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.

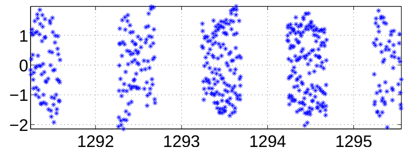
I Introduction

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 regularly sampled data as well as 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 ℓ^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 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 the data, we will simulate a realistic data set corresponding to a noisy sum of sine functions:

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

In practice, we will consider independent and identically distributed centred Gaussian noise ϵ_n with a signal to noise ratio of 10 in power mean (20 dB) and $K = 5$ sine functions of parameters given in the vectors of frequencies `f_th` (in day^{-1}), of amplitudes `A_th,` and of phase `phi_th`, 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]`.

II Spectral analysis with the Fourier Transform

II.1 Regular sampling case (optionnal)

We will first try to determine the parameters of a single sine function with the FFT. Although very simple, this problem can offer some surprises if the used tools are not well understood. . .

We will consider a regular sampling on the same time interval and for the same number of samples than data of the `data.mat` file.

1. What is the highest frequency which can be determined without errors from such data?
2. For such time samples, show the time and frequency (with FFT) representations of a single sine function with frequency `f_0=k_0*F_e/N;` whith any value for `k_0`. Can you determine without errors the parameters (frequency, amplitude and phase) of the sine function?
3. Do the same study for a sine function with frequency `f_0=(k_0+1/2)*F_e/N;` Compute the errors on the frequency and amplitude parameters? How can you explain such errors? Can you determine the phase of the sine function?
4. What do you suggest to improve such an analysis?
5. Add a second sine function (with the same amplitude) to the previous one. What is the minimal distance between the frequencies to distinguish them? How can you explain such a resolution? Do the same study for a ten times lower amplitude second sine function. Comment the obtained results. . .
6. For such regular sampling, build a signal corresponding to the 5 sine functions of the signal to study. Show it time and frequency representations (with FFT). Can you clearly distinguish the 5 frequencies? What do you suggest to do to improve such an analysis?

II.2 Irregular sampling case

In the irregular sampling case, one cannot compute the Fourier transform with the FFT. A simple way to compute the Fourier transform consists in introducing the matrix $\mathbf{W}$ such that $\boldsymbol{W}(\ell, c) = \exp(2j\pi t_\ell f_c)$. Vector $\widehat{\boldsymbol{x}}$ corresponding to the Fourier transform of vector $\boldsymbol{x}$ can be computed simply with $\widehat{\boldsymbol{x}} = \mathbf{W}^\dagger \boldsymbol{x}$. Note that in the case of the FFT, the matrix $\mathbf{W}$ is orthogonal ($\mathbf{W}^\dagger\mathbf{W} = \mathbf{W}\mathbf{W}^\dagger = N\boldsymbol{I}$) 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 =(-M:M)/M*fmax` (row vector with frequencies from 0 to `fmax` with a frequency sampling period `fmax/M`), the matrix $\mathbf{W}$ is computed with `W=exp(2*j*pi*t*freq)` and the frequency representation (periodogram) of data $\boldsymbol{x}$ is computed with `abs(W'*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` to compute and analyse the Fourier transform. Do not hesitate to account for a sufficiently small frequency sampling period.
2. For the considered time samples, show the time and frequency representations of a single sine function with any frequency `f_0.` Can you determine without errors the parameters (frequency, amplitude and phase) of the sine function?

3. For this time samples, build a data set corresponding to the 5 sine functions of the signal to study and show it time and frequency representations. Can you clearly distinguish the 5 frequencies?

4. Show the spectral window corresponding to these sampling time (it can be computed easily with `Win=W'*ones(N,1)/N;` where `N` is the number of samples of the signal). Comment the shape of such a spectral window... How van you interpret the frequency representation of the signal 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.
The data generated question 3 will be considered as your data set hereafter on which you will test the various sparse representation methods.

III Sparse representation with greedy algorithms

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

The previous example illustrates the limitations of spectral analysis from irregularly sampled data with the Fourier transform. 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 consist 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: $\forall k, \boldsymbol{w}_k^\dagger \boldsymbol{w}_k = N$).

Initialization $k = 0$	$\boldsymbol{r}_0 = \boldsymbol{x}$, $\Gamma_0 = \emptyset$, $\boldsymbol{a} = \boldsymbol{0}$.	
Iterations $n = 1 \ldots$	*a)*	Select the atom with index $k = \arg\max_k \lvert \boldsymbol{w}_k^\dagger \boldsymbol{r}_n \rvert$.
		Update the list of indices $\Gamma_n = \Gamma_{n-1} \cup \{k\}$.
	b)	Update the amplitude $a_k = a_k + \frac{1}{\boldsymbol{w}_k^\dagger \boldsymbol{w}_k} \boldsymbol{w}_k^\dagger \boldsymbol{r}_n$
		and the residuals $\boldsymbol{r}_n = \boldsymbol{r}_{n-1} - \frac{1}{\boldsymbol{w}_k^\dagger \boldsymbol{w}_k} \boldsymbol{w}_k^\dagger \boldsymbol{r}_n \cdot \boldsymbol{w}_k$.

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

Various statistical tests can be used to decide when stopping the iterations. Hereafter, we will consider a simple test on the residuals. More precisely, if the signal is correctly approximated, the residual $\boldsymbol{r}$ is composed of centred Gaussian noise with a variance σ^2 which is supposed to be known. In this case, the variable $T = ||\boldsymbol{r}||^2/\sigma_b^2$ has a χ^2 distribution with N

degrees of freedom. In practice, we will use as stopping rule $T < \tau$ with τ such that Proba($T < \tau$) = 95% (τ 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. How many frequencies do you detect with this algorithm? Are these frequencies correctly localized? Are the corresponding amplitudes correctly estimated

3. Analyse the results obtained with this algorithm during the iterations and try to characterize its main drawbacks (order of detection of the frequencies, false detection...).

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

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 iteration 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 of the data on these atoms:

$$\boldsymbol{a}(\Gamma_n) = \underset{\boldsymbol{a}(\Gamma_n)}{\arg\min} ||\boldsymbol{x} - \mathbf{W}_{\Gamma_n}\boldsymbol{a}(\Gamma_n)||^2 = (\mathbf{W}_{\Gamma_n}^\dagger \mathbf{W}_{\Gamma_n})^{-1}\mathbf{W}_{\Gamma_n}^\dagger \boldsymbol{x}$$

where $\boldsymbol{a}(\Gamma_n)$ corresponds to the components of $\boldsymbol{a}$ with indices Γ_n and $\mathbf{W}_{\Gamma_n}$ corresponds to the sub-matrix of $\mathbf{W}$ with the columns of indices Γ_n; and for the residual: $\boldsymbol{r}_n = \boldsymbol{x} - \mathbf{W}_{\Gamma_n}\boldsymbol{a}(\Gamma_n)$.

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

2. How many frequencies do you detect with this algorithm? Are these frequencies correctly localized? Are the corresponding amplitudes correctly estimated

3. 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 $\mathbf{W}$ where orthogonal, which is not the case here. As the aim is to obtain the lowest possible approximation error, it may be good 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 replacing the selection step *a)* of the OMP algorithm with the selection of index k allowing to minimize the approximation error when the column $\boldsymbol{w}_k$ is added to matrix $\mathbf{W}_{\Gamma_n}$:

$$k = \underset{k}{\arg\min} \underset{\boldsymbol{a}(\Gamma_n \cup \{k\})}{\arg\min} ||\boldsymbol{x} - \mathbf{W}_{\Gamma_n \cup \{k\}}\boldsymbol{a}(\Gamma_n \cup \{k\})||^2.$$

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 use for this part the Matlab function `ols` : `[a, ind] = ols(W,x,Inf,test);` with `test` $= \sigma_b^2\tau$.

1. How many frequencies do you detect with this algorithm? Are these frequencies correctly localized? Are the corresponding amplitudes correctly estimated
2. 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* 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 ℓ^1-norm:

$$\boldsymbol{a} = \arg\min_{\boldsymbol{a}} ||\boldsymbol{x} - \mathbf{W}\boldsymbol{a}||^2 + \lambda||\boldsymbol{a}||_1 \text{ avec } ||\boldsymbol{a}||_1 = \sum_k |\boldsymbol{a}_k|$$

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 function `min_l2_l1_0` which can be called from Matlab, after compilation (Matlab command `mex min_L2_L1_0`), with: `a1 = min_L2_L1_0(x,W,lambda,n_it_max);` where `n_it_max` is the maximal number of itérations of the algorithm. With such an algorithm, there is no stopping rule in terms of number of detected frequencies, but a regularization parameter λ to tune. It can be shown that the solution $\boldsymbol{a}_1$ satisfies: $|\mathbf{W}^\dagger(\boldsymbol{x} - \mathbf{W}\boldsymbol{a}_1)| \leq \lambda$ which means that the frequency representation of the residual is lower or equal to λ/N. The parameter λ can then be tuned with respect to the maximal authorized value of the frequency representation of the residual.

1. Test this algorithm for various values of λ until you obtain a satisfactory result (to check if the result is satisfactory you can have a look to the time and frequency representation of the residual as well as the value of the test τ used to stop the iterations of the previous algorithms).
2. 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, can you estimate with a least squares approach 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?