i, $i = 1, \ldots, n$ of the full not downsampled 60-th windows starting at t_{15104} opposed to that of the λ_i^5, $i = 1, \ldots, n$ of the downsampled set of samples from the same window.

In the wake of the shift strategy discussed in Section 5 (we merely took $h = 1, \ldots, 24 < H$), the value λ_i^κ can further be improved. After performing the

cluster analysis on the superimposed results for λ_i^κ, we can look at the λ_i^κ associated with each of these and compute their center of gravity (disregarding those that fall out of scope). Note that for the λ_i^κ no separate cluster analysis needs to be performed. The latter is illustrated in Figure 3.

Since the technique is being applied to a harmonic sound, an additional step can be performed to estimate the base frequency (per window) more accurately. Once the stable frequencies ϕ_i, $i = 1, \ldots, n$ are retrieved, we look for a harmonic relation between them. We divide every detected harmonic partial \phi_i$ by the integers $j = 1, \ldots, 40$ (which is the largest number of partials expected) and we add these quotients to the discovered ϕ_i, in this way creating a new larger cluster at the base frequency, which we call ϕ_1. The center of gravity of this larger cluster estimates the lowest partial of the harmonics. Using this estimate of the base frequency ϕ_1, all higher harmonic partials $j\phi_1$, $j =?60, \ldots, 60$ are reconstructed and substituted in one large rectangular 512×121 Vandermonde system (4), which serves as the coefficient matrix for the computation of the α_i, $i = 1, \ldots, 121$.

While moving from one window to the next over the course of the 9 seconds, the higher harmonic partials that are detected become weaker and fewer. So n decreases with time. We refer to Figure 4, where we again show the generalized eigenvalues λ_i and λ_i^5, before and after regularization, now for one of the middle audio windows. Fortunately, the number of partials remains large enough during the whole audio fragment to rebuild the harmonics as described. Since the final reconstructed guitar sound only makes use of the ϕ_i from the stable generalized eigenvalues, the reconstruction does not suffer from the electrical hum anymore.

```
[piece_total,Fs] = audioread('guitar-d3.wav');

N = 1024;
overlap = 0.75;
step = (1-overlap)*N;
nwindows = floor((length(piece_total)-1024)/step)+1;

piece_recon = zeros(1,(nwindows-1)*step+N);
div = [ones(1,step), repmat(2,1,step), repmat(3,1,step),...
       repmat(4,1,(nwindows-3)*step), ...
       repmat(3,1,step), repmat(2,1,step), ones(1,step)];

for k = 1:nwindows

    if ~mod(k,25)
        fprintf('\n Window %i of %i',k,nwindows)
    end
    signal = Signal(Fs,piece_total((k-1)*step+1:(k-1)*step+N));
```

```
bsolver  = BSolverVexpa( '--bsolver',   BSolverEsprit( ...
                                          '--ncols', 61, ...
                                          '--nterms',61) ...
                    , '--csolver',   CSolverVandermondeLS ...
                    , '--rate'   ,   5 ...
                    , '--shift'  ,   7 ...
                    , '--M'      ,   24 ...
                    , '--u-epsilon', 0.0005 ...
                    , '--u-minpts' , 4 ...
                    , '--s-epsilon', 0.001 ...
                    , '--s-minpts' , 4 ...
                    , '--plot'   , false ...
                    , '--time'   , false ...
                    );

if k == 60
        bsolver  = BSolverVexpa( '--bsolver',   BSolverEsprit( ...
                                          '--ncols', 61, ...
                                          '--nterms',61) ...
                    , '--csolver',   CSolverVandermondeLS ...
                    , '--rate'   ,   5 ...
                    , '--shift'  ,   7 ...
                    , '--M'      ,   24 ...
                    , '--u-epsilon', 0.0005 ...
                    , '--u-minpts' , 4 ...
                    , '--s-epsilon', 0.001 ...
                    , '--s-minpts' , 4 ...
                    , '--plot'   , true ...
                    , '--time'   , false ...
                    );
end

csolver = CSolverVandermondeLS('--nrows',512,'--delta',1);

B = bsolver.solve(signal);
B = B./abs(B);

if k == 60
    for j = 1:numel(B)
        close(j+2);
    end
    figure(1)
    title(['Figure 3 (left): Clusters of \lambda_i^5 ',...
           'values.'])
    figure(2)
```

```
    title(['Figure 3 (right): Clusters of \lambda_i^7 ',...
           'values.'])
    plot_base_terms(B);
    title(['Figure 2 (left): Generalized eigenvalues ',...
           '\lambda_i from the 60-th window.'])
    plot_base_terms(B.^5);
    title(['Figure 2 (right): Generalized eigenvalues ',...
           '\lambda_i^5 from the 60-th window.'])
elseif k == 885
    plot_base_terms(B);
    title(['Figure 4 (left): Generalized eigenvalues ',...
           '\lambda_i from a middle window.'])
    plot_base_terms(B.^5);
    title(['Figure 4 (right): Generalized eigenvalues ',...
           '\lambda_i^5 from a middle window.'])
end

if numel(B) > 1
    freq = imag(log(B)*Fs)/(2*pi);
    minfreq = min(freq(freq > 10));
    freq_partials = freq./((1:40)');
    freq_partials = freq_partials(:);
    freq_partials(freq_partials < max([0.9*minfreq,0])) = [];
    [IDX,~] = DBSCANm(freq_partials,1,ceil(0.35*numel(B)));
    IDXj = zeros(1,max(IDX));
    for j = 1:max(IDX)
        IDXj(j) = sum(IDX==j);
    end
    IDXj(IDXj > numel(B)/2) = 0;
    [~,I] = max(IDXj);
    if isempty(I) || IDXj(I) == 0
        B = 0;
        C = 0;
    else
        freq1 = mean(abs(freq_partials(IDX==I)));
        B = exp(2*pi*1i*(freq1*(-60:60))/Fs);
        C = csolver.solve(signal,B);
    end
else
    B = 0;
    C= 0;
end

params = MultiExponentialParameters(Fs,{B,C},'normalized');
signal_recon = params.construct(N);
piece_recon((k-1)*step+1:(k-1)*step+N) = ...
```

```
                piece_recon((k-1)*step+1:(k-1)*step+N) + ...
                real(signal_recon.samples);
end
```

Figure 3 (left): Clusters of λ_i^5 values.

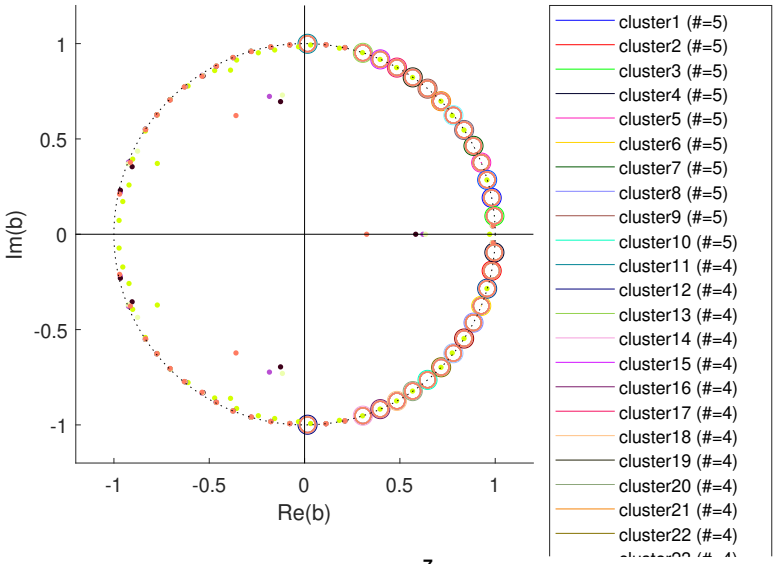

Figure 3 (right): Clusters of λ_i^7 values.

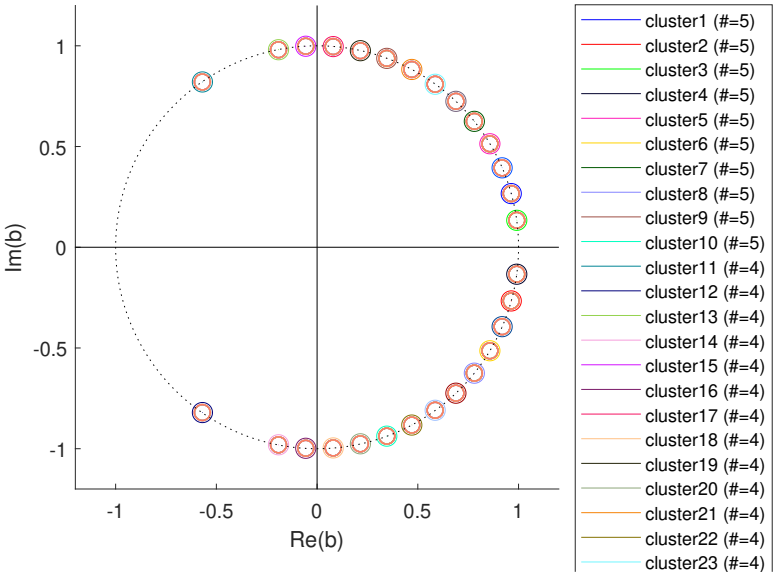

Figure 2 (left): Generalized eigenvalues λ_i from the 60-th window.

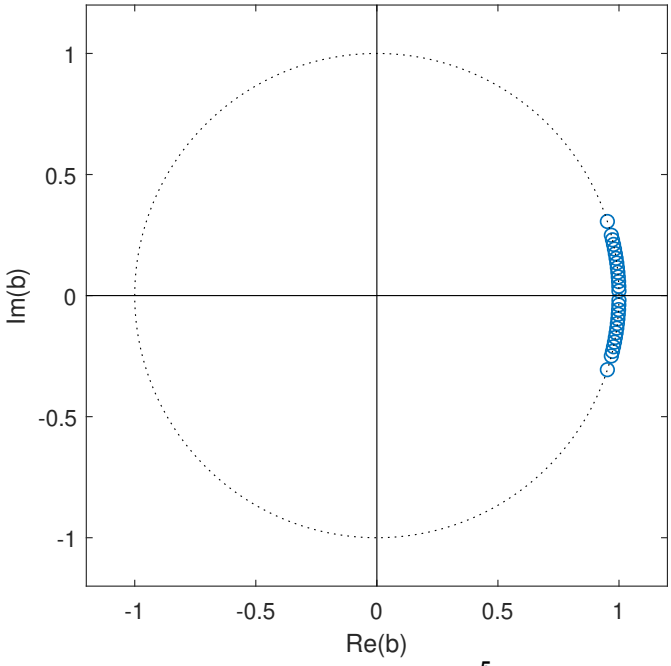

Figure 2 (right): Generalized eigenvalues λ_i^5 from the 60-th window.

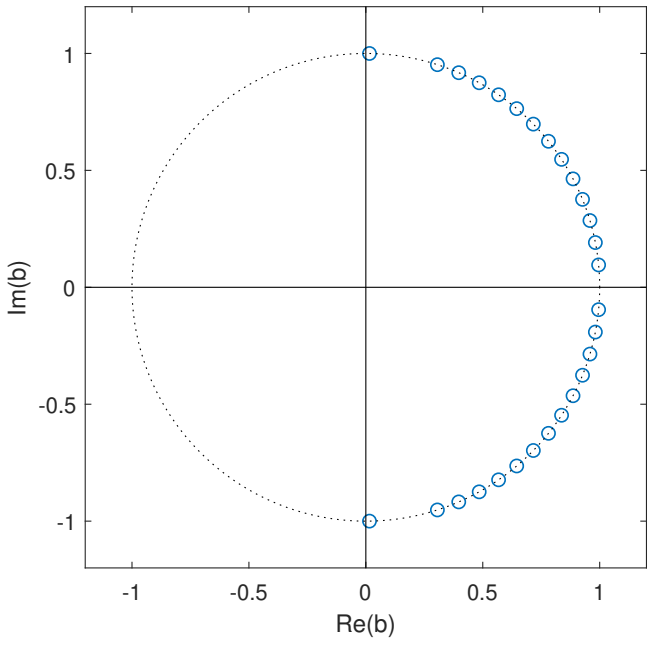

Figure 4 (left): Generalized eigenvalues λ_i from a middle window.

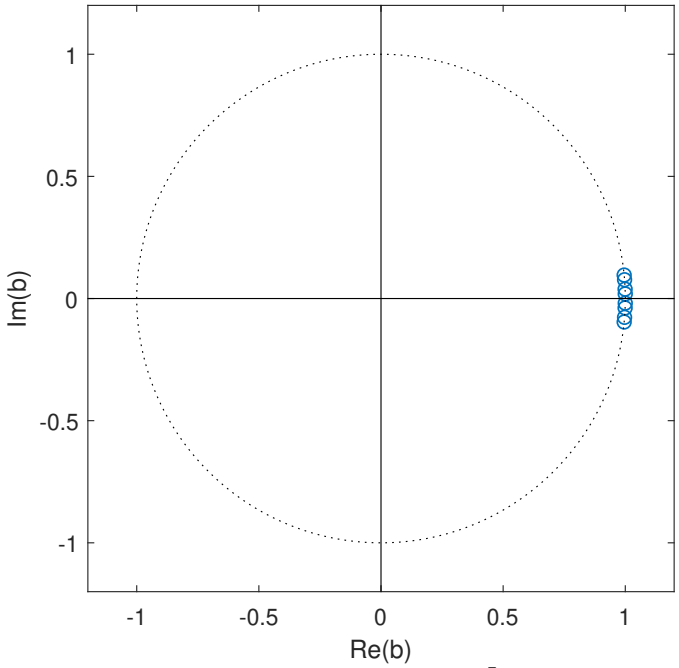

Figure 4 (right): Generalized eigenvalues λ_i^5 from a middle window.