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 5.1.

Contents

Script environment

This script depends on the random number generator state.

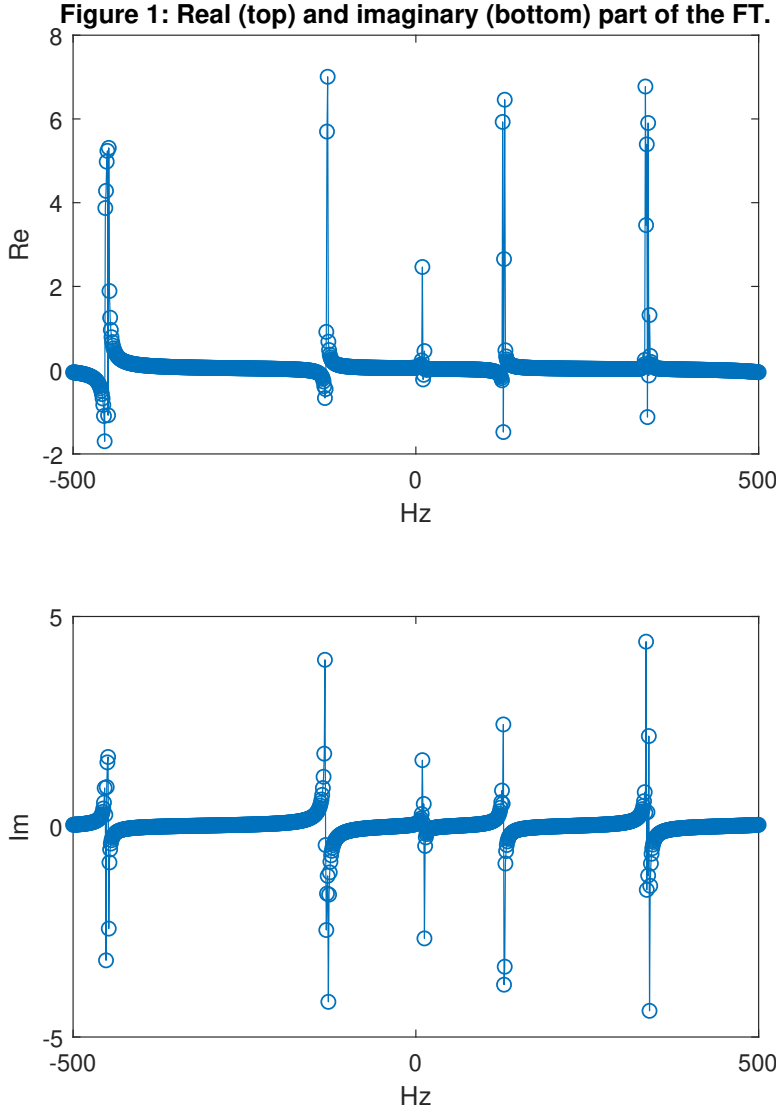
```
clear
close all
```

5.1 Collision-free example

For our first example the α_i and ϕ_i are given in Table 1. We take $\Omega = 1000$ and $\Delta = 1/\Omega$. The $n = 20$ frequencies ω_i form 5 clusters, as is apparent from the FT, computed from 1000 samples and shown in Figure 1. For completeness we graph the signal in Figure 2. In Figure 3 we show the generalized eigenvalues $\lambda_i = \exp(\phi_i \Delta)$, $i = 1, ..., 20$ computed from the noisefree samples, to illustrate the ill-conditioning of the problem as a result of the clustering of the frequencies.

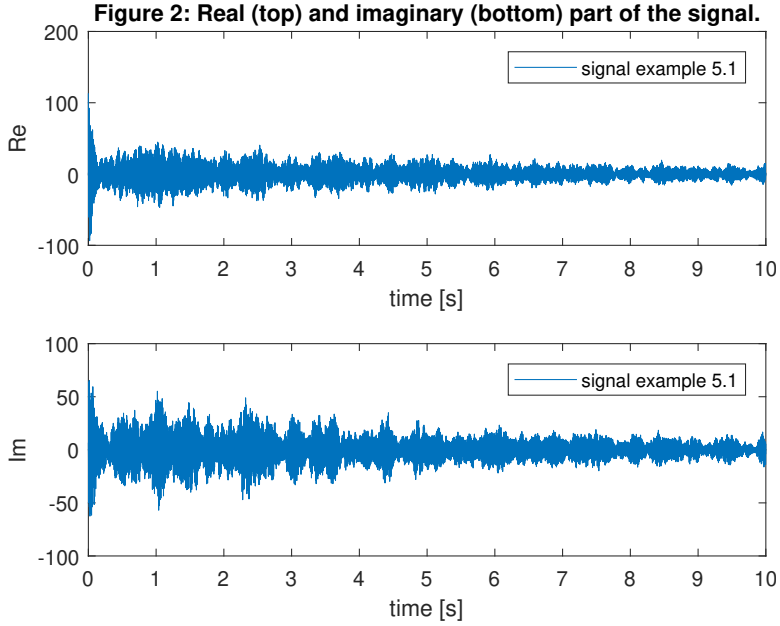
```
params = params_from_table1;
signal = params.construct( 10000, 'signal example 5.1');

plot_signal_FFT(signal,'--nsamples',1000,...
                    '--plot-symbol','-o','RI');
subplot(2,1,1);
title(['Figure 1: Real (top) and imaginary (bottom) part of '...
        'the FT.']);
subplot(2,1,2); title('');
```

Figure 1: Real (top) and imaginary (bottom) part of the FT.



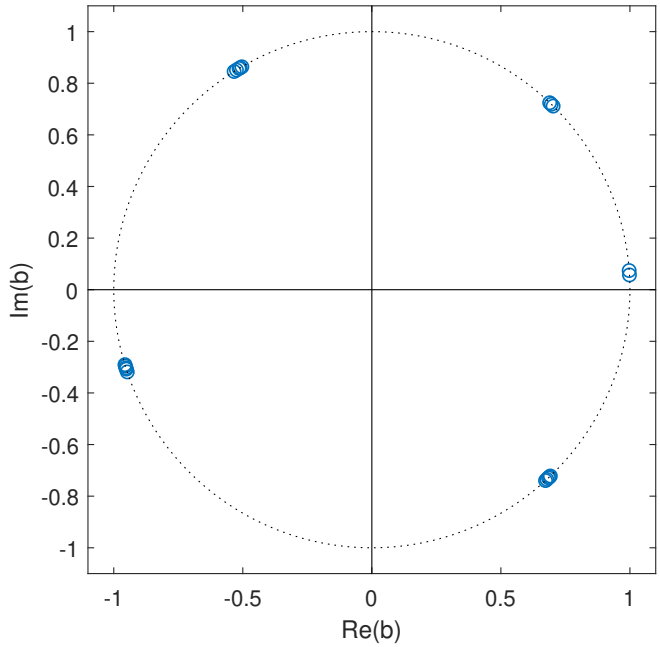

```
plot_signal(signal,'--plot-what','RIs');
subplot(2,1,1)
title(['Figure 2: Real (top) and imaginary (bottom) part of '...
       'the signal.']);
subplot(2,1,2); title('');
```

Figure 2: Real (top) and imaginary (bottom) part of the signal.




```
plot_base_terms( params.b, '--width', 1.1 );
title(['Figure 3: Generalized eigenvalues \lambda_i for (1) ',...
       'with data from Table 1.']);
```

Figure 3: Generalized eigenvalues λ_i for (1) with data from Table 1.



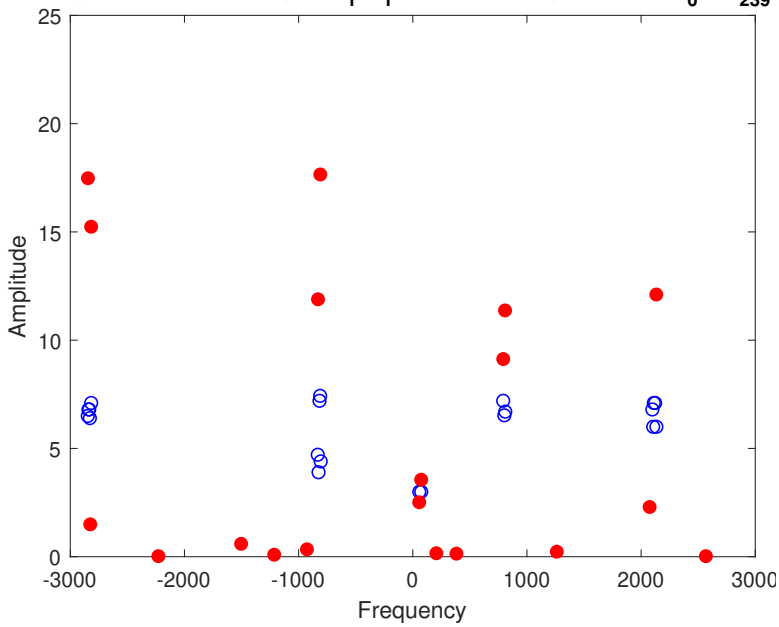
To (1) we add white Gaussian noise with SNR=32 dB.

```
signal.add_white_gaussian_noise(32,'db');
```

For comparison with our new method, we show in Figure 4 the (ω_i, β_i) results computed by means of ESPRIT using 240 samples, namely $f_0, ..., f_{239}$ for a signal space of dimension 20 and a noise space of dimension 40 (so total dimension N = 60 for which the best ESPRIT results were obtained).

```
bcsolver = MultiExponentialSolver...
            (BSolverEsprit('--nsamples',240,'--ncols',60,...
                           '--nrows',181,'--nterms',20),...
            CSolverVandermondeLS('--nrows',240),...
            '--nsamples',240);
params_esprit = bcsolver.solve(signal);

figure;
plot(params.frequency(), params.amplitude(), 'bo'); hold on;
plot(params_esprit.frequency(), params_esprit.amplitude(),...
        'r.' ,'MarkerSize', 20 );
title(['Figure 4: ESPRIT output (\omega_i, \beta_i),'...
        'i=1,...,20 computed from f_0,...,f_{239}'])
xlabel('Frequency'); ylabel('Amplitude')
ylim([0,25])
```

Figure 4: ESPRIT output (ω_i, β_i),i=1,...,20 computed from f_0,...,f_{239}

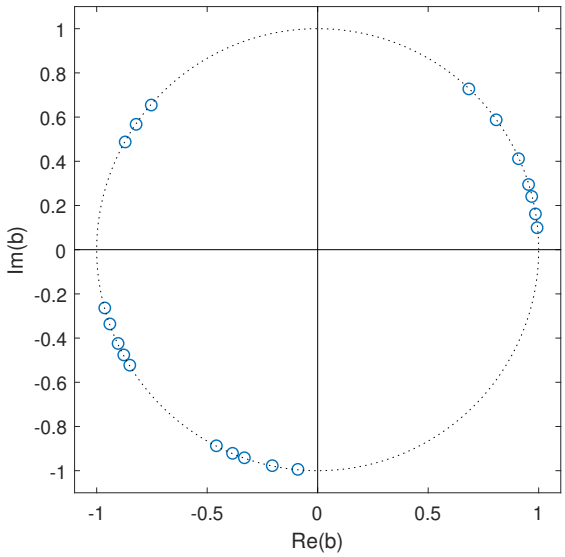


Next we choose $r = 11$ and $\rho = 5$. The originally clustered eigenvalues are now much better separated. To illustrate this we show in Figure 5 the noisefree

generalized eigenvalues λ_i^r, $i = 1, ..., 20$ of the r-fold undersampled exponential analysis problem.

```
r   = 11;
rho =  5;
plot_base_terms( params.b.^r, '--width', 1.1 );
title(['Figure 5: Generalized eigenvalues \lambda_i^r for (1) '...
           'with data from Table 1 and r = 11.']);
```

Figure 5: Generalized eigenvalues λ_i^r for (1) with data from Table 1 and r = 11.



With the noisy samples, we now take $N = 60 > n = 20$ and set up the 120×60 generalized eigenvalue problem (2) with the samples f_{jr}, $j = 0, \ldots, 179$ and the 120×60 Vandermonde system (9) that respectively deliver the λ_i^r and the α_i for $i = 1, \ldots, N$. With the samples $f_{jr+\rho}, j = 0, \ldots, 59$ we set up the 60×60 linear system (10) from which we compute the $\alpha_i \lambda_i^\rho, i = 1, \ldots, 60$ and subsequently the λ_i^ρ. This brings our total number of samples used also to 240, comparable to the ESPRIT procedure.

```
signal_r = signal.select(1:r:179*r+1,...
                         '--label','example1_signal_r');
signal_rho = signal.select((1:r:59*r+1)+rho,...
                         '--label','example1_signal_rho');

bcsolver = MultiExponentialSolver...
            (BSolverEsprit('--nsamples',180,'--ncols',60,...
                           '--nrows',121,'--nterms',60),...
            CSolverVandermondeLS('--nrows',120,'--delta',1),...
            '--nsamples',180);
```

```
csolver2 = CSolverVandermondeLS('--nrows',60,'--delta',1);

params_subsampling_r = bcsolver.solve(signal_r);
coeff_rho = csolver2.solve(signal_rho, params_subsampling_r.b);
lambda_rho = coeff_rho./params_subsampling_r.c;
```

Using the Euclidean algorithm, as explicited in Lemma 2, we recover from λ_i^r and λ_i^ρ the true frequencies ω_i with $p_1 = 1, p2 = -2$ and $p_1 r + p2\rho = 1$.

```
p1 =  1;
p2 = -2;
assert( p1*r+p2*rho == 1 );
lambda = (params_subsampling_r.b.^p1).*(lambda_rho.^p2);

params_subsampling = MultiExponentialParameters ...
                ( 1000     ...
                , {lambda,params_subsampling_r.c},'normalized'...
                );
```

With the new method we find the (ω_i, β_i) couples shown as blue dots in Figure 6. In Figure 6 the reader can clearly count the number of frequencies retrieved in each cluster, which is the correct number when comparing to the input values in Table 1. Clearly Figure 6 is a tremendous improvement over Figure 4.

```
figure;
plot( params.frequency(), params.amplitude(), 'ko'); hold on;
plot( params_subsampling.frequency(), ...
        params_subsampling.amplitude(), 'b.', 'MarkerSize', 20);
ylim([2,8])
xlabel('Frequency')
ylabel('Amplitude')
title(['Figure 6: Output (\omega_i,\beta_i), i=1,...,20 '...
        'computed from 240 samples f_{jr+\rho}, r=11, \rho=5.'])
```

Figure 6: Output (ω_i,β_i), i=1,...,20 computed from 240 samples $f_{jr+\rho}$, r=11, ρ=5.