



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

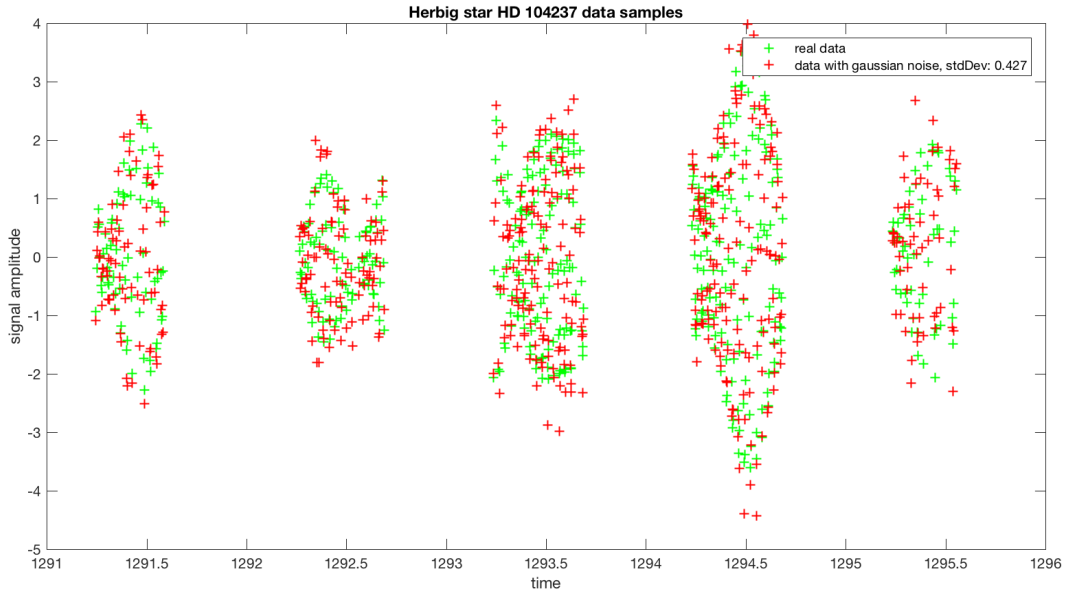


Figure 1: Original data samples and samples with additional noise in time domain

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.

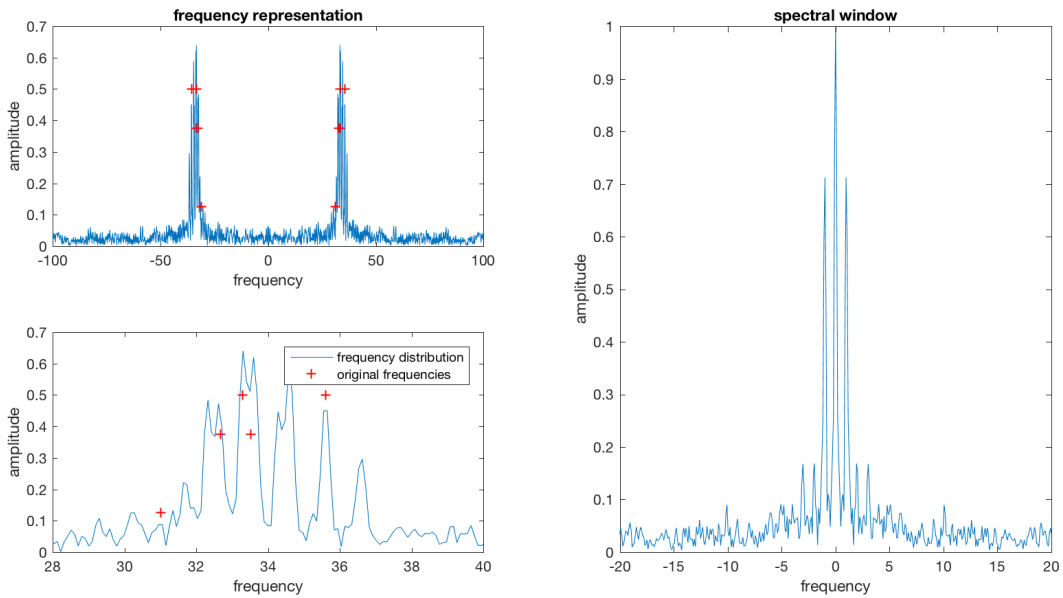


Figure 2: Frequency representation with original frequencies and spectral window

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. The phase, which essentially is an offset of the signal, can not be directly read out from the amplitude vs. frequency diagram,

but can be calculated from the Fourier-transformed data. The phase will also be affected by the noise intentionally introduced in the data set.

However, if a noisy sum of 5 sine signals is used, the underlying frequencies and amplitudes can not be easily determined anymore. This is due to the fact that the sum of signals and their respective noise adds 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 are located might be possible by taking into account the location of the highest peaks.

The spectral window corresponding to the sampling time is shown in fig. 2 on the right. The spectral window as represented in the figure shows the Fourier Transform of a rectangular window (basically consisting of  $\delta$ -functions). The main peak corresponds to the main frequency peak in the frequency domain of the signal - which can be useful in case of truncating specific frequencies or weighting the signal, since the spectral window (in this case similar to a sinc function) can cut off frequencies above the cutoff frequency when convoluted with the Fourier Transformed signal. This corresponds

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/SNR);
16 % add noise to the calculations
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*i*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,abs(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 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
81 % 3.1 Matching Pursuit Algorithm
82 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
83 r_n = x_n;
84 Gamma0 = [];
85 a = zeros(2049,1);
86 tau = chisq(0.95,N);
87 T = tau +1;
88 k = 1;
89
90 while T > tau
91     W_current = W(:,k);
92     [val, k] = max(abs(W*r_n));
93     Gamma0 = [Gamma0 k];
94     a(k) = a(k) + (1/((W_current')*W_current))*W_current'*r_n;
95     r_n = r_n - (((1/(W_current'*W_current)).*W_current'*r_n).*
96         W_current);
97     T = (norm(r_n)^2)/(sigma^2);
98 end
99 MethodOneIterations = size(Gamma0);
100 [MaxMP,MaxIdxMP] = findpeaks(abs(a), 'MinPeakHeight', 0.1);
101
102 figure
103 subplot(2,1,1)
104 plot(freq,abs(a));
105 hold on;
106 plot(f_th, A_th/2, 'r+');
107 hold on;
108 plot((freq(MaxIdxMP)),MaxMP, 'bo')
109 for num=1:length(f_th)
110     hold on;
111     line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r', '
112         LineStyle','—')
113 end
114 xlim([28 40])

```

```

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

```



```

155 [MaxOMP,MaxIdxOMP] = findpeaks(abs(a_plot),'MinPeakHeight',0.1);
156
157 figure
158 subplot(2,1,1)
159 plot(freq,abs(a_plot));
160 hold on;
161 plot(f_th,A_th/2,'r+')
162 hold on;
163 plot((freq(MaxIdxOMP)),MaxOMP,'bo')
164 for num=1:length(f_th)
165     hold on;
166     line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r','LineStyle','—')
167 end
168 xlim([28 40])
169 legend('reconstructed frequency','original frequencies')
170 xlabel('frequency')
171 ylabel('amplitude')
172 subplot(2,1,2)
173 plot(t,W*a_plot,'+')
174 xlabel('time')
175 ylabel('amplitude')
176 supitle('Orthogonal Matching Pursuit algorithm');
177 if SAVEDATA
178     set(gcf,'PaperUnits','points');
179     set(gcf,'PaperPosition',[0 0 900 450]);
180     saveas(gcf,'../images/omp.png')
181 end
182
183
184
185
186 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
187 % 3.3 Orthogonal Least Square
188 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
189 r_n = x_n;
190 Gamma0 = [];
191 W_g = [];
192 a = 0;
193 tau = chisqq(0.95,N);
194 T = tau +1;
195 k = 1;

```

```

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

```

```

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

```

```

278 supitle('Sparse representation with convex relaxation');
279 if SAVEDATA
280     set(gcf, 'PaperUnits', 'points');
281     set(gcf, 'PaperPosition', [0 0 900 450]);
282     saveas(gcf, '../images/convex.png')
283 end
284
285 % Save detected data to file:
286 if SAVEDATA
287     fileID = fopen('../images/img_data.txt','w');
288     fprintf(fileID, 'frequency ; amplitude\n ');
289     fprintf(fileID, '# Matching Pursuit, %d iterations\n', (
        MethodOneIterations(2)));
290     fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxMP)
        )] , [ freq(MaxIdxMP).' MaxMP]. ');
291     fprintf(fileID, '# Orthogonal Matching Pursuit, %d iterations\n'
        , (MethodTwoIterations(2)));
292     fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxOMP)
        )] , [ freq(MaxIdxOMP).' MaxOMP]. ');
293     fprintf(fileID, '# Orthogonal Least Square, %d iterations\n', (
        MethodThrIterations(1)));
294     fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxOLS)
        )] , [ freq(MaxIdxOLS).' MaxOLS]. ');
295     fprintf(fileID, '# Convex Relaxation\n');
296     fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(
        MaxIdxSparse))] , [ freq(MaxIdxSparse).' MaxSparse]. ');
297     fclose(fileID);
298 end

```