Paper: How to get high resolution results from sparse and coarsely sampled data.

Annie Cuyt, Wen-shin Lee

How to get high resolution results from sparse and coarsely sampled data.

Example 5.2.

Contents

Script environment

This script depends on the random number generator state.

```
clear
close all
```

5.2 Example where collisions occur

In Table 2 we list the α_i and λ_i of an exponential model, chosen in such a way that the aliasing causes terms to collide. This enables us to illustrate the workings of the technique explained in Section 4. Moreover, some of the terms in the $\alpha_i^{(1)}$ will cancel and so there is an additional challenge to retrieve the correct number of terms in $\alpha_i^{(0)}$. The bandwidth is again $\Omega = 1000$ and we take $\Delta = 1/\Omega$ and $r = 100$. We add white Gaussian noise to the samples with SNR=20 dB and start our computations. When subsampling, the 6 terms collide into 3, as indicated in Figure 7 by the singular value decomposition of $H_N^{(0)}$ with $N = 30$, which reveals its numerical rank equal to 3. Actually, terms 1, 2 and 3 collide, as well as terms 5 and 6:

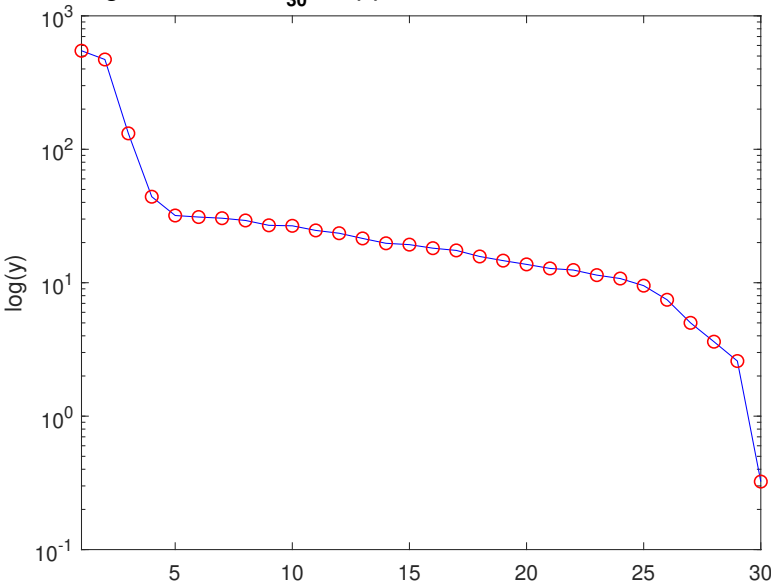
$$\phi(t_{rj}) = (\alpha_1 + \alpha_2 + \alpha_3)\exp(\phi_1 jr\Delta) + \alpha_4 \exp(\phi_4 jr\Delta) + (\alpha_5 + \alpha_6)\exp(\phi_5 jr\Delta)$$

We recall that $H_N^{(0)}$ is filled with the samples f_{jr}, $j = 0, ..., 59$ and not with the samples f_j, $j = 0, ..., 59$.

```
r = 100;
params = params_from_table2;
signal = params.construct(10000);
signal.add_white_gaussian_noise(20,'db');
```

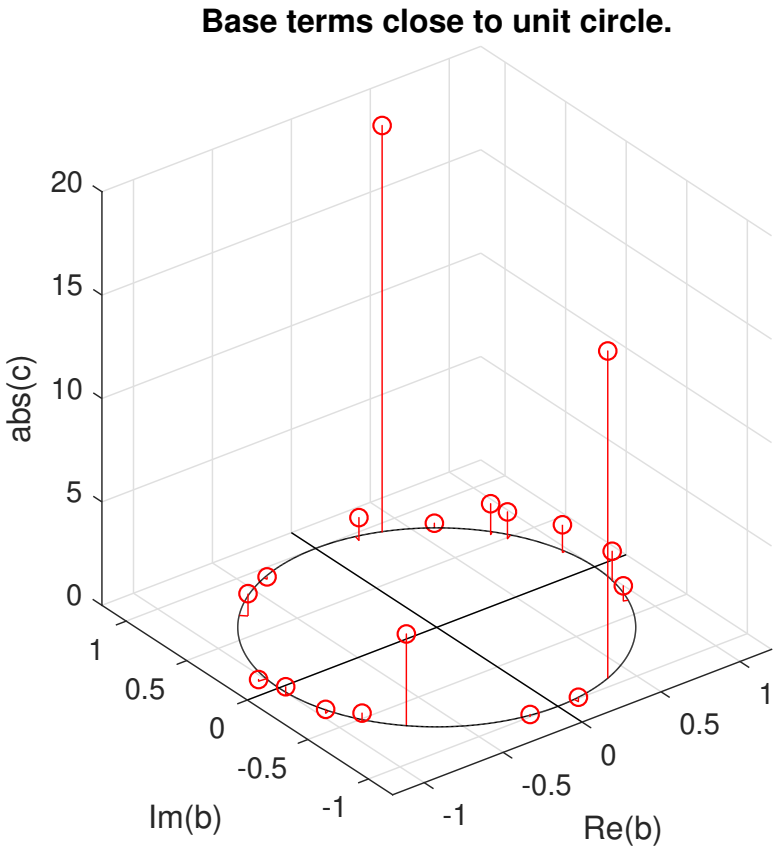
```
signal_r = signal.select(1:r:59*r+1);
plot_signal_SVD(signal_r,'--plot-what','log');
ylim([0.1,1000])
xlim([1,30])
title(['Figure 7: SVD of H_{30}^{(0)} for (1) with data from '...
        'table 2 and r = 100.'])
legend off
```

Figure 7: SVD of $H_{30}^{(0)}$ for (1) with data from table 2 and r = 100.



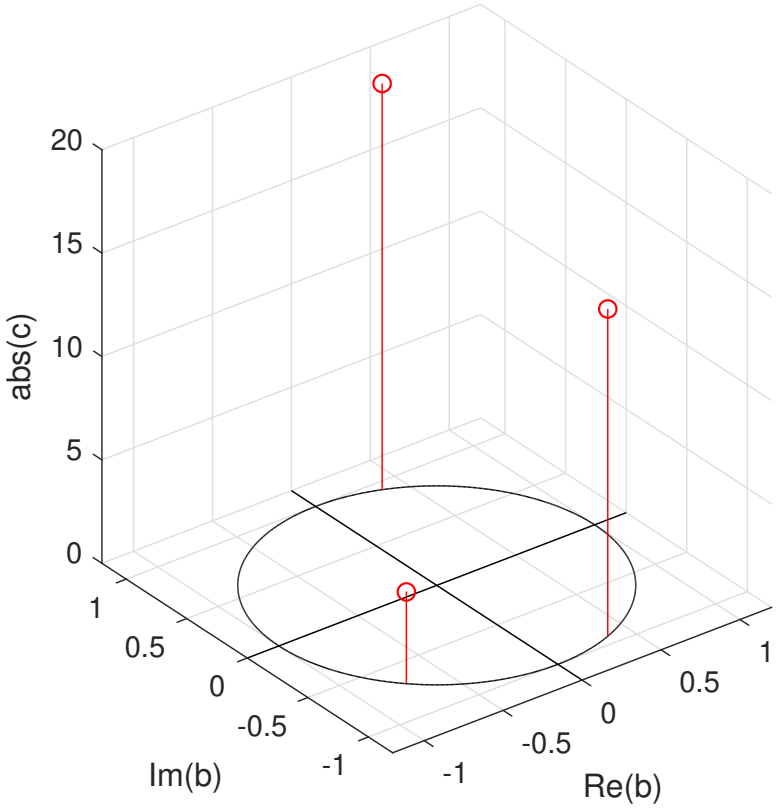
We set up the 30×30 generalized eigenvalue problem (2) with the samples f_{rj} , $j = 0, ..., 59$. After removing the terms too far from the unit circle, three terms stand out:

```
bsolver = BSolverEsprit('--nsamples',60,'--ncols',30,...
                        '--nrows',31,'--nterms',30);
csolver = CSolverVandermondeLS('--nrows',60,'--delta',1);

b = bsolver.solve(signal_r);
b(abs(b)>1.05 | abs(b)<0.95) = [];
c = csolver.solve(signal_r,b);

params_subsampling_r = MultiExponentialParameters(1/r,{b,c},'normalized');

plot3_base_terms(params_subsampling_r);
title('Base terms close to unit circle.')
```



We only keep the three terms with the largest amplitude:

```
a_r = params_subsampling_r.amplitude();
params_subsampling_r.remove(filter_components(a_r,3));
plot3_base_terms(params_subsampling_r);
title('Base terms with largest amplitude.')
```

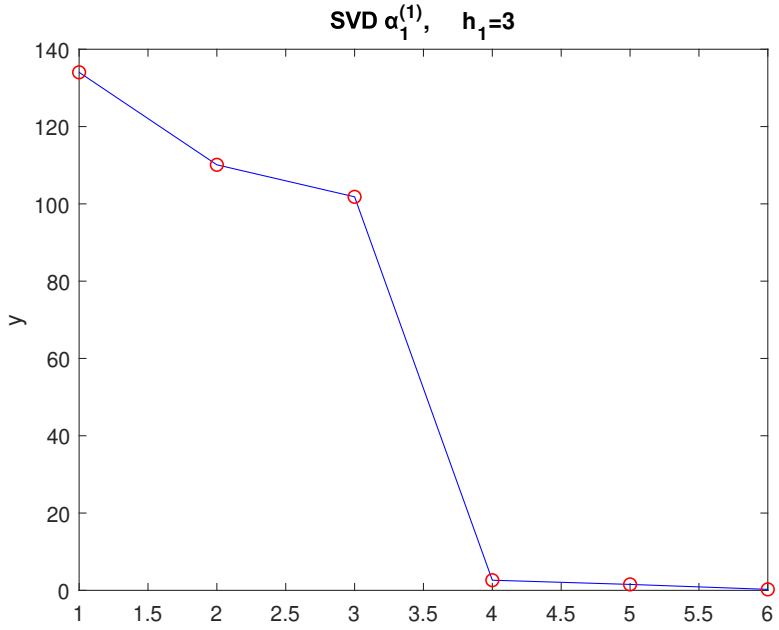


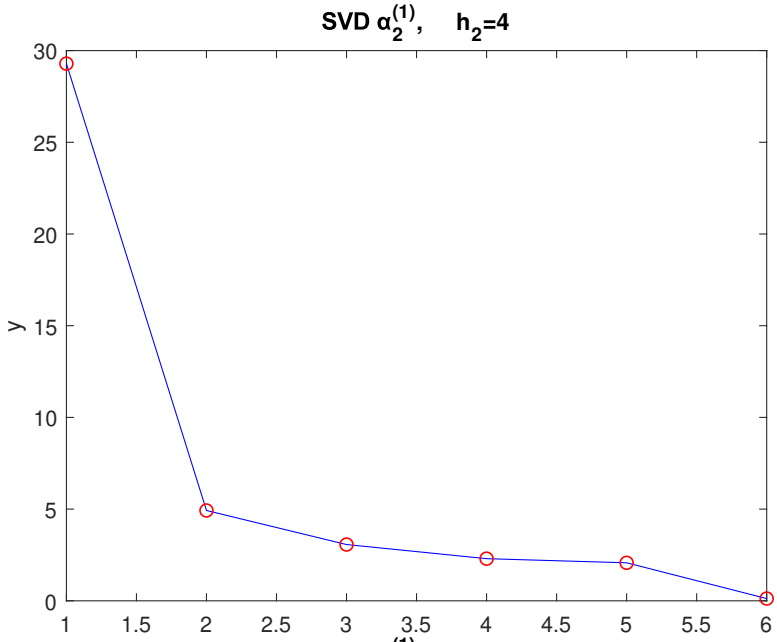
At this point we have not yet been able to recover the correct λ_i and α_i for the signal defined by the parameters in Table 2 (we have unearthed only 3 terms instead of 6) because of two reasons. First, the subsampling creates an aliasing effect and second the aliasing causes frequencies to collide. As explained in Section 4, we can disentangle the information in the collisions from more values $\alpha_i^{(1)}(k)$, $k = 1, 2, ...$, where $\alpha_i^{(0)} = \alpha_i^{(1)}(0)$, simply because the $\alpha_i^{(1)}(k)$ are themselves linear combinations of exponentials. To this end we now choose $\rho = 133$ and we set up the Vandermonde systems (14),

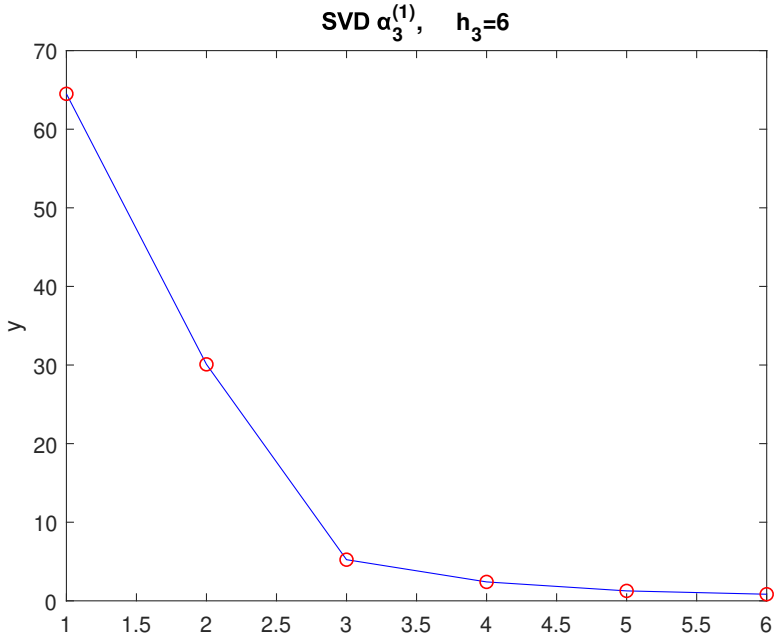
```
rho = 133;

csolver = CSolverVandermondeLS('--nrows',10,'--delta',1);
K = 11;
alpha = zeros(K,3);
for k=1:K
    signal_shifted = signal.select(1+k*rho:r:1+9*r+k*rho );
    alpha(k,:) = csolver.solve(signal_shifted,...
                               params_subsampling_r.b);
end
```

In total so far 170 samples are used. A singular value analysis of the Hankel matrices reveals the number of components that one can distinguish in and consequently extract from the $\alpha_i^{(1)}(k)$. The numerical ranks are 1, 2 and 3 (or a permutation thereof as a result of the randomness of the noise).

```
alpha_signal = cell(1,3);
h = zeros(1,3);
for i=1:3
    alpha_signal{i} = Signal(1,alpha(:,i));
    h(i) = sum(svd(alpha_signal{i}.hankel(6))>20);
    plot_signal_SVD(alpha_signal{i},'--n',6,'--plot-what','lin');
    title(sprintf(['SVD ',char(945),'_%d^{(1)},     h_%d=%d'],...
                  i,i,sum(h)));
    legend off
end
```





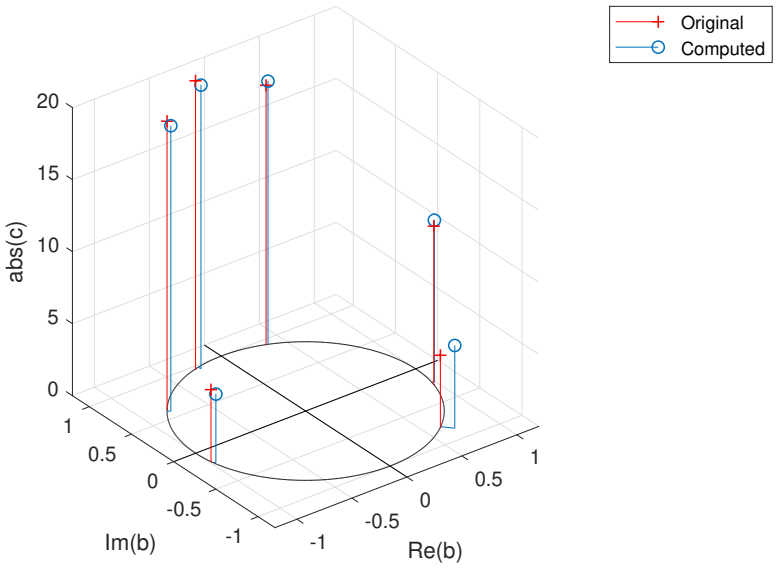


For $i = 1, 2, 3$ the generalized eigenvalue problems reveal the $\lambda_\ell^{(1)} = \exp(\phi\ell\rho\Delta)$, $\ell = h_i, ..., h_{i+1} - 1$. The respective Vandermonde systems with unknowns $\alpha_{h_i}, ..., \alpha_{h_{i+1}-1}$ and right hand sides $\alpha_i^{(1)}(0), ..., \alpha_i^{(1)}(11)$ reveal the α_l , $l = h_i, ..., h_{i+1} - 1$ in (13). Again we retain only the $h_{i+1} - h_i$ components with largest $|\alpha_l|$.

```
params_shifted = cell(1,3);
for i = 1:3
    bcsolver = MultiExponentialSolver...
            (BSolverEsprit('--nsamples',11,'--ncols',5,...
                           '--nrows',7,'--nterms',h(i)),...
            CSolverVandermondeLS('--nrows',11,'--delta',1),...
            '--nsamples',11);
    params_shifted{i} = bcsolver.solve(alpha_signal{i});
end
```

From the $\lambda_\ell^{(0)} = \exp(\phi_\ell r\Delta)$, $l = h_i, \ldots, h_{i+1} - 1$, $i = 1, \ldots, n_0$ and $\lambda_\ell^{(1)} = \exp(\phi_\ell \rho\Delta)$, $l = 1, \ldots, n$ the imaginary part of ϕ_i can be recovered as indicated in Lemma 2: with $p_1 = 4$ and $p_2 = -3$ we have $p_1 r + p_2 \rho = 1$.

```
p1 =  4;
p2 = -3;

alpha = [];
b_r = [];
b_rho = [];
for i = 1:3
    alpha = [alpha,params_shifted{i}.c];
    b_r = [b_r,repmat(params_subsampling_r.b(i),1,h(i))];
    b_rho = [b_rho,params_shifted{i}.b];
end

b = b_r.^p1 .* b_rho.^p2;

params_final = MultiExponentialParameters(...
                        signal.sampling_frequency,...
                        {b,alpha}, 'normalized');
plot3_base_terms(params,params_final,...
                    '--legend',{'Original','Computed'});
```


Original
Computed
20
15
10
5
0
abs(c)
1
0.5
0
-0.5
-1
Im(b)
-1
-0.5
0
0.5
1
Re(b)