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

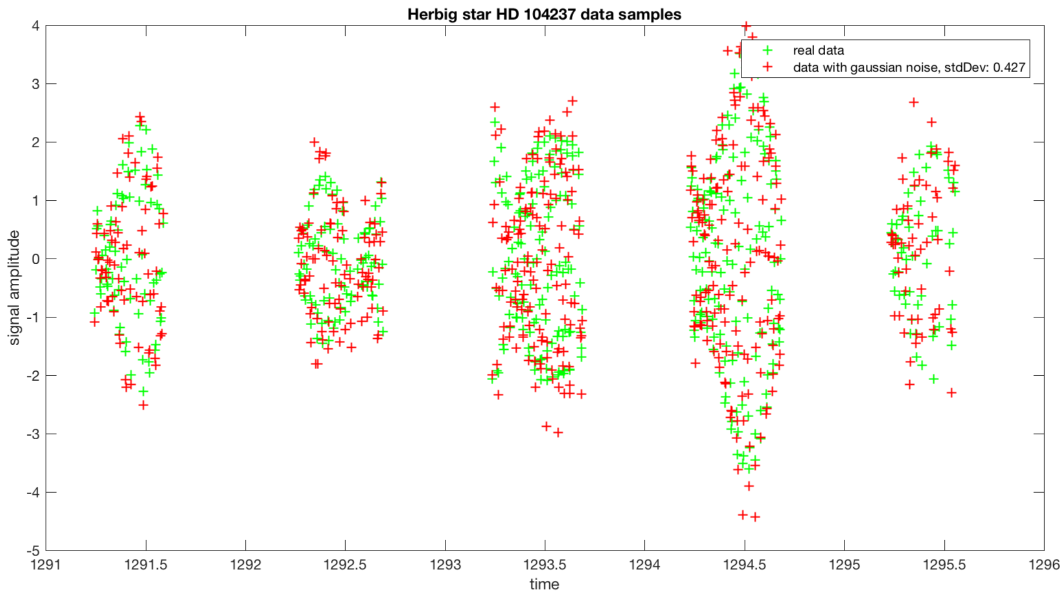

Figure 1: Original data samples and samples with additional noise in time domain

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

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

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 $\boldsymbol{W}$, such as

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

with $\hat{\boldsymbol{x}}$ being the Fourier-Transform of vector $\boldsymbol{x}$. As stated above, the FFT can not be used here, since the Matrix $\boldsymbol{W}$ 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.

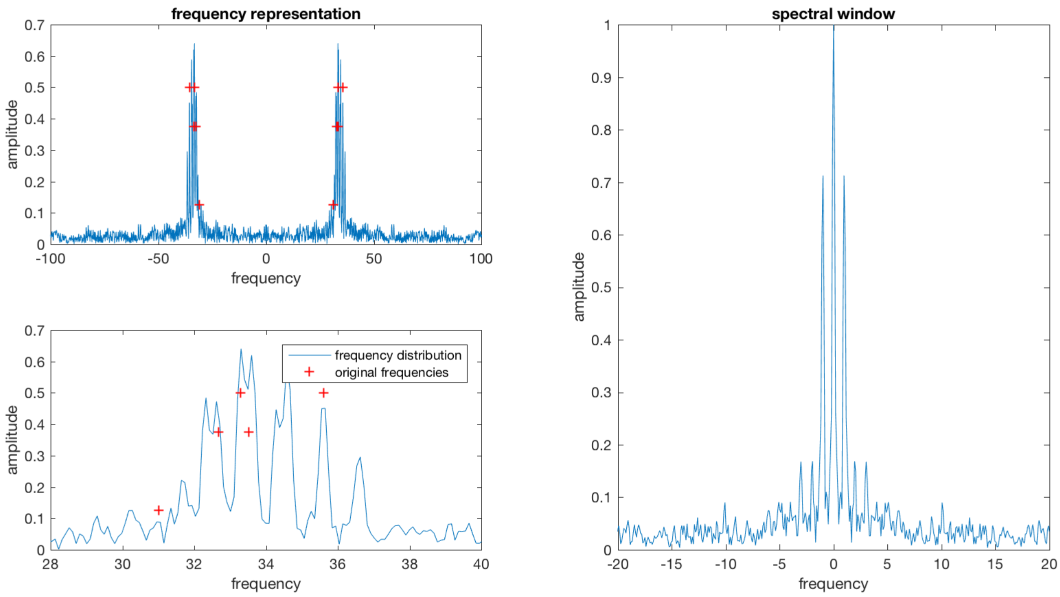

Figure 2: Frequency representation with original frequencies and spectral window

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 is 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 δ-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

```
close all; clc; clear all;
load('data.mat')
SAVEDATA = false;

x = 0;
SNR = 10;

for k=1 : 5
    x = x + (A_th(k) * sin(2*pi*f_th(k)*t + phi_th(k)));
end

% calculate noise amplitude
pms = sumsqr(x)/length(x);
% standard Deviation
sigma = sqrt(pms/SNR);
% add noise to the calculations
noise =  sigma * randn(1,length(x));
x_n = x + transpose(noise);

figure
plot(t,x,'g+')
hold on;
plot(t,x_n,'r+')
legend('real data',sprintf('data with gaussian noise, stdDev: %0.3f',sigma))
title('Herbig star HD 104237 data samples')
ylabel('signal amplitude')
xlabel('time')
if SAVEDATA
    set(gcf, 'PaperUnits', 'points');
    set(gcf, 'PaperPosition', [0 0 900 450]);
```

```
    saveas(gcf,'../images/data.png')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2.2  irregular sampling case
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fmax = 100;
M = 1024;
N = length(x_n);
freq =(-M:M)/M*fmax;
W=exp(2*1i*pi*t*freq);
periodogram =  abs(W'*x_n)/N;

figure
subplot(2,2,1)
plot(freq,periodogram)
% plot real frequencies
hold on
plot(f_th,A_th/2,'r+')
hold on
plot((-1.*f_th),A_th/2,'r+')
xlabel('frequency')
ylabel('amplitude')
title('frequency representation')

subplot(2,2,3)
plot(freq,periodogram)
hold on;
plot(f_th,A_th/2,'r+')
xlim([28 40])
xlabel('frequency')
ylabel('amplitude')
legend('frequency distribution','original frequencies')

% plot spectral window
subplot(2,2,[2 4])
Win=W'*ones(N,1)/N;
plot(freq,abs(Win))
xlim([-20 20])
title('spectral window')
xlabel('frequency')
ylabel('amplitude')
```

```
if SAVEDATA
    set(gcf, 'PaperUnits', 'points');
    set(gcf, 'PaperPosition', [0 0 900 450]);
    saveas(gcf,'../images/data_freq.png')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3.1 Matching Pursuit Algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r_n = x_n;
Gamma0 = [];
a = zeros(2049,1);
tau = chisqq(0.95,N);
T = tau +1;
k = 1;

while T > tau
    W_current = W(:,k);
    [val, k] = max(abs(W'*r_n));
    Gamma0 = [Gamma0 k];
    a(k) = a(k) + (1/((W_current')*W_current))*W_current'*r_n;
    r_n = r_n - (((1/(W_current'*W_current)).*W_current'*r_n).*W_current);
    T = (norm(r_n)^2)/(sigma^2);
end
MethodOneIterations = size(Gamma0);
[MaxMP,MaxIdxMP] = findpeaks(abs(a),'MinPeakHeight',0.1);

figure
subplot(2,1,1)
plot(freq,abs(a));
hold on;
plot(f_th,A_th/2,'r+')
hold on;
plot((freq(MaxIdxMP)),MaxMP,'bo')
for num=1:length(f_th)
   hold on;
   line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r','LineStyle','--')
end
xlim([28 40])
```

```
legend('reconstructed frequency','original frequencies')
xlabel('frequency')
ylabel('amplitude')
subplot(2,1,2)
plot(t,W*a,'+')
suptitle('Matching Pursuit (pre-whitening) algorithm')
if SAVEDATA
    set(gcf, 'PaperUnits', 'points');
    set(gcf, 'PaperPosition', [0 0 900 450]);
    saveas(gcf,'../images/mp.png')
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3.2 Orthogonal Matching Pursuit Algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r_n = x_n;
Gamma0 = [];
W_g = [];
a = [];
tau = chisqq(0.95,N);
T = tau +1;
k = 1;

while T > tau
    [val, k] = max(abs(W'*r_n))
    Gamma0 = [Gamma0 k];

    W_g = [];
    for l=1:length(Gamma0)
        W_g = [W_g W(:,Gamma0(l))];
    end

    a = ((W_g'*W_g)^(-1))*W_g'*x_n;
    r_n = x_n - W_g*a;
    T = (norm(r_n)^2)/(sigma^2);
end
a_plot = zeros(2049,1);
for ind=1:length(Gamma0)
   a_plot(Gamma0(ind)) = a(ind);
end
MethodTwoIterations = size(Gamma0);
```

```
[MaxOMP,MaxIdxOMP] = findpeaks(abs(a_plot),'MinPeakHeight',0.1);

figure
subplot(2,1,1)
plot(freq,abs(a_plot));
hold on;
plot(f_th,A_th/2,'r+')
hold on;
plot((freq(MaxIdxOMP)),MaxOMP,'bo')
for num=1:length(f_th)
    hold on;
   line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r','
      LineStyle','--')
end
xlim([28 40])
legend('reconstructed frequency','original frequencies')
xlabel('frequency')
ylabel('amplitude')
subplot(2,1,2)
plot(t,W*a_plot,'+')
xlabel('time')
ylabel('amplitude')
suptitle('Orthogonal Matching Pursuit algorithm');
if SAVEDATA
    set(gcf, 'PaperUnits', 'points');
    set(gcf, 'PaperPosition', [0 0 900 450]);
    saveas(gcf,'../images/omp.png')
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3.3 Orthogonal Least Square
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r_n = x_n;
Gamma0 = [];
W_g = [];
a = 0;
tau = chisqq(0.95,N);
T = tau +1;
k = 1;
```

```
test = (sigma^2)*tau;
a_vec = [];
while T > tau
    [val, k] = ols(W,x_n,Inf,test);
    Gamma0 = [Gamma0 k];

    W_g = [];
    for l=1:length(Gamma0)
        W_g = [W_g W(:,Gamma0(l))];
    end

    a = ((W_g'*W_g)^(-1))*W_g'*x_n;
    a_vec = [a_vec a];

    r_n = x_n - W_g*a;
    T = (norm(r_n)^2)/(sigma^2);
end

a_plot = zeros(2049,1);
for ind=1:length(Gamma0)
    a_plot(Gamma0(ind)) = a_vec(ind);
end
MethodThrIterations = size(Gamma0);
[MaxOLS,MaxIdxOLS] = findpeaks(abs(a_plot),'MinPeakHeight',0.1);

figure
subplot(2,1,1)
plot(freq,abs(a_plot));
hold on;
plot(f_th,A_th/2,'r+')
hold on;
plot((freq(MaxIdxOLS)),MaxOLS,'bo')
for num=1:length(f_th)
    hold on;
    line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r','
        LineStyle','--')
end
xlim([28 40])
xlabel('frequency')
ylabel('amplitude')
legend('reconstructed frequency','original frequencies')
subplot(2,1,2)
```

```
plot(t,W*a_plot,'+')
xlabel('time')
ylabel('amplitude')
suptitle('Orthogonal Least Square');
if SAVEDATA
    set(gcf, 'PaperUnits', 'points');
    set(gcf, 'PaperPosition', [0 0 900 450]);
    saveas(gcf,'../images/ols.png')
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 4 Sparse representation with convex relation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lambda_max = max(abs(W'*x_n));
lambda = 0.06 * lambda_max;
n_it_max = 100000;

a1 = min_L2_L1_0(x_n,W,lambda,n_it_max);

[MaxSparse,MaxIdxSparse] = findpeaks(abs(a1),'MinPeakHeight',0.03);

figure
subplot(2,1,1)
plot(freq,abs(a1));
hold on;
plot(f_th,A_th/2,'r+')
hold on;
plot((freq(MaxIdxSparse)),MaxSparse,'bo')
for num=1:length(f_th)
   hold on;
   line([f_th(num) f_th(num)], [0 A_th(num)/2], 'Color','r','
       LineStyle','--')
end
xlim([28 40])
xlabel('frequency')
ylabel('amplitude')
legend('reconstructed frequency','original frequencies')
subplot(2,1,2)
plot(t,W*a1,'+')
xlabel('time')
ylabel('amplitude')
```

```
suptitle('Sparse representation with convex relaxation');
if SAVEDATA
    set(gcf, 'PaperUnits', 'points');
    set(gcf, 'PaperPosition', [0 0 900 450]);
    saveas(gcf,'../images/convex.png')
end

% Save detected data to file:
if SAVEDATA
    fileID = fopen('../images/img_data.txt','w');
    fprintf(fileID, 'frequency ; amplitude\n ');
    fprintf(fileID, '# Matching Pursuit, %d iterations\n', (MethodOneIterations(2)));
    fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxMP))] , [ freq(MaxIdxMP).' MaxMP].');
    fprintf(fileID, '# Orthogonal Matching Pursuit, %d iterations\n', (MethodTwoIterations(2)));
    fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxOMP))] , [ freq(MaxIdxOMP).' MaxOMP].');
    fprintf(fileID, '# Orthogonal Least Square, %d iterations\n', (MethodThrIterations(1)));
    fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxOLS))] , [ freq(MaxIdxOLS).' MaxOLS].');
    fprintf(fileID, '# Convex Relaxation\n');
    fprintf(fileID, [repmat(' %0.3f ; %0.3f \n', 1, length(MaxIdxSparse))] , [ freq(MaxIdxSparse).' MaxSparse].');
    fclose(fileID);
end
```