



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

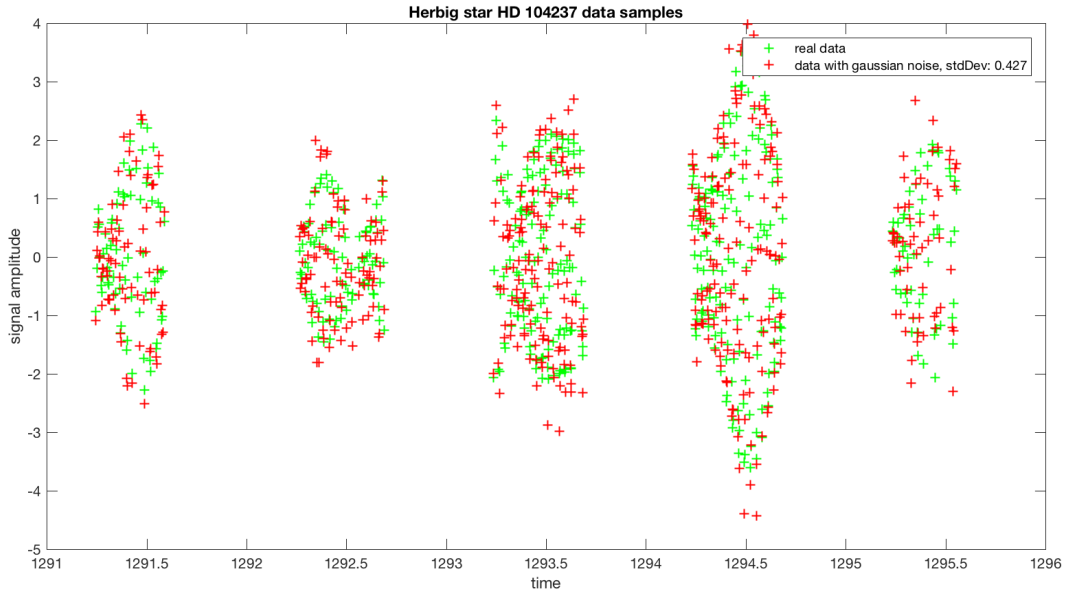


Figure 1: Original data samples and samples with additional noise in time domain

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.

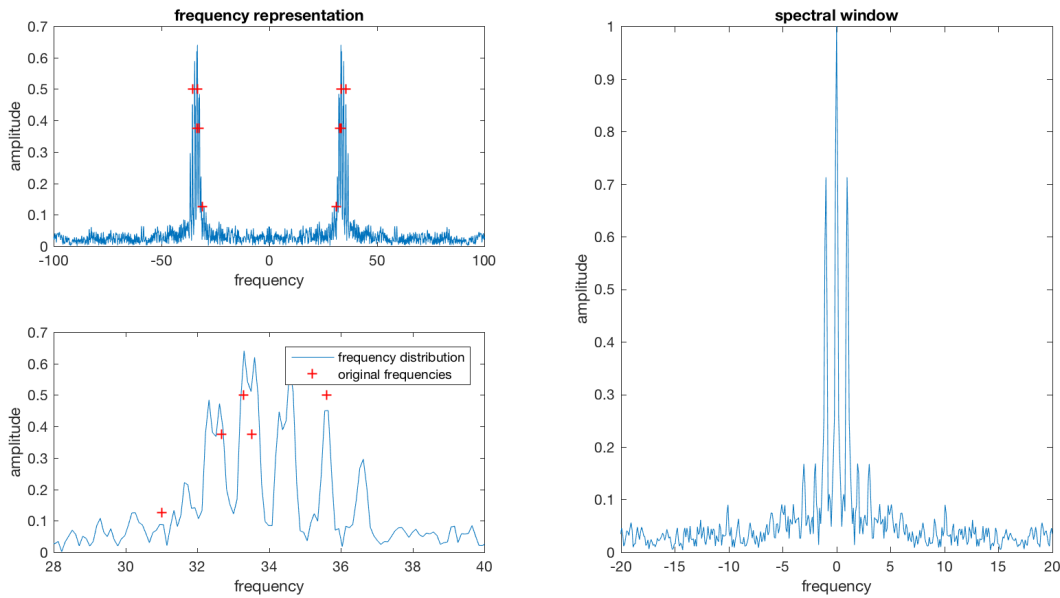


Figure 2

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

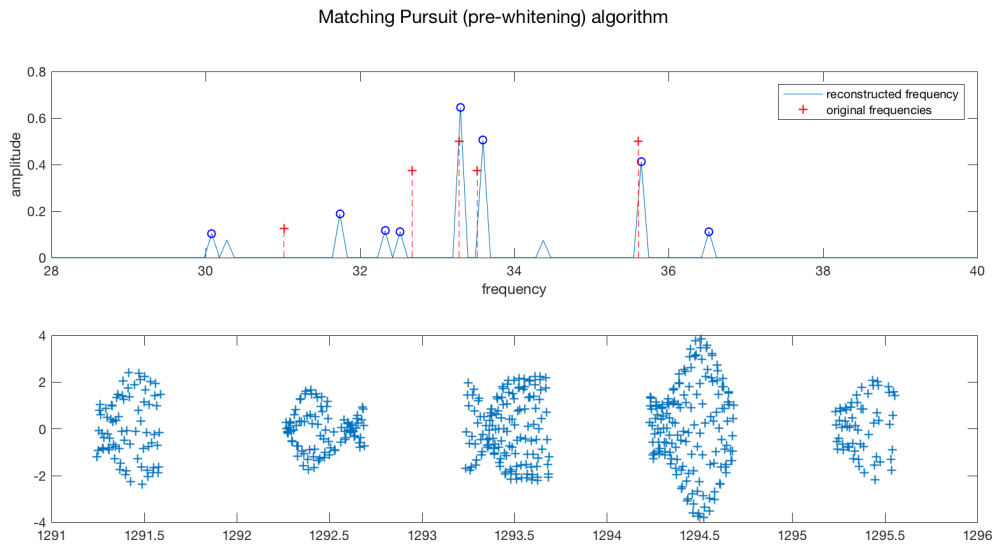


Figure 3

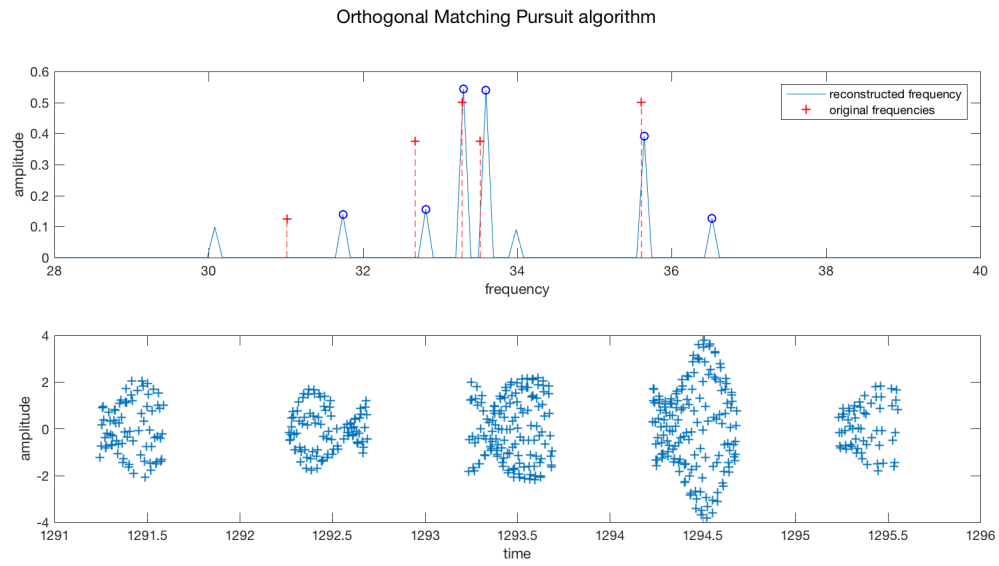


Figure 4

3.2 Orthogonal Matching Pursuit (OMP) algorithm

3.3 Orthogonal Least Square (OLS)

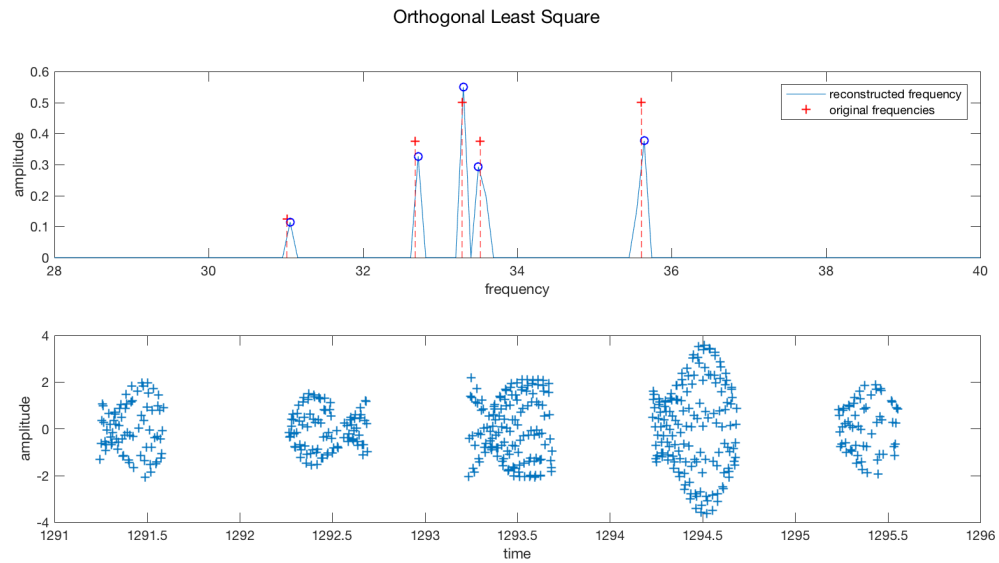


Figure 5

4 Sparse representation with convex relaxation

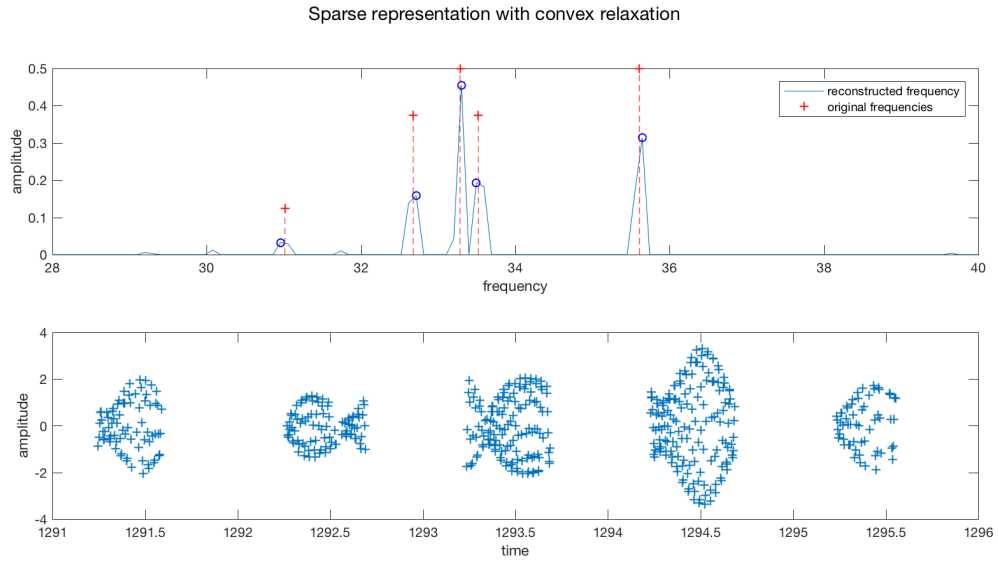


Figure 6

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 SAVEADATA = 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 SAVEADATA
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,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     %size(a)
148     %a_vec = [a_vec a];
149
150     r_n = x_n - W_g*a;
151     T = (norm(r_n)^2)/(sigma^2)
152 end
153 a_plot = zeros(2049,1);
154 for ind=1:length(Gamma0)

```

```

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

```

```

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

```

```

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

```

```

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

```