# Spectral analysis and sparse representations laboratory

**The listings of all programs have to be given with your laboratory report.**

**All results and graphs have to be explained.**

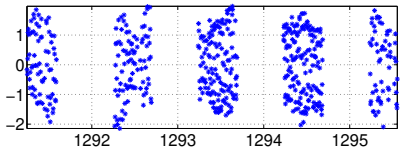
## I Introduction

The aim of this laboratory is to illustrate simultaneously the spectral analysis of irregularly sampled data, in particular for line spectra, and the sparse representations of signals.

In a first step we will study the problems related to spectral analysis with the Fourier Transform for the frequently encountered case in astronomy of irregularly sampled data.

Then, we will illustrate the behaviour of classical methods for sparse representations of signals for this very difficult case of line spectra analysis from irregularly sampled data. We will study both "greedy" algorithms and "convex relaxation" approaches (penalizing the least squares cost function with the $\ell^1$-norm). We will illustrate the use of these tools on a simulated example corresponding to a realistic astronomical data set.

Vectors `t` and `y` available in the `data.mat` file (`load data.mat`) correspond to the sampling time and radial velocities (shown on the opposite image) of a 5 nights observation of the Herbig star HD 104237 (from which the orbital effects due to the binary stars has been subtracted).




As it is very difficult to understand the signal processing methods working directly on real data, we will simulate a realistic data set corresponding to a noisy sum of sine functions irregularly sampled for time $\{t_n\}_{n=1\dots N}$:

$$x(t_n) = \sum_{k=1}^{K} A_k \cos(2\pi f_k t_n + \phi_k) + \epsilon_n. \tag{1}$$

with $K = 5$ sine functions of parameters given in the vectors of frequencies `f_th` (in day$^{-1}$), of amplitudes `A_th,` and of phase `phi_th` (in radian), with:

- `f_th = [31.012 32.675 33.283 33.521 35.609]`,
- `A_th = [.25 .75 1 .75 1]`,
- `phi_th = [0.393 0.996 0.492 0.281 0.596]`.

In practice, we will consider independent and identically distributed centred Gaussian noise $\epsilon_n$ with a signal to noise ratio of 10 in power mean (20 dB).

1† For a given data set (stored in a vector `y`) give the expression of the noise variance $\sigma^2$ to have a 20dB Signal to Noise Ratio. Remember that $\mathrm{SNR}_{\mathrm{dB}} = 10 \log 10 \frac{P_y}{P_\epsilon}$ where $P_y$ and $P_\epsilon$ respectively correspond to the power mean of the data and of the noise.



2. **Generate two data sets** for this model from the given sampling time `t`: the first one `x_0` for a single sine function, the second one `x` with the given 5 sine functions. A single sine function sampled at time $\{t_n\}_{n=1\dots N}$ (stored un a vector `t`) can be generated using the code: `x0 = A*cos(2*pi*f0*t +phi);`

## II Spectral analysis with the Fourier Transform

### II.1 Estimation for known frequencies

The model of Eq. (1) can also be written with pure frequencies:

$$x(t_n) = \sum_{k=-K}^{K} c_k e^{2j\pi f_k t_n} + \epsilon_n, \tag{2}$$

with $f_{-k} = -f_k$ and $c_0$ the mean value of the signal (which is equal to zero in Eq. (1)). The model of Eq. (2) is linear with respect to the (complex) amplitude parameters but non linear with respect to the frequency parameters, while the model of Eq. (1) is also non linear with respect to the phase parameters. Therefore, it is easy to estimate the amplitudes when the frequencies are known, using the model of Eq. (2).

1† **Show** that model of Eq. (2) can be written with matrix notations $\underline{\boldsymbol{x}} = \mathbf{W}\underline{\boldsymbol{c}} + \underline{\boldsymbol{\epsilon}}$ when the frequencies $\{f_k\}$ of the model are known. **Specify** the dimension and the value of the components of matrix $\mathbf{W}$.

2† **Give** the expression of the maximum likelihood estimator of the amplitudes $\underline{\boldsymbol{c}}$ in such a case. Note that as $\mathbf{W}$ is a complex valued matrix, the equivalent of the transposed operator is the conjugate transposed operator $\mathbf{W}^\dagger$ (computed with `W'` in matlab).

3. **Write a Matlab code** to estimate the complex amplitudes from your data sets `x_0` and `x`. Can you retrieve the real amplitudes $A_k$ and the phase $\phi_k$ from these complex amplitudes $\underline{\boldsymbol{c}}$?

### II.2 Estimation for unknown frequencies

When the frequencies are unknown, one immediately think to use the Fourier Transform of the signal to estimate these frequencies. However, in the irregular sampling case, one cannot compute the Fourier transform with the FFT. A simple way to compute the Fourier transform for frequencies $\{f_c\}_c$ consists in introducing the matrix $\mathbf{G}$ such that $\mathbf{G}(r,c) = \exp(2j\pi t_r f_c)$. Vector $\underline{\widehat{\boldsymbol{x}}}$ corresponding to the Fourier transform of vector $\underline{\boldsymbol{x}}$ can be computed simply with $\underline{\widehat{\boldsymbol{x}}} = \mathbf{G}^\dagger \underline{\boldsymbol{x}}$. Note that in the case of the FFT, the matrix $\mathbf{G}$ is orthogonal ($\mathbf{G}^\dagger\mathbf{G} = \mathbf{G}\mathbf{G}^\dagger = N\boldsymbol{I}$) which is not the case for irregularly sampled data. With Matlab, to compute the Fourier Transform of a signal sampled at time in vector `t` (column vector), at frequencies given in a vector `freq_grid =(-M:M)/M*nu_max` (row vector with frequencies from 0 to `nu_max` with a frequency sampling period `nu_max/M`), the matrix $\mathbf{G}$ is computed with `G=exp(2*j*pi*t*freq_grid)` and the frequency representation (periodogram) of data $\underline{\boldsymbol{x}}$ is computed with `abs(G'*x)/N;`

1. What is the highest frequency which can be determined without errors from such data? With Matlab, **build** a vector of frequencies `freq_grid`, with a **thin frequency grid**, to compute and analyze the Fourier transform. Do not hesitate to account for a sufficiently **small frequency sampling period** − **large** `M`.

2. **Plot the time and frequency representations** of your data set `x_0` corresponding to a single sine function with frequency `f_0`. Can you determine without errors the parameters (frequency, amplitude and phase) of the sine function? Compare the result in the noisy and the noise free cases...

3. **Plot the time and frequency representations** of your data set `x` corresponding to the 5 sine functions. Can you clearly distinguish the 5 frequencies? Compare the result in the noisy and the noise free cases...

4. **Plot the spectral window** corresponding to these sampling time (it can be computed easily with `Win=G'*ones(N,1)/N;` where `N` is the number of samples of the data). Comment the shape of such a spectral window... How can you **interpret** the frequency representations of your data sets from this window?

When they study real valued irregularly sampled data, astronomers prefer using the Lomb-Scargle periodogram, which correspond to looking for sine functions (with positive frequencies) instead of the usual frequency representation which correspond to looking for complex exponentials (with positive and negative frequencies). However, for simplicity, we will continue our study using the usual frequency representation.

# III Sparse representation with greedy algorithms

## III.1 *Pre-whitenning* or *Matching Pursuit* (MP) algorithms

The previous example illustrates the limitations of spectral analysis with the Fourier transform from irregularly sampled data. Seeking to retrieve frequencies of sine functions from such kind of data, some astronomers proposed, as early as the 1970s, to use iterative techniques, sometime called *pre-whitenning* techniques. The principle of this techniques is very simple: it consists in computing the frequency maximizing the frequency representation of the data, to subtract the corresponding signal from the data, and to carry on the frequency analysis from this residual. Such iterative methods have been formalized in the 1990s in the more general framework of sparse representations, more specifically in terms of "greedy" algorithms, such as the *Matching Pursuit*. The matching pursuit algorithm is detailed Tab. 1. Note that in our case, one has: $\underline{\boldsymbol{g}}_\ell^\dagger \underline{\boldsymbol{g}}_\ell = N$, for any $\ell$th column $\underline{\boldsymbol{g}}_\ell$ of matrix $\mathbf{G}$ (`gl = G(:,l);`).

Various statistical tests can be used to decide when stopping the iterations. Hereafter, we will consider a simple test based on the residuals. More precisely, if the signal is correctly approximated, the residual $\underline{\boldsymbol{r}}$ is composed of centred Gaussian noise with a variance $\sigma^2$ (which is supposed to be known). In this case, the variable $T = ||\underline{\boldsymbol{r}}||^2/\sigma^2$ has a $\chi^2$ distribution with $N$ degrees of freedom. In practice, we will use as stopping rule $T < \tau$ with $\tau$ such that Proba$(T < \tau) = 95\%$ ($\tau$ can be computed with the Matlab function `tau = chisqq(0.95,N);`), which means that the residuals cannot be distinguished from noise with a 95% probability.



1. **Implement** the *Matching Pursuit* algorithm with Matlab to estimate the frequencies in your data.

2. For both datasets `x_0` and `x`, how many frequencies do you detect with this algorithm? Are these frequencies correctly localized? Are the corresponding amplitudes correctly estimated?

3. For dataset `x`, **analyze** the results obtained with this algorithm during the iterations and try to characterize its main drawbacks (order of detection of the frequencies, false detection...).

| | | |
|---|---|---|
| Initialization $n = 0$ | $\underline{\boldsymbol{r}}^{(0)} = \underline{\boldsymbol{x}}$, $\Gamma^{(0)} = \emptyset$, $\underline{\boldsymbol{c}} = \underline{\mathbf{0}}$. | |
| Iterations $n = 1 \dots$ | *a)* | Select the atom with index $\ell = \arg\max_\ell \lvert\underline{\boldsymbol{g}}_\ell^\dagger \underline{\boldsymbol{r}}_{n-1}\rvert$. |
| | | Update the list of indices $\Gamma^{(n)} = \Gamma^{(n-1)} \cup \{\ell\}$. |
| | *b)* | Update the amplitude $c_\ell = c_\ell + \frac{\underline{\boldsymbol{g}}_\ell^\dagger \underline{\boldsymbol{r}}^{(n-1)}}{\underline{\boldsymbol{g}}_\ell^\dagger \underline{\boldsymbol{g}}_\ell}$ |
| | | and the residuals $\underline{\boldsymbol{r}}^{(n)} = \underline{\boldsymbol{r}}^{(n-1)} - \frac{\underline{\boldsymbol{g}}_\ell^\dagger \underline{\boldsymbol{r}}^{(n-1)}}{\underline{\boldsymbol{g}}_\ell^\dagger \underline{\boldsymbol{g}}_\ell} \underline{\boldsymbol{g}}_\ell$. |

Table 1: Principle of the *Matching Pursuit* algorithm for the sparse representation of a signal $\underline{\boldsymbol{x}}$ in a dictionary (matrix) $\mathbf{G}$ ($\underline{\boldsymbol{x}} \approx \mathbf{G}\underline{\boldsymbol{c}}$ with sparse vector $\underline{\boldsymbol{c}}$).

## III.2 *Orthogonal Matching Pursuit* (OMP) algorithms

One drawback of the *Matching Pursuit* algorithm is due to the fact that only the amplitude of the newly selected atom is updated at each iteration. Thus, the algorithm can select the same atom at several iterations to obtain a correct amplitude estimation. This drawback is simple to correct. The update step *b)* has to be modified to update the amplitude of all the selected atoms using an orthogonal projection, with a least squares procedure, of the data on these atoms:

$$\underline{\boldsymbol{c}}(\Gamma^{(n)}) = \underset{\underline{\boldsymbol{c}}(\Gamma^{(n)})}{\arg\min} ||\underline{\boldsymbol{x}} - \mathbf{G}_{\Gamma^{(n)}} \underline{\boldsymbol{c}}(\Gamma^{(n)})||^2 = (\mathbf{G}_{\Gamma^{(n)}}^\dagger \mathbf{G}_{\Gamma^{(n)}})^{-1} \mathbf{G}_{\Gamma_n}^\dagger \underline{\boldsymbol{x}}$$

where $\boldsymbol{c}(\Gamma^{(n)})$ corresponds to the components of $\underline{\boldsymbol{c}}$ with indices $\Gamma^{(n)}$ and $\mathbf{G}_{\Gamma^{(n)}}$ corresponds to the sub-matrix of $\mathbf{G}$ with the columns of indices $\Gamma^{(n)}$. Then, the residual can be computed as: $\underline{\boldsymbol{r}}^{(n)} = \underline{\boldsymbol{x}} - \mathbf{G}_{\Gamma^{(n)}} \underline{\boldsymbol{c}}(\Gamma^{(n)})$.

1. Implement the *Orthogonal Matching Pursuit* algorithm with Matlab to estimate the frequencies in your data sets.

2. For both data sets `x_0` and `x`, how many frequencies do you detect with this algorithm? Are these frequencies correctly localized? Are the corresponding amplitudes correctly estimated?

3. For data set `x`, analyse the results obtained with this algorithm during the iterations and try to characterize its main drawbacks and to compare its results with the *Matching Pursuit* ones.

### III.3 *Orthogonal Least Square* (OLS)

The main drawback of the previous algorithms is due to the selection step. This step would be consistent if the columns of matrix $\mathbf{G}$ where orthogonal, which is not the case here. As the aim is to obtain the lowest possible approximation error, it may be interesting to account for such a property during the selection step. The *Orthogonal Least Square* algorithm (which is often confused with the OMP algorithm) consists in substituting the selection step *a)* of the OMP algorithm with the selection of index $k$ allowing to minimize the approximation error when the column $\underline{\boldsymbol{g}}_k$ is added to matrix $\mathbf{G}_{\Gamma^{(n)}}$:

$$\ell = \arg\min_{\ell} \; \arg\min_{\underline{\boldsymbol{c}}(\Gamma^{(n)}\cup\{\ell\})} ||\underline{\boldsymbol{x}} - \mathbf{G}_{\Gamma^{(n)}\cup\{\ell\}}\underline{\boldsymbol{c}}(\Gamma^{(n)}\cup\{\ell\})||^2.$$

Such an algorithm requires, of course, more computing time than the MP and OMP algorithms, especially if it implementation is not optimized. Therefore, you will not implement this algorithm but only analyse its results using the Matlab function `ols` using the command line: `[c, ind] = ols(G,x,Inf,test);` with `test` $= \sigma^2\tau$.

1. For both data sets `x_0` and `x`, how many frequencies do you detect with this algorithm? Are these frequencies correctly localized? Are the corresponding amplitudes correctly estimated?

2. For data set `x`, analyze the results obtained with this algorithm during the iterations and try to characterize its main drawbacks and to compare its results with the *Matching Pursuit* and *Orthogonal Matching Pursuit* ones.

## IV Sparse representation with convex relaxation

Another way to obtain a sparse approximation of a signal consists in minimizing a quadratic cost function penalized with a $\ell^1$-norm:

$$\underline{\boldsymbol{c}} = \arg\min_{\underline{\boldsymbol{c}}} ||\underline{\boldsymbol{x}} - \mathbf{G}\underline{\boldsymbol{c}}||^2 + \lambda||\underline{\boldsymbol{c}}||_1 \text{ with } ||\underline{\boldsymbol{c}}||_1 = \sum_k |c_k|.$$

Note that such problem corresponds to a convex relaxation of the original sparse approximation one which involves the $\ell^0$ pseudo-norm. Such a solution can also be viewed as a penalization approach to solve the inverse problem associated to $\underline{\boldsymbol{x}} = \mathbf{G}\underline{\boldsymbol{c}} + \underline{\boldsymbol{\epsilon}}$ using the penalization function $\Omega(\underline{\boldsymbol{x}}) = ||\underline{\boldsymbol{c}}||_1$. It can also be interpreted as a maximum *a posteriori* estimator of the problem for a Laplace prior on $\underline{\boldsymbol{c}}$.

Such a cost function is convex so has no local minima and, despite its non differentiability, many algorithms can be used to find a minimizer. You will use for that the optimization algorithm coded in the `min_l2_l1_0` function which can be called with the Matlab code: `c1 = min_L2_L1_0(x,G,lambda,n_it_max);` where `n_it_max` is the maximal number of iterations of the algorithm (Note that for this algorithm, the maximal number of iterations is not related to the number of detected frequencies, and it has to be high enough for the algorithm to converge toward the solution). With such an algorithm, there is no stopping rule in terms of number of detected frequencies, but a regularization parameter $\lambda$ to tune. It can be shown that the solution $\underline{\boldsymbol{c}}_{\text{opt}}$ satisfies: $|\mathbf{G}^\dagger\underline{\boldsymbol{r}}_{\text{opt}}| \leq \lambda$ with $\underline{\boldsymbol{r}}_{\text{opt}} = \underline{\boldsymbol{x}} - \mathbf{G}\underline{\boldsymbol{c}}_{\text{opt}}$, which means that the frequency representation of the residual is lower or equal to $\lambda/N$. The parameter $\lambda$ can then be tuned with respect to the maximal authorized value of the frequency representation of the residual.

1. For both data sets `x_0` and `x`, test this algorithm for various values of $\lambda$ until you obtain a satisfactory result (to check if the result is satisfactory you can have a look to the time and frequency representations of the residual as well as the value of the test $\tau$ used to stop the iterations of the previous algorithms).

2. For data set `x`, how many frequencies do you detect with this algorithm? Are these frequencies correctly localized? Are the corresponding amplitudes correctly estimated?

3. You can notice that the frequencies detected with this method are often arranged in pairs on the frequency grid, around the actual frequency values. By keeping only one of these pairs of frequencies, you can estimate with a least squares approach (as used in § II.1) the amplitudes of the corresponding frequencies. When doing so, how many frequencies do you detect with this algorithm? Are these frequencies correctly localized? Are the corresponding amplitudes correctly estimated?

## V Conclusion

Conclude on the considered problem and on the different approaches used trying to solve it... In practice what do you suggest to do?