



UNIVERSITÉ
TOULOUSE III
PAUL SABATIER



M2TSI - SPACEMASTER

UE31 LABORATORY REPORT

Spectral Analysis and Sparse Representation

Authors:

Arthur Scharf
Andreas Wenzel

January 23, 2017

1 Introduction

In this report the spectral analysis of irregularly sampled data - in particular line spectra - is performed using the Fourier Transform. Also, the sparse representation of signals is evaluated by applying different classical methods such as "greedy" algorithms and "convex relation" approaches. These methods are applied in particular to irregular sampled data, since this is the most realistic representation of acquired data in astronomy.

2 Spectral analysis with Fourier Transform

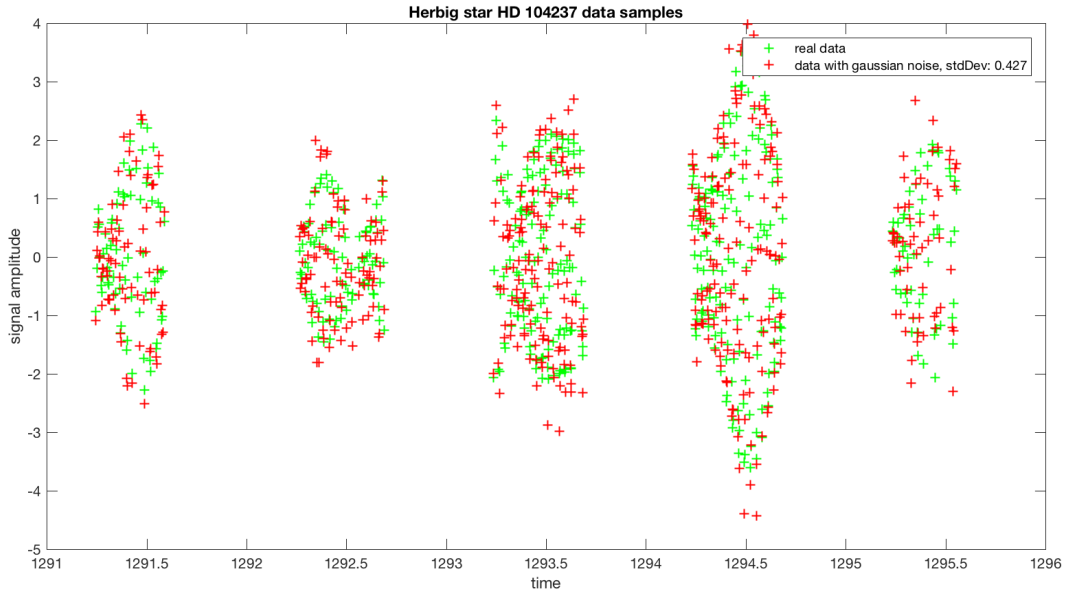


Figure 1: Original data samples and samples with additional noise in time domain

Since working on a real data set might be difficult, in particular when it comes to understanding the underlying techniques of the methods aforementioned, we will simulate our irregular sampled data, for which an initial dataset was given (c.f appendix A). Using this amplitude, phase, time, frequency and the appropriate radial velocity data, the following formula was used to create a realistic data set that is basically a noisy sum of sine functions

$$x(t_n) = \sum_{k=1}^K A_k \sin(2\pi\nu_k t_n + \phi_k) + \epsilon_n \quad (1)$$

As for the Signal-to-Noise ratio 20dB in power mean were used, and for the number of sine functions $K = 5$, as our initial data set contained 5 different amplitudes and its according phases

and frequencies (c.f. appendix A). Figure 1 shows the generated data, where the green part corresponds to our simulated data without noise (hereafter seen as "real" or "original" data) and the red part including noise. The noisy data set will be used from here on in all upcoming calculations.

2.1 Irregular sampling case

In the case of irregular sampled data we can not apply the Fast-Fourier-Transform (FFT) as we would have in the regular case. However, the Fourier-Transform can be computed by introducing a Matrix \mathbf{W} , such as

$$\mathbf{W}(l, c) = \exp(2j\pi t_l f_c) \quad , \quad \hat{\mathbf{x}} = \mathbf{W}^\dagger \mathbf{x} \quad (2)$$

with $\hat{\mathbf{x}}$ being the Fourier-Transform of vector \mathbf{x} . As stated above, the FFT can not be used here, since the Matrix \mathbf{W} must be orthogonal for the FFT - which it is not in the case of irregular sampled data.

Nevertheless, having this tool handy, we can now perform the Fourier Transform on irregular sampled data, e.g. on our data set represented in fig. 1.

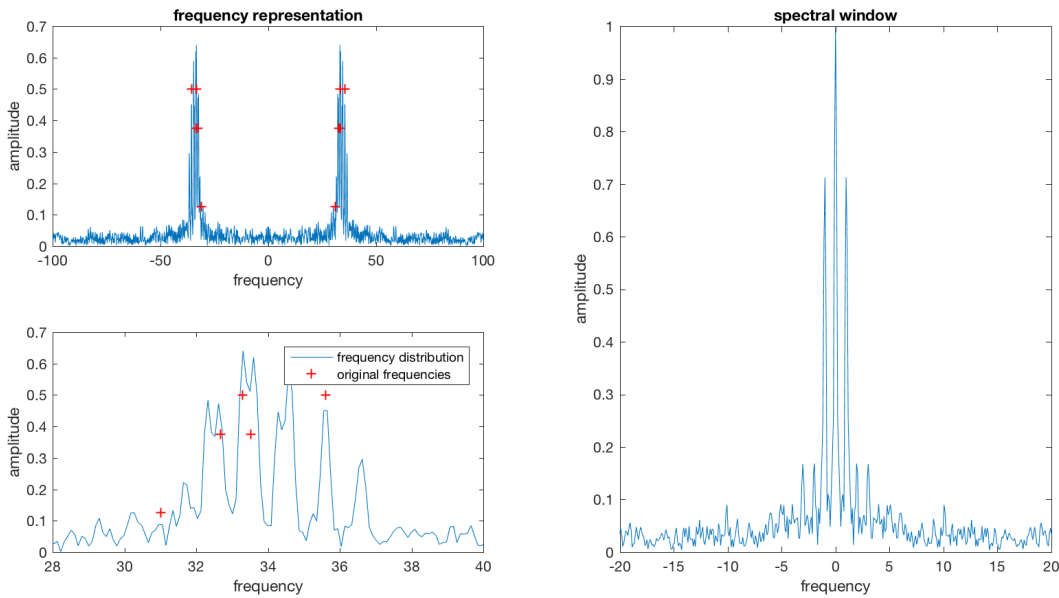


Figure 2: Frequency representation with original frequencies and spectral window

For the beginning we only use one sine-function (in eq. (1), thus $K=1$). In this case it is very easy to determine the frequency and amplitude of the underlying sine-signal in the frequency representation of data, even if there are quite high side lobes.

However, if a noisy sum of 5 sine signals is used, the underlying frequencies and amplitudes

can
we
also
read
out
the
phase
from

can not be easily determined anymore. This is due to the fact that the sum of signals and their noises add up, so that some peaks might be indistinguishable from others or side lobes. This can be seen in fig. 2 on the left side, where the red cross-hair markings correspond to the original frequency and amplitude, and the blue waveform to the frequency representation of the sum of signals. It is easy to see that the original frequencies and amplitudes can not be read out - only a rough estimation on where the frequencies might be is possible by taking into account the location of the highest peaks.

Computing the spectral window of the given frequency representation in fig. 2 on the left, reveals

yea, what does the spectral window tell me?

The appropriate MATLAB code used to create the initial data set and calculate the frequency representations can be found in appendix B.

3 Sparse representation with greedy algorithm

3.1 Pre-whitening or Matching Pursuit (MP) algorithm

3.2 Orthogonal Matching Pursuit (OMP) algorithm

3.3 Orthogonal Least Square (OLS)

4 Sparse representation with convex relaxation

A Initial Data Set

```
f_th =    31.0120    32.6750    33.2830    33.5210    35.6090
A_th =     0.2500     0.7500     1.0000     0.7500     1.0000
phi_th =  0.3930    0.9960    0.4920    0.2810    0.5960
t =      1291.2      ...   1295.6 [514 x 1]
y =       -0.1648      ...   -0.4816 [514 x 1]
```

B MATLAB Code

```
1 close all; clc; clear all;
2 load('data.mat')
3 SAVEDATA = false;
4
5 x = 0;
6 SNR = 10;
7
8 for k=1 : 5
9     x = x + (A_th(k) * sin(2*pi*f_th(k)*t + phi_th(k)))
10 end
11
12 % calculate noise amplitude
13 pms = sumsqr(x)/length(x);
14 % standard Deviation
15 sigma = sqrt(pms/10);
16
17 noise = sigma * randn(1,length(x));
18 x_n = x + transpose(noise)
19
20 figure
21 plot(t,x,'g+')
22 hold on;
23 plot(t,x_n,'r+')
24 legend('real data',sprintf('data with gaussian noise, stdDev: %0.3f',
    ,sigma))
25 title('Herbig star HD 104237 data samples')
26 ylabel('signal amplitude')
27 xlabel('time')
28 if SAVEDATA
29     set(gcf, 'PaperUnits', 'points');
30     set(gcf, 'PaperPosition', [0 0 900 450]);
```

```

31     saveas(gcf, '../images/data.png')
32 end
33
34 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
35 % 2.2 irregular sampling case
36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37 fmax = 100;
38 M = 1024;
39 N = length(x_n);
40 freq = (-M:M)/M*fmax;
41 W=exp(2*j*pi*t*freq);
42 periodogram = abs(W*x_n)/N;
43
44 figure
45 subplot(2,2,1)
46 plot(freq,periodogram)
47 % plot real frequencies
48 hold on
49 plot(f_th, A_th/2, 'r+')
50 hold on
51 plot((-1.*f_th), A_th/2, 'r+')
52 xlabel('frequency')
53 ylabel('amplitude')
54 title('frequency representation')
55
56 subplot(2,2,3)
57 plot(freq,periodogram)
58 hold on;
59 plot(f_th, A_th/2, 'r+')
60 xlim([28 40])
61 xlabel('frequency')
62 ylabel('amplitude')
63 legend('frequency distribution','original frequencies')
64
65 % plot spectral window
66 subplot(2,2,[2 4])
67 Win=W*ones(N,1)/(N);
68 plot(freq,(Win))
69 xlim([-20 20])
70 title('spectral window')
71 xlabel('frequency')
72 ylabel('amplitude')

```

```

73
74 if SAVEDATA
75     set(gcf, 'PaperUnits', 'points');
76     set(gcf, 'PaperPosition', [0 0 900 450]);
77     saveas(gcf, '../images/data_freq.png')
78 end
79
80 error('ernsafbashjnfasrg')
81 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
82 % 3.1 Matching Pursuit Algorithm
83 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
84 r_n = x_n;
85 Gamma0 = [];
86 a = zeros(2049,1);
87 tau = chisqq(0.95,N)
88 T = tau +1;
89 k = 1;
90
91 while T > tau
92     W_current = W(:,k);
93     [val, k] = max(abs(W*r_n));
94     Gamma0 = [Gamma0 k];
95     a(k) = a(k) + (1/((W_current')*W_current))*W_current'*r_n;
96     r_n = r_n - (((1/(W_current'*W_current)).*W_current'*r_n).*
97         W_current);
98     T = (norm(r_n)^2)/(sigma^2);
99 end
100 MethodOneIterations = size(Gamma0);
101 [MaxMP,MaxIdxMP] = findpeaks(abs(a),'MinPeakHeight',0.1);
102
103 figure
104 subplot(2,1,1)
105 plot(freq,abs(a));
106 hold on;
107 plot(f_th, A_th/2, 'r+')
108 hold on;
109 plot((freq(MaxIdxMP)),MaxMP, 'bo')
110 for num=1:length(f_th)
111     hold on;
112     line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r','
113         LineStyle','—')
114 end

```

```

113 xlim([28 40])
114 legend('reconstructed frequency','original frequencies')
115 xlabel('frequency')
116 ylabel('amplitude')
117 subplot(2,1,2)
118 plot(t,W*a,'+')
119 subtitle('Matching Pursuit (pre-whitening) algorithm')
120 if SAVEDATA
121     set(gcf, 'PaperUnits', 'points');
122     set(gcf, 'PaperPosition', [0 0 900 450]);
123     saveas(gcf, '../images/mp.png')
124 end
125
126
127 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
128 % 3.2 Orthogonal Matching Pursuit Algorithm
129 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
130 r_n = x_n;
131 Gamma0 = [];
132 W_g = [];
133 a = [];
134 tau = chisqq(0.95,N)
135 T = tau +1;
136 k = 1;
137
138 while T > tau
139     [val, k] = max(abs(W*r_n))
140     Gamma0 = [Gamma0 k];
141
142     W_g = [];
143     for l=1:length(Gamma0)
144         W_g = [W_g W(:,Gamma0(l))];
145     end
146
147     a = ((W_g'*W_g)^(-1))*W_g'*x_n;
148     %size(a)
149     %a_vec = [a_vec a];
150
151     r_n = x_n - W_g*a;
152     T = (norm(r_n)^2)/(sigma^2)
153 end
154 a_plot = zeros(2049,1);

```



```

155 for ind=1:length(Gamma0)
156     a_plot(Gamma0(ind)) = a(ind);
157 end
158 MethodTwoIterations = size(Gamma0);
159 [MaxOMP,MaxIdxOMP] = findpeaks(abs(a_plot),'MinPeakHeight',0.1);
160
161 figure
162 subplot(2,1,1)
163 plot(freq,abs(a_plot));
164 hold on;
165 plot(f_th,A_th/2,'r+')
166 hold on;
167 plot((freq(MaxIdxOMP)),MaxOMP,'bo')
168 for num=1:length(f_th)
169     hold on;
170     line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r','
        LineStyle','—')
171 end
172 xlim([28 40])
173 legend('reconstructed frequency','original frequencies')
174 xlabel('frequency')
175 ylabel('amplitude')
176 subplot(2,1,2)
177 plot(t,W*a_plot,'+')
178 xlabel('time')
179 ylabel('amplitude')
180 supitle('Orthogonal Matching Pursuit algorithm');
181 if SAVE_DATA
182     set(gcf,'PaperUnits','points');
183     set(gcf,'PaperPosition',[0 0 900 450]);
184     saveas(gcf,'../images/omp.png')
185 end
186
187
188
189
190 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
191 % 3.3 Orthogonal Least Square
192 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
193 r_n = x_n;
194 Gamma0 = [];
195 W_g = [];

```

```

196 a = 0;
197 tau = chisqq(0.95,N)
198 T = tau +1;
199 k = 1;
200 test = (sigma^2)*tau;
201 a_vec = [];
202 while T > tau
203     [val, k] = ols(W,x_n,Inf,test);
204     Gamma0 = [Gamma0 k];
205
206     W_g = [];
207     for l=1:length(Gamma0)
208         W_g = [W_g W(:,Gamma0(l))];
209     end
210
211     a = ((W_g'*W_g)^(-1))*W_g'*x_n;
212     a_vec = [a_vec a];
213
214     r_n = x_n - W_g*a;
215     T = (norm(r_n)^2)/(sigma^2)
216 end
217
218 a_plot = zeros(2049,1);
219 for ind=1:length(Gamma0)
220     a_plot(Gamma0(ind)) = a_vec(ind);
221 end
222 MethodThrIterations = size(Gamma0);
223 [MaxOLS,MaxIdxOLS] = findpeaks(abs(a_plot),'MinPeakHeight',0.1);
224
225 figure
226 subplot(2,1,1)
227 plot(freq,abs(a_plot));
228 hold on;
229 plot(f_th,A_th/2,'r+')
230 hold on;
231 plot((freq(MaxIdxOLS)),MaxOLS,'bo')
232 for num=1:length(f_th)
233     hold on;
234     line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r','LineStyle','—')
235 end
236 xlim([28 40])

```

```

237 xlabel('frequency')
238 ylabel('amplitude')
239 legend('reconstructed frequency','original frequencies')
240 subplot(2,1,2)
241 plot(t,W*a_plot,'+')
242 xlabel('time')
243 ylabel('amplitude')
244 supitle('Orthogonal Least Square');
245 if SAVEDATA
246     set(gcf,'PaperUnits','points');
247     set(gcf,'PaperPosition',[0 0 900 450]);
248     saveas(gcf,'../images/ols.png')
249 end
250
251
252 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
253 % 4 Sparse representation with convex relation
254 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
255 lambda_max = max(abs(W*x_n));
256 lambda = 0.06 * lambda_max;
257 n_it_max = 100000;
258
259 a1 = min_L2_L1_0(x_n,W,lambda,n_it_max);
260
261 [MaxSparse,MaxIdxSparse] = findpeaks(abs(a1),'MinPeakHeight',0.03);
262
263 figure
264 subplot(2,1,1)
265 plot(freq,abs(a1));
266 hold on;
267 plot(f_th,A_th/2,'r+')
268 hold on;
269 plot((freq(MaxIdxSparse)),MaxSparse,'bo')
270 for num=1:length(f_th)
271     hold on;
272     line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r','LineStyle','—')
273 end
274 xlim([28 40])
275 xlabel('frequency')
276 ylabel('amplitude')
277 legend('reconstructed frequency','original frequencies')

```

```

278 subplot(2,1,2)
279 plot(t,W*a1,'+')
280 xlabel('time')
281 ylabel('amplitude')
282 title('Sparse representation with convex relaxation');
283 if SAVEDATA
284     set(gcf, 'PaperUnits', 'points');
285     set(gcf, 'PaperPosition', [0 0 900 450]);
286     saveas(gcf, '../images/convex.png')
287 end
288
289 % Save detected data to file:
290 if SAVEDATA
291     fileID = fopen('../images/img_data.txt','w');
292     fprintf(fileID, 'frequency ; amplitude\n ');
293     fprintf(fileID, '# Matching Pursuit, %d iterations\n', (
        MethodOneIterations(2)));
294     fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxMP)
        )] , [ freq(MaxIdxMP).' MaxMP]. ' ');
295     fprintf(fileID, '# Orthogonal Matching Pursuit, %d iterations\n'
        , (MethodTwoIterations(2)));
296     fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxOMP)
        )] , [ freq(MaxIdxOMP).' MaxOMP]. ' ');
297     fprintf(fileID, '# Orthogonal Least Square, %d iterations\n', (
        MethodThrIterations(1)));
298     fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxOLS)
        )] , [ freq(MaxIdxOLS).' MaxOLS]. ' ');
299     fprintf(fileID, '# Convex Relaxation\n');
300     fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(
        MaxIdxSparse))] , [ freq(MaxIdxSparse).' MaxSparse]. ' ');
301     fclose(fileID);
302 end

```