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 5.3.

Contents

Script environment

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

```
clear
close all
```

5.3 Example where cancellation occurs

The cancellation strategy is most clearly illustrated on a noisefree example, where exact cancellation happens. The actual occurrence of this situation in case of real-life data is extremely small, but we primarily want to show that the proposed sub-Nyquist method is capable of recovering from it.

Let

$\phi(t) = \exp(2\pi it) - \exp(2\pi i21t) + \exp(2\pi i41t) - \exp(2\pi i61t) + e^{i2\pi 72/100}\exp(2\pi i11t) - e^{(i2\pi 32/100)}\exp(2\pi i31t) + \exp(2\pi i9t)$.

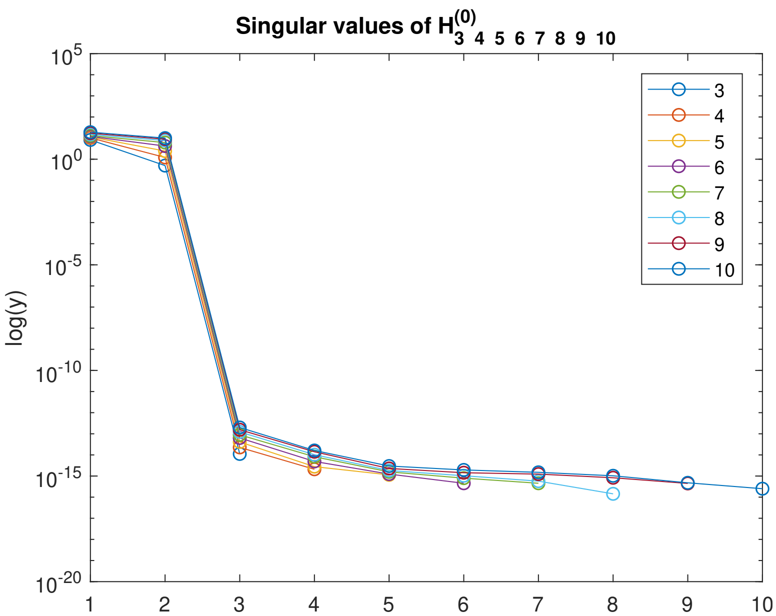
We take $\Omega = 100$, $\Delta = 0.01$ and sample $f_j = \phi(j\Delta)$ for particular values of j. With $r = 5$ the first four terms cancel each other and the fifth and sixth term collide:

$f_{5j} = 0\exp(2\pi i5j/100) + \left(e^{i2\pi 72/100} - e^{(i2\pi 32/100)}\right)\exp(2\pi i55j/100) + \exp(2\pi i45j/100)$.

```
params = params_from_example_5_3;
nsamples = 1000;
signal = params.construct(nsamples);
```

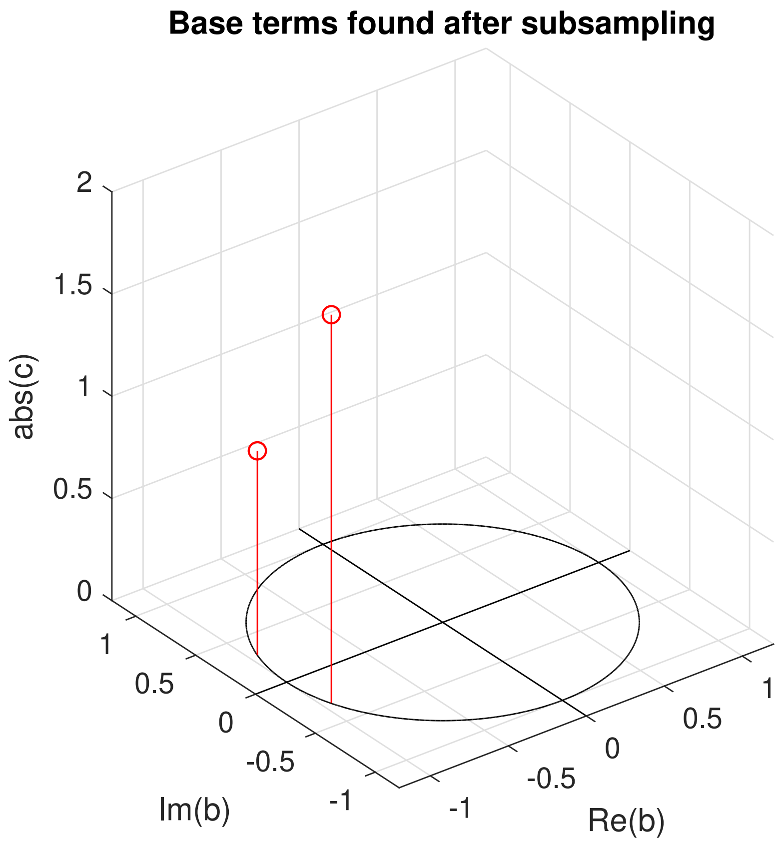
So from the samples $f_0, f_5, f_{10}, ...$ only two terms can be retrieved:

```
r = 5;
signal_r = signal.select(1:r:nsamples,...
                              '--label','subsampled signal');
plot_signal_SVD(signal_r,'--n',3:10,'--plot-what','log');
title('Singular values of H^{(0)}_{3  4  5  6  7  8  9  10}')
```



The 2 eigenvalues that we can already compute, are $\lambda_5^{(0)} = \exp(2\pi i 55/100)$ and $\lambda_7^{(0)} = \exp(2\pi i 45/100)$. From the Vandermonde system we find $\alpha_5^{(0)} = e^{i2\pi 72/100} - e^{i2\pi 32/100}$ and $\alpha_7^{(0)} = 1$.

```
bcsolver = MultiExponentialSolver...
            (BSolverEsprit('--nsamples',4,'--ncols',2,...
                           '--nrows',3,'--nterms',2),...
            CSolverVandermondeLS('--nrows',2,'--delta',1),...
            '--nsamples',4);

params_r = bcsolver.solve(signal_r);
plot3_base_terms(params_r);
title('Base terms found after subsampling')
```

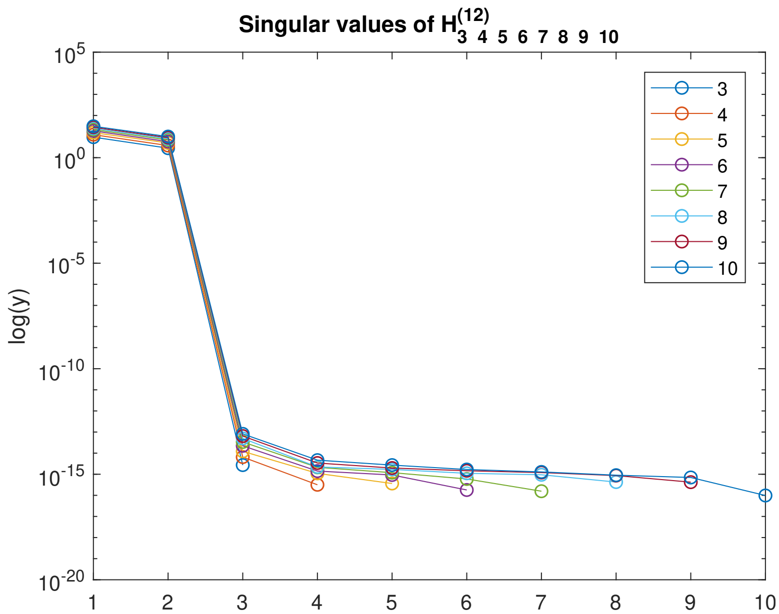


We now need to ask ourselves whether n_0 truly equals 2 or whether some cancellation of terms has happened. With $\rho = 12$ we find that

$$f_{5j+12k} = \left(e^{i2\pi 12k/100} - e^{i2\pi 52k/100} + e^{i2\pi 92k/100} - e^{i2\pi 32k/100}\right)\exp(2\pi i 5j/100) + \left(e^{i2\pi 72/100}e^{i2\pi 32k/100} - e^{i2\pi 32/100}e^{i2\pi 72k/100}\right)\exp(2\pi i 55j/100) + e^{(i2\pi 8k/100)}\exp(2\pi i 45j/100).$$

For $k = 1$ we hit an accidental zero for the coefficient of $\exp(2\pi i 55j/100)$ and therefore $\mathrm{rank} H_N^{(12)} = 2$, $N \geq 2$.

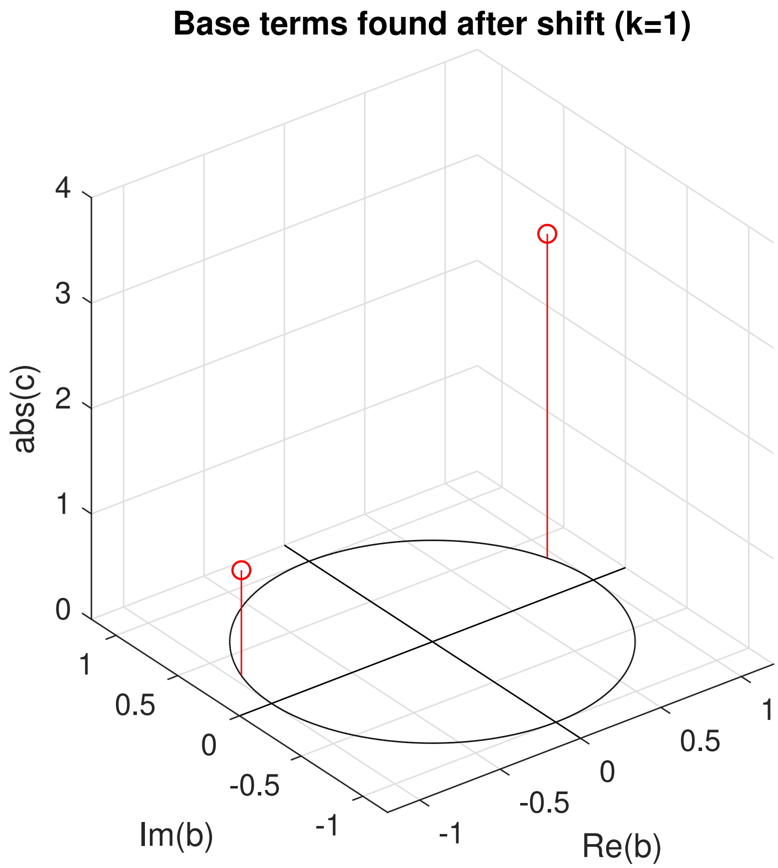
```
rho = 12;
signal_shifted1 = signal.select(1+rho:r:nsamples,...
                              '--label','shifted signal (k=1)');
plot_signal_SVD(signal_shifted1,'--n',3:10,'--plot-what','log');
title('Singular values of H^{(12)}_{3  4  5  6  7  8  9  10}')
```



The generalized eigenvalues, namely $\lambda_1^{(0)} = \exp(2\pi i5/100)$ and $\lambda_7^{(0)} = \exp(2\pi i45/100)$, also belong to the n_0 eigenvalues that are identifiable from the evaluations at the multiples of $r\Delta$, since a shift does not change the generalized eigenvalues, only their coefficients. From the Vandermonde system we find $\alpha_1^{(1)}(1)$ and $\alpha_7^{(1)}(1)$.

Apparently n_0 equals at least 3, because in the first bunch computed from the samples f_{5j+12k} with $k = 0$ we find two eigenvalues and in the second bunch with $k = 1$ we find one more.

```
bcsolver = MultiExponentialSolver...
            (BSolverEsprit('--nsamples',4,'--ncols',2,...
                           '--nrows',3,'--nterms',2),...
            CSolverVandermondeLS('--nrows',2,'--delta',1),...
            '--nsamples',4);

params_shifted1 = bcsolver.solve(signal_shifted1);
plot3_base_terms(params_shifted1);
title('Base terms found after shift (k=1)')
```

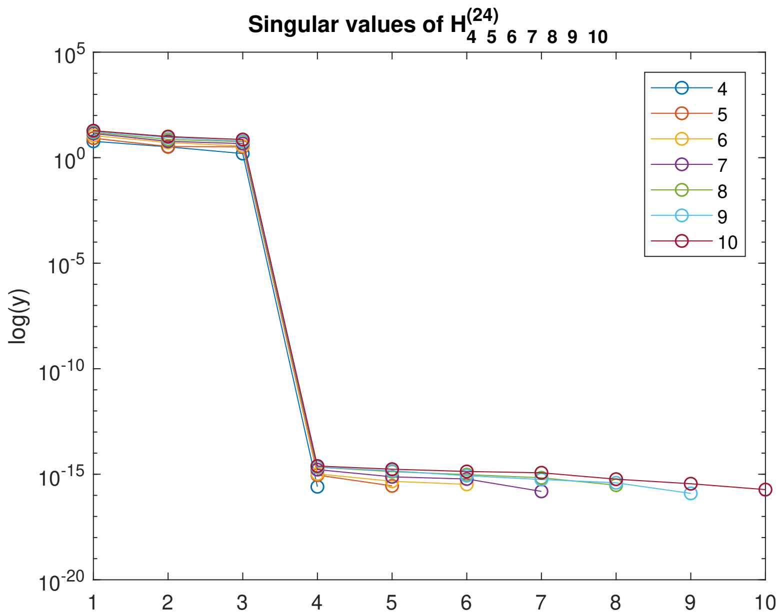


Let us turn our attention to larger values of k to have the current estimate of $n_0 = 3$ confirmed and to extract all n distinct terms. As described in Section 4 on the disentangling of collisions, we continue sampling at multiples of the shift, namely we collect the $f_{jr+k\rho}$ for $k > 1$. With $k = 2$ we obtain

$f_{5j+24} = \left(e^{i2\pi 24/100} - e^{i2\pi 4/100} + e^{i2\pi 84/100} - e^{i2\pi 64/100}\right)\exp(2\pi i 5j/100) + \left(e^{i2\pi 36/100} - e^{i2\pi 76/100}\right)\exp(2\pi i 55j/100) e^{(i2\pi 16/100)}\exp(2\pi i 45j/100)$

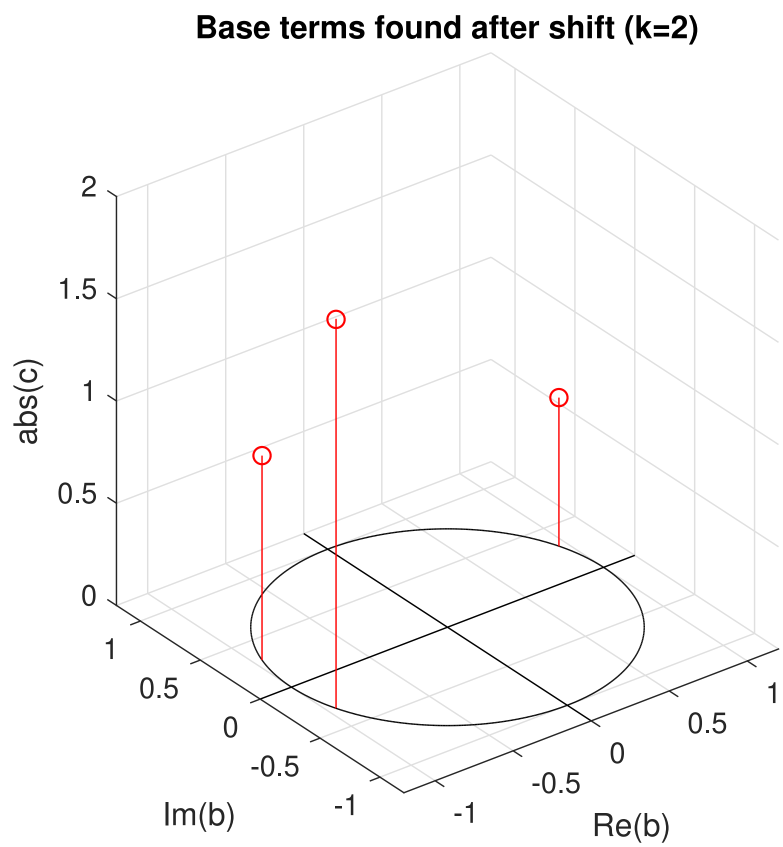
and $\mathrm{rank} H_N^{(24)} = 3$, $N \geq 4$.

```
signal_shifted2 = signal.select(1+2*rho:r:nsamples,...
                               '--label','shifted signal (k=1)');
plot_signal_SVD(signal_shifted2,'--n',4:10,'--plot-what','log');
title('Singular values of H^{(24)}_{4  5  6  7  8  9  10}')
```



Merely for completeness we compute the generalized eigenvalues. We find $\lambda_1^{(0)} = \exp(2\pi i 5/100)$, $\lambda_5^{(0)} = \exp(2\pi i 55/100)$ and $\lambda_7^{(0)} = \exp(2\pi i 45/100)$, which confirms our earlier obtained result. Hence $n_0 = 3$. We also compute the values $\alpha_i^{(1)}(2)$, $i = 1, 5, 7$ from the Vandermonde system.

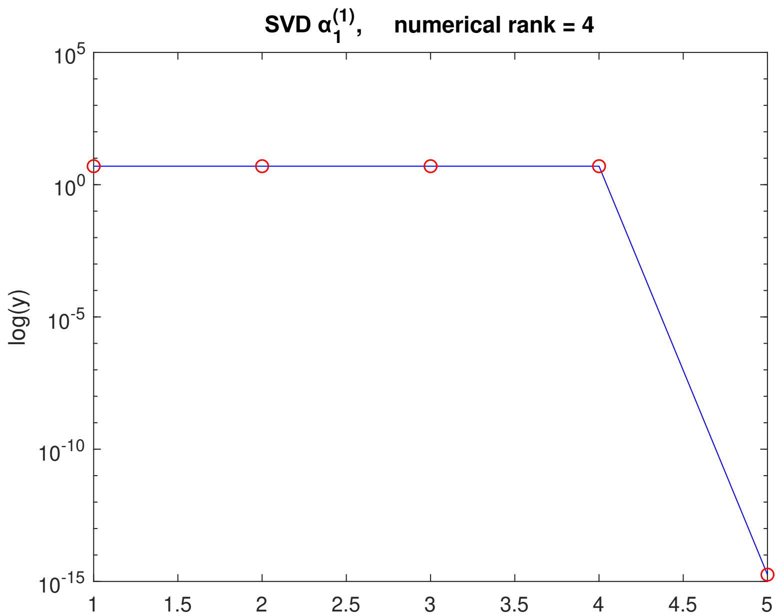
```
bcsolver = MultiExponentialSolver...
            (BSolverEsprit('--nsamples',6,'--ncols',3,...
                           '--nrows',4,'--nterms',3),...
            CSolverVandermondeLS('--nrows',3,'--delta',1),...
            '--nsamples',6);

params_shifted2 = bcsolver.solve(signal_shifted2);
plot3_base_terms(params_shifted2);
title('Base terms found after shift (k=2)')
```

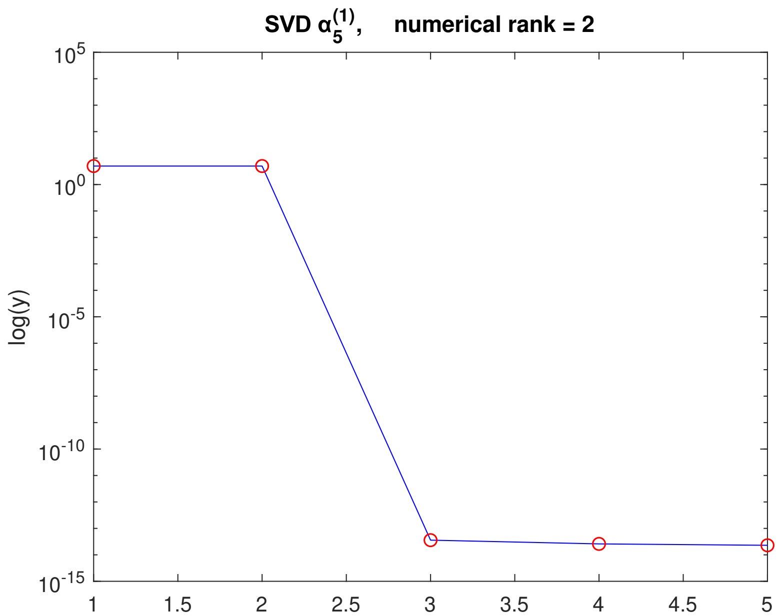


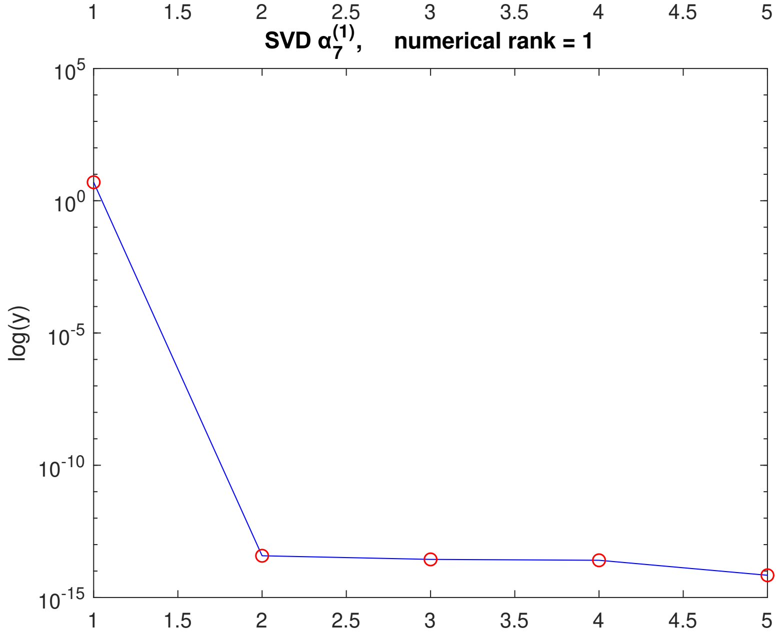
The purpose now is to find out how many terms are in the expression $\alpha_i^{(1)}(k)$ for each i retrieved so far. We compute $\alpha_1^{(1)}(k), \alpha_5^{(1)}(k), \alpha_7^{(1)}(k)$ for $k \geq 3$ from the Vandermonde system. When pursuing shifts up to $k = 9$, we find that for $i = 1$ the rank is 4, for $i = 5$ the rank is 2 and for $i = 7$ the rank is 1, leading to a grand total of $n = 7$ distinct terms.

```
csolver = CSolverVandermondeLS('--nrows',3,'--delta',1);

K = 9;
alpha_i_1_k = zeros(3,K+1);
for k = 0:K
    signal_shiftedk = signal.select(1+k*rho:r:1+k*rho+2*r);
    alpha_i_1_k(:,k+1) = csolver.solve(...
                                signal_shiftedk,params_shifted2.b);
end

alpha_signal = cell(1,3);
I = [1,5,7];
h = zeros(1,3);
```

```
for i = 1:3
    alpha_signal{i} = Signal(1,alpha_i_1_k(i,:));
    plot_signal_SVD(alpha_signal{i},'--n',5,'--plot-what','log');
    h(i) = sum(svd(alpha_signal{i}.hankel(0)) > 1e-10);
    title(sprintf(['SVD ',char(945),'_%d^{(1)},     numerical rank = %d'],...
                  I(i),h(i)));
    legend off;
end
```



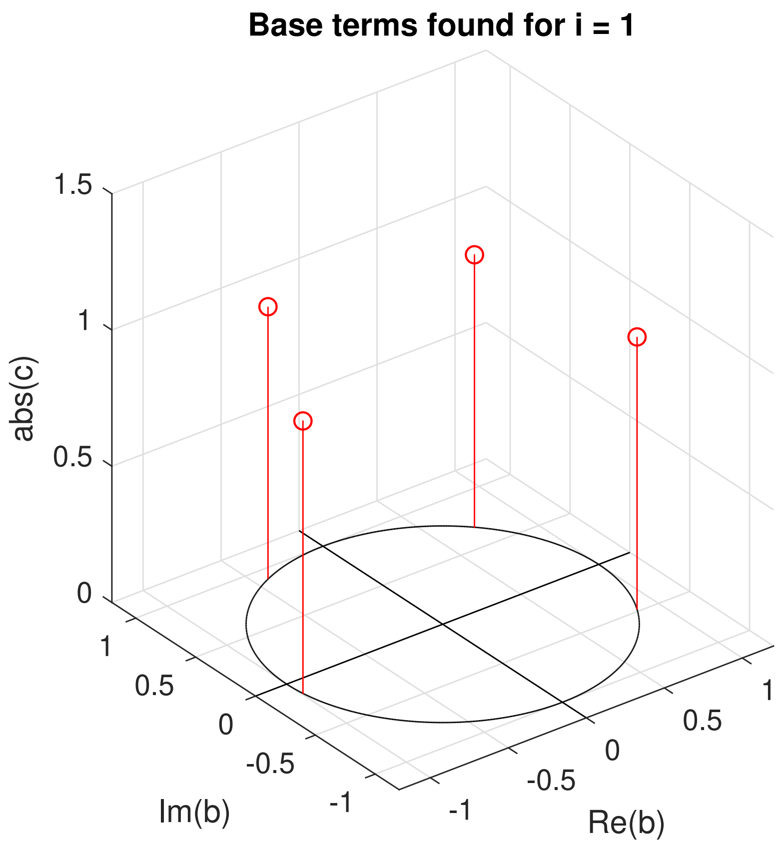


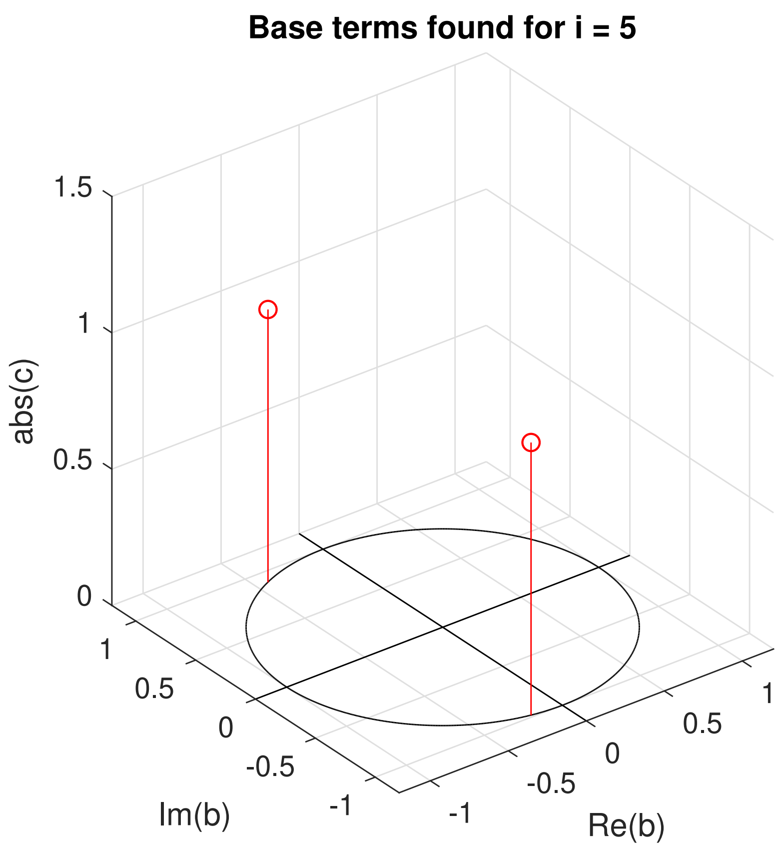


We now separate the terms that are hiding in each of the collisions by computing the generalized eigenvalues. We find $\lambda_1^{(1)} = \exp(2\pi i 12/100)$, $\lambda_2^{(1)} = \exp(2\pi i 52/100)$, $\lambda_3^{(1)} = \exp(2\pi i 92/100)$, , $\lambda_4^{(1)} = \exp(2\pi i 32/100)$, $\lambda_5^{(1)} = \exp(2\pi i 32/100)$, $\lambda_6^{(1)} = \exp(2\pi i 72/100)$, $\lambda_7^{(1)} = \exp(2\pi i 8/100)$.

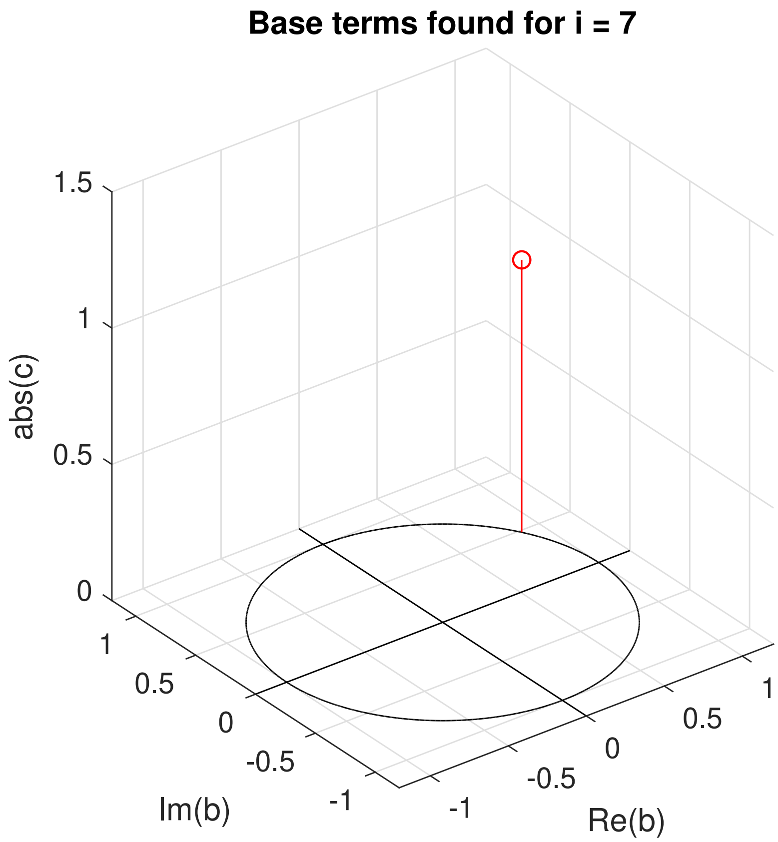
```
params_shifted_final = cell(1,3);
for i = 1:3
```

```
    bcsolver = MultiExponentialSolver...
            (BSolverEsprit('--nsamples',2*h(i),'--ncols',h(i),...
                           '--nrows',h(i)+1,'--nterms',h(i)),...
            CSolverVandermondeLS('--nrows',h(i),'--delta',1),...
            '--nsamples',2*h(i));
    params_shifted_final{i} = bcsolver.solve(alpha_signal{i});
    plot3_base_terms(params_shifted_final{i});
    title(sprintf('Base terms found for i = %d',I(i)))

end
```





Base terms found for i = 5
abs(c)
1.5
1
0.5
0
Im(b)
1
0.5
0
-0.5
-1
Re(b)
-1
-0.5
0
0.5
1



At this stage we have all the information to reconstruct the non-aliased generalized eigenvalues

```
p1 = 5;
p2 = -2;

alpha = [];
b_r = [];
b_rho = [];
for i = 1:3
    alpha = [alpha,params_shifted_final{i}.c];
    b_r = [b_r,repmat(params_shifted2.b(i),1,h(i))];
    b_rho = [b_rho,params_shifted_final{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'});
```