Contents

Part I

Theoretical aspects

Chapter 1

Problem position

1. Objective

This study is devoted to the calibration of an infrasound system, which consists of its sensor and its Wind Noise Reduction System and which is assumed to be a linear time-invariant system, commonly called a linear filter in the literature. It is expected that most results could be applied to the calibration of seismometers, using a reference instrument.

The specifications set out in the Operational Manual [1] provide Minimum Requirements for the calibration of Infrasound Stations: the frequency response shall be within $\pm 5\%$ in amplitude gain from the nominal response in the PTS passband (0.02 Hz to 4 Hz). That represents a challenging issue. It should be noted that no Minimum Requirements are listed for the phase response of Infrasound Stations, yet it is stated in part 4.5 Calibration of the Operational Manual, that: "The calibration minimum requirements for infrasound stations do not currently include phase measurement; however, these measurements are necessary to establish the full system response that is required for data processing at the International Data Centre."

The method to estimate the response of a infrasound system, denoted System Under Test (SUT) in the following, is based on the use of a second sensor, denoted System of Reference (SREF), whose frequency response is perfectly known. The calibration method is based on the presence of a quasi permanent background signal all over the Word [**?**]. Still in [**?**], experimental results have shown that this background signal has a large enough frequency bandwidth and energy to satisfy the PTS requirements.

This background signal induces two signals denoted $e_u(t)$ and $e_r(t)$ in the figure 1.1 A useful property is that a large part of these two signals is

[1] CTBT/WGB/TL-11,17/17/REV.5 Operational Manual for Infrasound Monitoring and the International Exchange of Infrasound Data

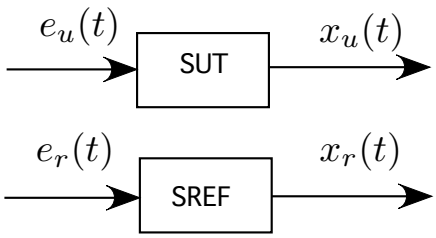

Figure 1.1: *Measurement process. SUT denotes the sensor under test, it consists of a sensor and its noise reduction system. SREF denotes the reference sensor. SUT and SREF are assumed to be linear filters.* $e_u(t)$ *and* $e_u(t)$ *denote the non observable input signals.* $x_u(t)$ *and* $x_r(t)$ *are sampled and recorded.*

simultaneously present on the two sensors. The main reason of the presence of this same signal is related to the spatial coherence which is due to the fact that the signal wavelength is much larger than the distances involved in the measurement process: for a common acoustical velocity of 300 m/s, and the highest frequency of interest saying 4 Hz, the wavelength is 75 m which is much larger than the distance between the two sensors, which is less than 3 meters. [c1]It is worth to notice that the air turbulences, which can be at first order characterized by the ratio v/f where v is its velocity of around a few meters per second, e.g. 4 m/s, induces spatial coherence phenomena for frequencies under $f < 0.02$ Hz, see chapter ... [c1]*momo*:

[c1]In the following the common part, denoted $s(t)$, is shortly called the *coherent signal*. However spatial coherence does not mean that there is a commun part for the inputs of the two sensors. It means that a part of the SUT signal is a filtering of a part of the SREF signal. But if such situation occurs, the filter involved is not identifiable. In the following we assume that this filter is the identity filter. More specifically we let: [c1]*momo*:

$$\begin{cases} e_u(t) &= s(t) + w_u(t) \\ e_r(t) &= s(t) + w_r(t) \end{cases} \tag{1.1}$$

where $w_u(t)$ and $w_r(t)$ are such that the correlation between them are zero for any couple of times. In the following they are shortly called *noise*. They are mainly due to the wind.

Unfortunately the signal $s(t)$ is not *observable*. It follows that, in presence of the additive noises $w_u(t)$ and $w_r(t)$, the problem is ill-conditioned in the sense that they are an infinity of possible solutions.

2. Experimental testbed

In may 2015, an experimental setup has been deployed in the IS26 located in Freyung in Germany:

- The different sensors under test/reference are reported table 1.1. Each reference sensor was installed in the same vault as the sensor under test, and was connected to a newly installed reference inlet through a short pipe. The reference inlet port was positioned at a short distance (less than a few meters) from the main manifold of the Wind Noise Reduction System. These sensors have been checked before installation. We have two kinds of SREF, one is MB2005, the other MB3. The SUT sensor is a MB3. For our calibration problem both SREF sensors can be considered as almost identical.

- The environment is particularly quiet, due to the Bavarian Forest where the station is located. It follows that the wind is very low during large periods of time.

- The data are recorded in continuous at the IDC, in testbed.

It is worth to notice that the calibration problem is a little bit different of the detection problem which consists to provide an alert when the system is out of specifications.

Site	SUT	info.	SREF	info.
	model	serie #	model	serie #
H1/C1	MB3	00020	MB2005	6046
H2/C2	MB3	00017	MB2005	5125
H3/C3	MB3	00014	MB2005	6039
H4/C4	MB3	00023	MB2005	5104
H5/C5	MB3	00012	MB2005	7095
H6/C6	MB3	00011	MB3	00007
H7/C7	MB3	00021	MB3	00008
H8/C8	MB3	00015	MB3	00022

Table 1.1: *sensor specifications: all sensors have the same theoretical sensitivity of* 0.02 *V/Pa. The MB3s have self noise lower than that of the MB2005s. The sites associated to the "Wind Noise Reduction System" are labelled H whereas the sites of reference sensors are labelled C. In the site 1, we have wind informations, as direction, velocity, etc*

3. Main issue

The main issue is related to the under-determination of the problem (also known as blind identification). More specifically in our model we have four unknowns for only two observations. The four unknowns are the frequency response of the SUT, the spectral shape of the coherent signal and the spectral shapes of the two noises. The two observations are the output signals on both sensors, see figure 2.3.

A common way to solve this kind of problem is to introduce some *a priori* knowledges/assumptions. The drawback is what happens when the *a priori* knowledges are not well-verified.

Here a list of *a priori* knowledges that could be considered:

- the signals are stationary in the whole frequency bandwidth of interest. More specifically we have to precise what we mean in terms of stationarity duration. The worst case is for the low frequency, i.e. for long period. For example if the target accuracy requires to integrate the signal over five periods and if we want to analyze the spectral content around the frequency of 0.01 Hz, about 10 minutes of stationarity are needed.

  Let us notice that, even under the stationarity assumption the problem is still under-determined.

- the coherent signal and the noises can be modeled with parametrical models such as auto-regressive paradigm. In [**?**] a generalized autoregressive conditionally heteroskedastic (GARCH) model has been used to model the wind. This approach has not been investigated here.

- the noises on the two sensors are uncorrelated and white. This is the simplest parametrical model which depends on only one parameter. In this case the frequency response is identifiable, but observations and also the physical aspects of the wind noises deny this assumption.

- the system under test is a linear filter with *given* numbers of poles and zeroes (parametrical approach). This approach has not been already investigated. It follows that, for the calibration, implying the numbers of poles and zeroes could hide a singularity and then induce a bias. To illustrate this statement, saying that a system is of order 1 can hide in the estimated response the presence of a non suspected resonance. [c1]But in the alert issue framework, we see that something is wrong, therefore we suspect that it could be an efficient way for test.

  [c1] *Benoit*: A reformuler / preciser.

- assuming that the two noises are uncorrelated, the indetermination disappears if we assume that the ratio between the two noise levels is known. That is a realistic way in high frequency because the NRS works well and the ratio is about the inverse of the number of inlets.

- assuming that the two noises are uncorrelated, the indetermination disappears if we assume that one of the two noises is negligible, typically it could be the case for the SUT which is provided with a noise reduction system (NRS). But we also want to detect if this NRS is working well.

  The advantage of the assumption is that the second noise could be very large, we only have to use the formula (**??**). The drawback is that no test does exist to provide an answer to the question: is the noise on the SUT negligible? Therefore if the assumption becomes wrong the error could be very large.

- assuming that the two noises are uncorrelated, the indetermination can be a fortiori removed if both noises are negligible. The advantage of this second assumption is double: (i) we can use any of the two formulas (**??**) or (**??**), and (ii) we have an efficient way to test if or not the two noises are negligible. The main drawback is it could be difficult to find time windows where the condition is verified for a longperiod, particularly in the high frequency band. A solution is to consider shorter window sizes in the high frequency ranges.

In summary, our conclusion is that, to get the expected accuracy, we have to consider that both noises are negligible. To test that the MSC (see below for specific definition) is used along time windows whose lengths vary in relation with the frequency ranges. From picturing manner, the MSC measures the fact that the estimated values exhibit very low dispersion, hence low noises, along a few consecutive windows.

Chapter 2

Spectral analysis for SUT estimation

In this chapter we derive the expression of the three spectral components associated with the two outputs signals, as a function of the responses of the two sensors. Without *a priori* knowledges the problem is ill-conditioned. An approach based on the MSC can be used to remove the uncertainty. That leads to an estimation algorithm by replacing the spectral components by consistent estimates. These estimates are obtained via Welch's technic. Finally a filter bank approach is shown to be useful. Indeed spectral estimation needs that the signals are stationary for long periods of time, which can be very challenging particularly in the low frequency domain. For circumvent this difficulty we propose to separate the signals in different frequency bands.

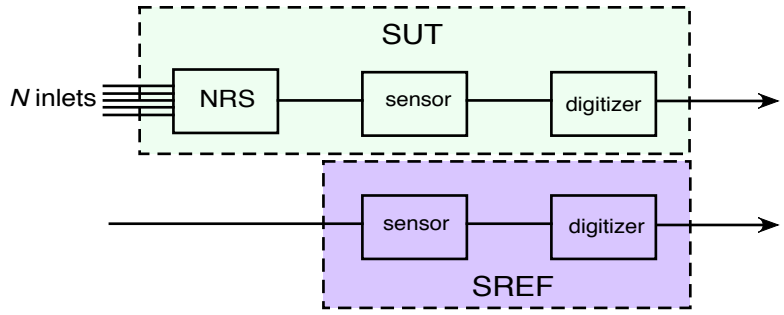

Figure 2.1: *The calibration systems consists of two channels: (i) the SUT with the noise reduction system (NRS), the sensor and the digitizer and (ii) the SREF with the sensor and the digitizer. Because the two digitizers are identical, the response ratio between the two channels does not depend on the digitizer. It follows that the response ratio corrected by the SREF response is an estimate of the SUT response, including the sensor and the NRS.*

Therefore the main quantity of interest is the frequency response ratio between the two channels. It can be estimated by cross-correlation between the two outputs, assuming some hypotheses as stationarity.

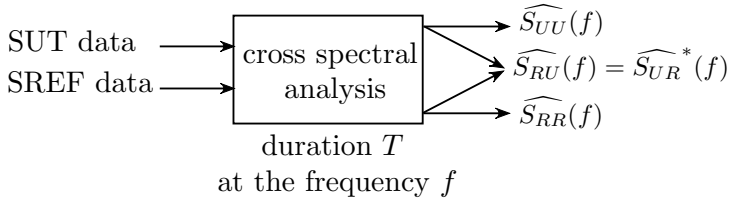

Figure 2.2: $\widehat{S_{UU}}(f)$, $\widehat{S_{RR}}(f)$, $\widehat{S_{UR}}(f)$ *denote respectively the spectrum on channel SUT, the spectrum on channel SREF and the cross-spectrum between them. In the absence of noise the ratio between the 2 responses is given by the ratio* $\widehat{S_{UU}}(f)/\widehat{S_{RR}}(f)$

1. Objective

The calibration consists to estimate the impulse response of the SUT or equivalently its frequency response. In the absence of *a priori* knowledges, i.e. non parametrical approach, the impulse response is defined by a sequence of values whose the number is related to the bounds of the frequency bandwidth. More specifically, in infrasonic system the lowest frequency is 0.02 Hz, corresponding to periodicities around 50 s. On the other hand the largest frequency leads to choose the sampling frequency to 20 Hz. Therefore, in the absence of *a priori* knowledges, the impulse response consists of $50 \times 20 = 1000$ real valued points. Hence the frequency response consists of 500 complex valued points. However these values are obtained via statistical estimators and therefore several sequences are required to reach a good accuracy.

There are basically two ways to estimate a linear filter either in time domain or in frequency domain. In both cases to get a good accuracy it is necessary to do averaging. The stochastic description is well-suited for that. The averaging can be applied using blocks of data or by adaptive approach (as Kalman filter or recursive least square). In the adaptive approach we adjust progressively the parameters of interest. This approach has been investigated leading to no interesting results. The main reason is the large variability of the observation. Indeed this kind of approach needs slow evolution of the stationarity. Numerical studies have shown that the best results are obtained if we take into account a small number of observation blocks and omit all the others.

In summary, in the following we only consider that the averaging is performed by block and we have adopted a frequency analysis. Basic tools for that is the spectral representation of wide sense stationary process (see annex 6).

Remark on the deterministic approach (more details page 81)

Considering only time intervals with almost zero noises, we can ask: why do we use stochastic approach. Indeed the simple ratio of the short time Fourier transforms of the two observations, where the term short time is related to the long period, is the searched response. But there is also an equivalent to the MSC test. In the deterministic approach we have to validate the assumption that there is no noise and that in some way by locking if the performed ratio is almost constant along successive windows. That can be expressed by saying that the two sequences of Fourier along a few windows are correlated, and that leads to the MSC-like test.

2. Observation model

The notations are reported figure 2.3.

2.1 Continuous time model

In the introduction discussion, we have explained that the model of signals are given by expressions (1.1), which are re-written below:

$$\begin{cases} e_u(t) &= s(t) + w_u(t) \\ e_r(t) &= s(t) + w_r(t) \end{cases} \tag{2.1}$$

The common signal $s(t)$ is the key of the estimation. On the other hand the noises induce a loss of identifiability and appear as a nuisance factor which can be characterized by a loss of coherence (LOC).

It follows that the observation signals write:

$$\begin{cases} x_u(t) &= g_u(t) \star (s(t) + w_u(t)) \\ x_r(t) &= g_r(t) \star (s(t) + w_r(t)) \end{cases} \tag{2.2}$$

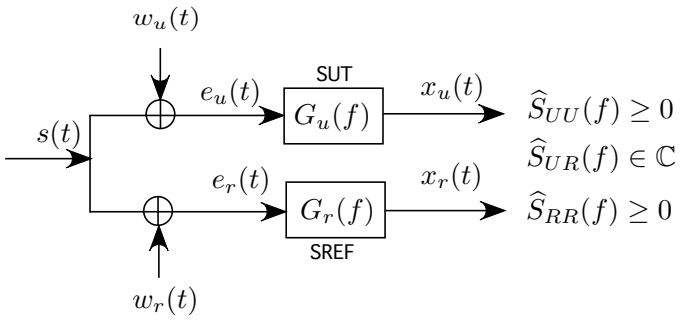

Figure 2.3: *Basic observation schema.* $\widehat{S}_{UU}(f)$, $\widehat{S}_{RR}(f)$ *and* $\widehat{S}_{UR}(f)$ *represent the three estimated spectral components under stationary assumptions*

The stationarity and the absence of correlation between $s(t)$, $w_u(t)$ and $w_r(t)$ play a fundamental role in the identifiability of the response of the SUT. We assume that

- $s(t)$ is a wide-sense stationary process with zero-mean and spectral density $\gamma_s(f)$, which is the Fourier transform of the covariance fonction:

$$R(\tau) = \mathbb{E}\left[s(t+\tau)s(t)\right] \;\rightleftharpoons\; S(f) = \int R(\tau)e^{-2j\pi f\tau}d\tau$$

- $w_u(t)$ and $w_r(t)$ are two stationary processes with zero-mean and respective spectral densities $\gamma_u(f)$ and $\gamma_r(f)$.

- $s(t)$, $w_u(t)$ and $w_r(t)$ are jointly independent. More specifically we have for any couple of time (t, t'):

$$\begin{aligned} \mathbb{E}\left[w_u(t)w_r(t')\right] &= 0 \\ \mathbb{E}\left[w_u(t)s(t')\right] &= 0 \\ \mathbb{E}\left[w_r(t)s(t')\right] &= 0 \end{aligned}$$

- $G_u(f)$ represents the frequency response of the SUT which is the combination of the sensor under test, the frequency response of the anti-noise pipes and the digitalizer. $G_u(f)$ is the parameter to estimate.

- $G_r(f)$ represents the frequency response of the SREF. $G_r(f)$ is known and has been checked just before the installation.

Under these assumptions, the spectral content is fully described by the three components saying the auto-spectrum of the signal of the SUT, denoted $S_{UU}(f)$, the auto-spectrum of the signal of the SREF, denoted $S_{RR}(f)$ and the cross-spectrum, denoted $S_{UR}(f)$. They are related to the input signals and the responses of the sensors by the three following expressions:

$$\left\{ \begin{array}{rcl} S_{UU}(f) &=& |G_u(f)|^2(\gamma_s(f) + \gamma_u(f)) \\ S_{RR}(f) &=& |G_r(f)|^2(\gamma_s(f) + \gamma_r(f)) \\ S_{UR}(f) &=& G_u(f)G_r^*(f)\gamma_s(f) \end{array} \right. \tag{2.3}$$

It is worth to notice that the cross-spectrum does not depend on the noises because the assumptions of non correlation.

2.2 Discrete time model

We consider that the frequency band of interest is $(1/\tau_c, F_M)$. F_M allows to replace the continuous time signals by a time series with a sampling frequency $F_s = 2F_M$. In infrasonic $F_s = 20$ Hz. τ_c implies to estimate the response up to this period. In the infrasonic context $\tau_c = 50$ seconds. That implies to reach in the frequency domain the frequency $1/\tau_c$. Taking a regular grid on the unit circle, as it is with FFT algorithm, the number L of frequency bins will be of an order of magnitude of $\tau_c F_s$.That leads to take L around 1000. However in the following numerical studies we have investigated the response up to 500 seconds, leading to FFT length of 10,000.

In the following the index k refers to the frequency kF_s/L. For example $\gamma_{s,k} = \gamma_s(kF_s/L)$. Because the signal is real, all spectral representations have hermitian symmetry and it is only necessary to consider the positive part of the frequency band. Finally the spectrum at frequency 0 is real valued and will be omitted. Therefore the spectral representation is restricted to the values of the frequency index k going from 1 to $K = L/2$.

From the expressions (2.3) we obtain for the discrete Fourier transform of the spectral components:

$$\left\{\begin{array}{rcl} S_{UU,k} &=& |G_{u,k}|^2(\gamma_{s,k}+\gamma_{u,k}) \geq 0 \\ S_{RR,k} &=& |G_{r,k}|^2(\gamma_{s,k}+\gamma_{r,k}) \geq 0 \\ S_{UR,k} &=& G_{u,k}G^*_{r,k}\gamma_{s,k} \in \mathbb{C} \end{array}\right. \tag{2.4}$$

and from the expression (2.10) the MSC expression in the discrete frequency domain:

$$\mathrm{MSC}_k = \frac{|S_{UR,k}|^2}{S_{UU,k}S_{RR,k}} \tag{2.5}$$

2.3 Resolution of (2.4) w.r.t. $G_{u,k}$

It is worth to notice tat the last equation of the expression (2.4) leads to the following expression of the phase of $G_{u,k}$:

$$\arg G_{u,k} \;=\; \arg G_{r,k} - \arg S_{UR,k} \tag{2.6}$$

For the module of $G_{u,k}$, the problem is a little bit more complicate. Indeed the problem is under-determined in the sense where an infinity of solutions exists.

Case of known noise level ratio

In the particular case where we know the ratio between $\gamma_{u,k}$ and $\gamma_{r,k}$, the equations (2.3) can be solved w.r.t. to $G_{u,k}$. Denoting $\rho_k = \gamma_{r,k}/\gamma_{u,k}$ we easily derive:

$$\begin{aligned} |G_{u,k}| = \frac{1}{2}\sqrt{\frac{S_{UU,k}}{S_{RR,k}}}\,|G_{r,k}| \times \\ \left((1-\rho_k)\sqrt{\mathrm{MSC}_k} + \sqrt{4\rho_k + (1-\rho_k)^2\mathrm{MSC}_k}\right) \end{aligned} \tag{2.7}$$

Typically ρ_k could be considered as equal to the number of inlets in the noise reduction system, e.g. 96.

Under-determination

In the equations (2.3) the response $G_{r,k}$ is perfectly known, the unknows are $G_{u,k} \in \mathbb{C}$, $\gamma_{s,k} \in \mathbb{R}^+$, $\gamma_{u,k} \in \mathbb{R}^+$ $\gamma_{r,k} \in \mathbb{R}^+$.

It follows that, in absence of *a priori* knowledges, the system (2.3) is under-determined with 1 degree of freedom (d.o.f.). That means that for all k we can, for example, choose arbitrarily the ratio $\rho_k = \gamma_{u,k}/\gamma_{r,k}$ from 0 to infinity, then solve the systems (2.3). If one of the two noises is zero, i.e.

the noise ratio is either 0 or infinite, the solution becomes also unique and is given by:

- if $\gamma_{u,k} = 0$, i.e. $\rho_k = 0$

$$G_{u,k} = R_k^{\sup} G_{r,k}, \quad \text{with} \quad R_k^{\sup} = \frac{S_{UU,k}}{S_{UR,k}^*} \tag{2.8}$$

- if $\gamma_{r,k} = 0$, , i.e. $\rho_k = \infty$

$$G_{u,k} = R_k^{\inf} G_{r,k}, \quad \text{with} \quad R_k^{\inf} = \frac{S_{UR,k}}{S_{RR,k}} \tag{2.9}$$

  It is easy to show that $|R_k^{\inf}| \leq |R_k^{\sup}|$, using that the MSC is less than 1.

Unfortunately there is no way to test if one of the two noises is zero. However it is possible to test that the two noises $w_{u,k}$ and $w_{r,k}$ are almost negligible. In this case the two estimators given by (2.8) and (2.9) are almost equal and almost unbiased.

The assumptions of negligible noises can be tested by thresholding the Magnitude Square Coherence (MSC) defined by:

$$\text{MSC}_k = \frac{|S_{UR,}|^2}{S_{UU,k} S_{RR,k}} \tag{2.10}$$

In the model (2.3) MSC also writes:

$$\text{MSC}_k = \frac{1}{(1 + \text{SNR}_{u,k}^{-1})(1 + \text{SNR}_{r,k}^{-1})} \tag{2.11}$$

where $\text{SNR}_{r,k}^{-1} = \gamma_{r,k}/\gamma_{s,k}$ and $\text{SNR}_{u,k}^{-1} = \gamma_{u,k}/\gamma_{s,k}$. It follows that the MSC is 1, iff the two noises are zero. We have to keep in mind:

- that these equations assume stationarity,
- that $G_{u,k}$ is performed during the period of time where the two noises are sufficiently low,
- that we have no access to the true values of the spectral components but only estimates, inducing dispersion.

3. Spectral analysis

The classical moment method leads to replace in the formulas (2.5), (2.6), (2.8) and (2.9) the true spectral components by consistent estimates. In the absence of *a priori* knowledges that is given by a non parametrical spectral analysis. More details about the statistics of these quantities, as the ratio of spectral components or the MSC, are presented in annexe 7.

Based on that, we replace the spectral components $S_{UU,k}$, $S_{RR,k}$ and $S_{UR,k}$ by their consistent estimates $\widehat{S_{UU,k}}$, $\widehat{S_{RR,k}}$ and $\widehat{S_{UR,k}}$, obtained via a Welch's approach by averaging successive periodograms, with a typical overlap of 50% and Hann's window. It is worth to notice that the number of frequency bins is related to the long period signals whereas the accuracy of the estimates are related to the number of windows which are used for averaging the periodograms, typically we have chosen 5 windows.

Also we can find in the annex 7 the details to calculate the probability distributions of $\widehat{\mathrm{MSC}}_k$ and of the ratios $\widehat{S_{UU,k}}/|\widehat{S_{UR,k}}|$ and $|\widehat{S_{UR,k}}|/\widehat{S_{RR,k}}$. We also provide Matlab programs that performs these distributions.

Remark: it is worth to notice that, in any case, the both following inequalities are satisfied:

$$|R_k^{\inf}| \leq |R_k^{\sup}| \tag{2.12}$$

$$|\widehat{R}_k^{\inf}| \leq |\widehat{R}_k^{\sup}| \tag{2.13}$$

but we have no information about the rank of these four values. It means that the true values can be outside of the estimated ones. Moreover the true values can be also outside of the expectation of the two r.v. $\widehat{R}_k^{\inf}$ and $\widehat{R}_k^{\sup}$. However when the noises vanish the discrepancy goes to zero.

3.1 MSC test

It is worth noticing that we have only an estimate of the MSC not the true value. Therefore a test function has been determined to ensure with a given confidence level, typically 95%, to be over a target-value. The target-value is related to the accuracy we want on the estimation of the response of the SUT. In this case an important parameter is the stationarity duration of the signals. For example we see on figure 2.4 that with 5 windows, i.e. about 4 minutes (for long period of 50 seconds), the MSC must be over 0.97 to ensure an RMSE of 5% on the amplitude response. For such requirements the MSC threshold of the test is about 0.99, see figure 2.5.

It is worth to notice that the approach is strongly conservative. Indeed we keep only a few periods of signals, but that could be largely enough if we considered that the calibration has to be done once a year.

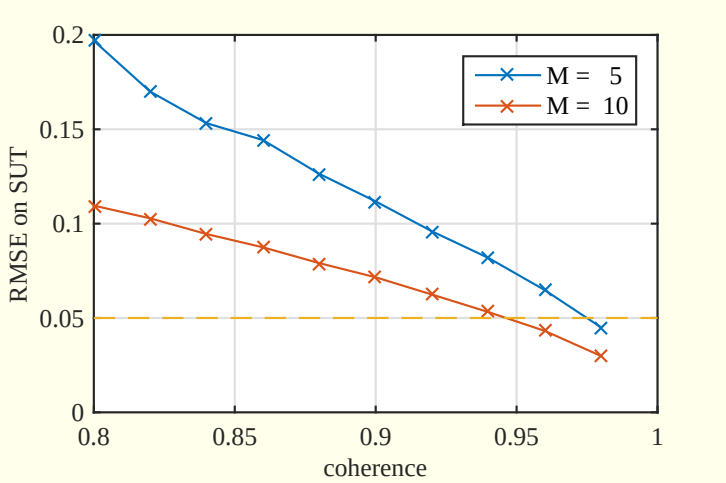

Figure 2.4: *Simulation: for simulating the sensors are two different IIR(2,2) with important resonances are used. The commun coherent signal* $s(t)$ *is white. The length of the frequency analysis is* 300 *seconds, i.e.* 6000 *points. The spectral components are performed on* M *times this duration. The root mean square error is performed by integrating on the full band. These curves have been obtained with the program* `estimHanalysis`.

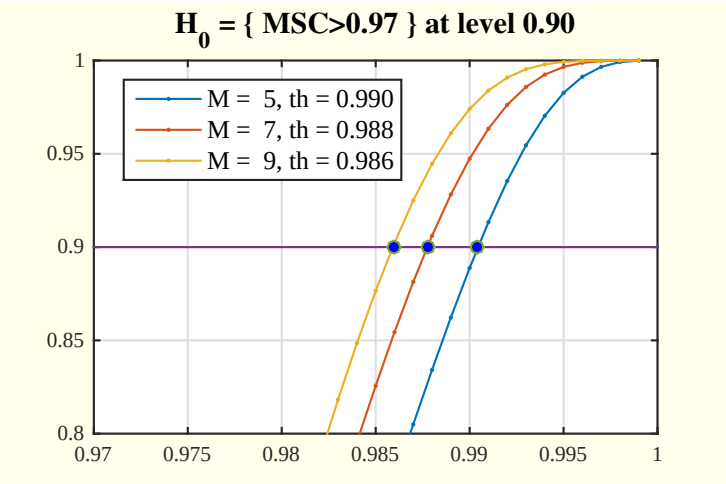

Figure 2.5: *Cumulative function of* $\widehat{\mathrm{MSC}}$ *for different values of the number* M *of averaging under the hypothesis that the MSC is* 0.96. *These curves provide the threshold to test the hypothesis* $H_0 = \{\mathrm{MSC} > 0.96\}$ *with a confidence level of* 90%.

It is important to remark that an RMSE of 5% means that the measurement has only a probability of 0.7 to be in this interval, if we assume gaussian asymptotic behavior. Therefore we could expect that we consider at first an accuracy of 10% and reduce in second step this number by aggregating many measurements. The problem is when the MSC is far from 1 a large indetermination appears which does not leads to a gaussian asymptotic behavior *around the true value*. That is reported on the theoretical curves of the figure 2.6. We see that the two ratios are distributed to a gaussian distribution but at a wrong location. It is not possible to correct this bias because that would assume that the noise ratio is exactly know.

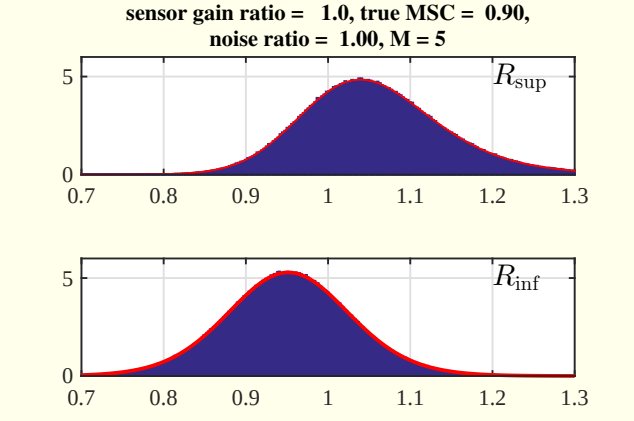

Figure 2.6: *Asymptotic distributions of the two ratios present in the formulas* (??) *and* (??). *These distributions depend on the* MSC, *but also on the noise levels and that is not known in practical case. These curves are obtained with the program* `CIHestimate.m`. *The expressions are proved in the annex (see also the provided toolbox).*

3.2 SUT estimation

In each time window with an MSC above the selected threshold, we can use any of the formulas (2.8) or (2.9) replacing the true values by estimated values. The estimated values are distributed as it is reported figure 2.6. The theoretical expression of the distributions of both ratios are given in annex 7. In first approximation, we can assume for large values of MSC that the bias is zero and only stays the variance.

Therefore if we assume that the different values of the ratios along a large period of time are identically distributed and statistically independent, we can improve the accuracy by averaging. Also using the level of confidence of each ratio, i.e. its variance which is related to the estimated MSC, we can perform a weighted average with weights in the inverse of this variance

following:

$$\widehat{R} \;=\; \frac{1}{L}\sum_{\ell=1}^{L} w_\ell \widehat{R}_\ell \tag{2.14}$$

where L denotes the number of periods of time with MSC above the selected threshold,

$$\widehat{R}^{\sup}_{\ell,k} = \frac{\widehat{S}^{(\ell)}_{UU,k}}{\widehat{S}^{*(\ell)}_{UR,k}} \quad \text{or} \quad \widehat{R}^{\inf}_{\ell,k} = \frac{\widehat{S}^{(\ell)}_{UR,k}}{\widehat{S}^{(\ell)}_{RR,k}} \tag{2.15}$$

and where w_ℓ equals the inverse of the variances, whose approximate values are given by (7.14) and (7.15) and replacing true values by estimated values:

$$\text{for } R^{\sup}: \qquad 1/w_\ell = \frac{1}{2(2M+1)} \frac{\widehat{S}^{(\ell)}_{UU,k}}{\widehat{S}^{(\ell)}_{RR,k}} \frac{1-\widehat{\mathrm{MSC}}_\ell}{\widehat{\mathrm{MSC}}_\ell^{\,2}} \quad \text{and} \tag{2.16}$$

$$\text{for } R^{\inf}: \qquad 1/w_\ell = \frac{1}{2(2M+1)} \frac{\widehat{S}^{(\ell)}_{UU,k}}{\widehat{S}^{(\ell)}_{RR,k}} (1-\widehat{\mathrm{MSC}}_\ell) \tag{2.17}$$

That is implemented in the fonction `fbankanalysis.m`, look at the flag `weightingflag`.

Remark

It is worth to notice that this weighted average is a little bit optimist because based on the assumption that the distribution is identical and the estimates independent. We might be tempted to use this assumption to identify the parameter of interest as it follows: we start from the equation (2.4), labeled by ℓ for different observed periods:

$$\left\{ \begin{array}{rcl} S^{(\ell)}_{UU,k} &=& |G_{u,k}|^2(\gamma_{s,k}+\gamma_{u,k}) \\ S^{(\ell)}_{RR,k} &=& |G_{r,k}|^2(\gamma_{s,k}+\gamma_{r,k}) \\ S^{(\ell)}_{UR,k} &=& G_{u,k}G^*_{r,k}\gamma_{s,k} \end{array} \right. \tag{2.18}$$

Under the assumption that the unlabeled unknown, saying $\gamma_{s,k}$, $\gamma_{u,k}$ and $\gamma_{r,k}$ are identical, we can estimate these values. But it is likely that this approach is not fruitful. Indeed if we look at the distribution of $R^{\sup}_k$ and $R^{\inf}_k$ as reported figure 2.7, it is different from the theoretical distribution given figure 2.6. The main reason is that we mixed many different values of the MSCs.

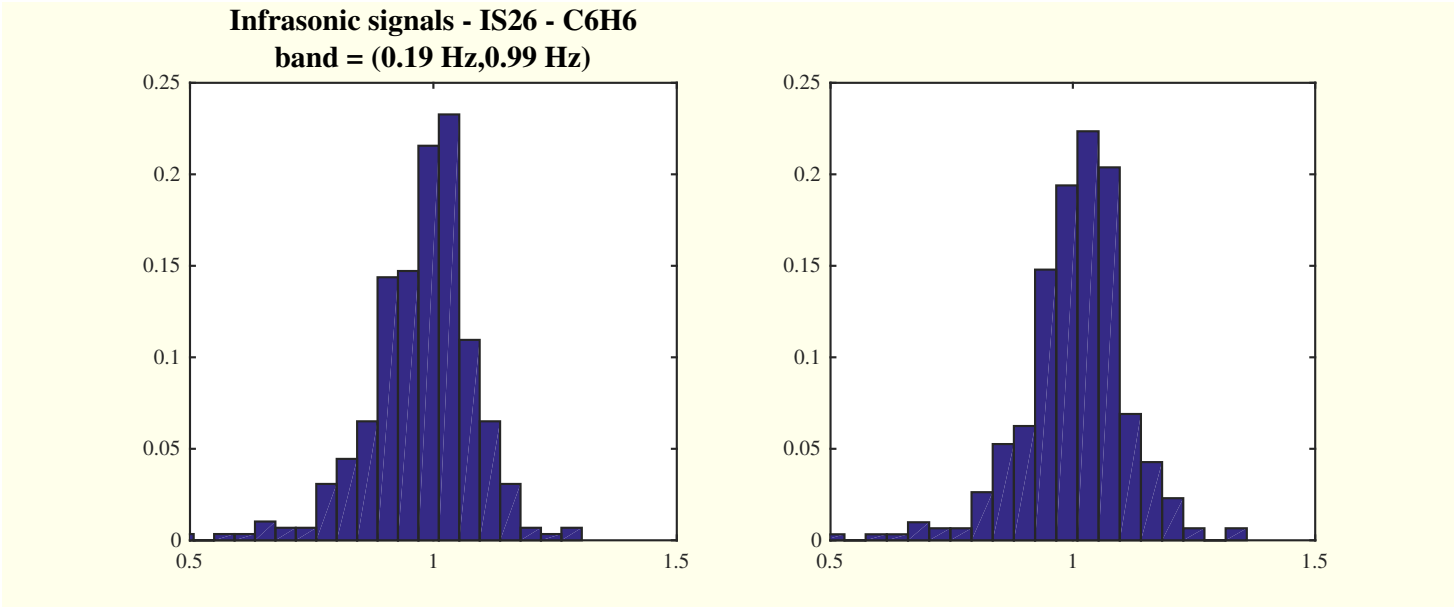

Figure 2.7: *Left:* R_{inf}*, right:* R_{sup}*. The values are obtained in the band of interest of the filter 3. The selected MSC threshold is* 0.95*.*

4. Using a filter bank

We had said that the spectral approach is based on stationarity property. But the real signals do not present permanent stationarity. Therefore we have to use time windows where this stationarity can be verified. Moreover is it maybe possible that in the high frequency range, saying e.g. around 1 Hz, the time window length could be chosen shorter than in the low frequency range, saying e.g. around 0.02 Hz. It is the main reason to propose a filter bank process in the full processing pipeline. The general pipeline proposed for the estimation process consists:

- of a filter bank described in a file of settings which is characterized by a sequence of following descriptors: type, frequency lower bound, upper frequency bound, order, desired stationarity duration etc. Commonly used type is Butterworth (available on Matlab).

- of a spectral estimation process. At the output of a filter, the two signals (one for the SUT the other for the SREF) are shared in non-overlapping blocks to perform the spectral estimates. Longer the size of a block more accurate the spectral analysis. But that assumes stationarity, and the real signals do not exhibit permanent stationarity. Therefore we can choose in the setting file the desired stationarity duration in term of multiple of the longest period in the band (inverse of lower frequency bound). Typically we take 5 times the longest period, expecting that we can find such length with stationarity to estimate

the spectral components. Even with that, the accuracy is not in accordance with the PTS requirements, but by averaging in a long period of times (many days), we can reach such requirements.

- Each block is then shared in overlapping sub-blocks. We can choose the windowing and the overlap rate. Typically we consider Hann windowing and 50% of overlapping. The periodograms in the different sub-blocks are averaged to provide a spectral estimation.

  Only estimates in the bandwidth of the filter are retained.

- MSC is performed in each bandwidth. If the MSC is above the selected threshold, the value of the ratio $\widehat{S}_{UU,k}/\widehat{S}_{RU,k}$ is saved.

- along the recorded data, for a given frequency index k, the values $\widehat{S}_{UU,k}/\widehat{S}_{RU,k}$ are averaged with weighting factors in accordance with $\widehat{\text{MSC}}$ values.

- Finally the estimation of the SUT response is derived from the SREF response.

5. Summary for SUT estimation

- SCP: for spectral components

- DFT for discrete Fourier transform (usually computed by FFT). The DFT consists of as many input points and output points. Periodogram refers to the magnitude square of the DFT.

In the provided function `fbankanalysis.m` (see annex), from the input values `SCPperiod_sec` and `ratioDFT2SCP`, we compute the length of the DFT which is the ratio `SCPperiod_sec`/`ratioDFT2SCP`. Using the `overlapDFT` we derive the number of DFTs is needed for computing the SCP.

For example if `SCPperiod_sec` = 1,000 seconds, `overlapDFT` = 0.5 and `ratioDFT2SCP` = 5, the DFTs are computed on 500/5 = 200 seconds, then for a sampling frequency of 20 Hz, on 4,000 samples. Because the overlap is 0.5 we shift of 100 seconds between 2 successive DFTs. It is worth to notice that the resolution is related to the inverse of the DFT time window length, in our example about 1/200 = 0.005 Hz.

In the provided function `fbankanalysis.m`, we then compute all DFTs for the full input signals. For example if the full duration of signals is $3{,}600 \times 24$ seconds, we have $(3{,}600 \times 24)/100 = 864$ DFTs to compute. Then to compute the SCPs with an overlap `overlapDFT` = 0, we move on RHS of 9 DFTs. if `overlapDFT` = 1/10, we move on 8 DFTs. That means that the possible overlap for the SCPs are only on the block frontier of the DFTs which has no practical effect.

In practice we advise:

- `SCPperiod_sec` depending of the frequency band to analyze (see section 5.). Lower the analysis frequency band, greater the `SCPperiod_sec`. But greater the `SCPperiod_sec` more difficult could be the probability to find a almost stationary time interval.
- `ratioDFT2SCP` $= 5$. It is worth to notice that if we increase `ratioDFT2SCP`, for fixed `SCPperiod_sec` we reduce the DFT time window length and therefore the resolution. We found empirically that `ratioDFT2SCP` $= 5$ is a good value in terms of compromise resolution/accuracy.
- `overlapDFT` $= 0.5$
- `overlapSCP` $= 0$

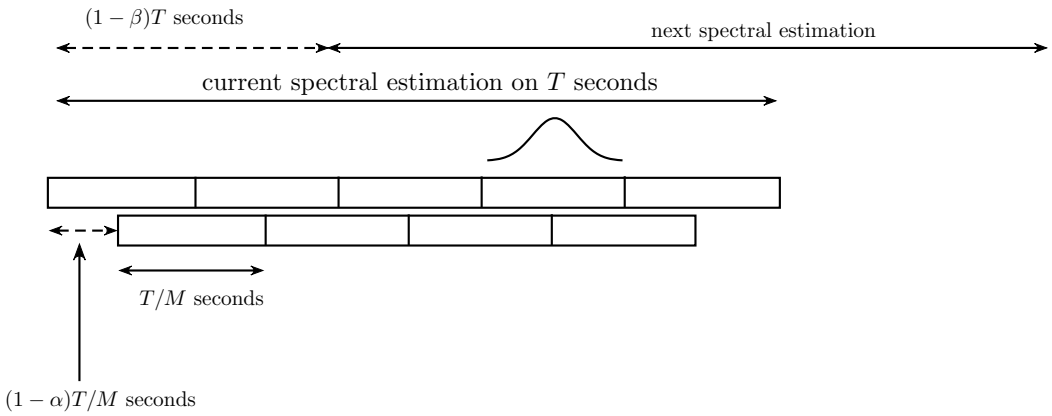

Figure 2.8: *Spectral estimation with* $\alpha = 50\%$ *overlapping for FFT block. The variance of the estimate is related to the number of windows. If the number of disjoint FFT block is* $M = 5$*, and if the length in seconds of the spectral analysis window is 500 seconds, the length in second of the time window for 1 FFT is 100 seconds. The frequency resolution is related to the length of the FFT which is* 100 *seconds, hence* 0.01 *Hz. The window shape is related to the leakage which is defined as the effect of transferring energy from the bands where the energy is high to the bands where the energy is low. The overlapping for successive SCP analysis is* $\beta = 0$*.*

Filter bank analysis

Let us recall that in a first step, the signals are filtered in adjacent bands in such a way to use different periods whose the main interest is to consider short periods, if necessary, in the high frequency bands. The table 2.1 is an example of pavement, consisting of 5 bands with log-spaced filter parameters in the band $(0.01 - 5)$ Hz with a variable window length[c0]. Because the

[c0] See also recent PMCC reports.

frequency band (Hz)	stationarity period (second)
$[0.02 - 0.20]$	400
$[0.20 - 1.00]$	200
$[1.00 - 2.00]$	100
$[2.00 - 3.00]$	50
$[3.00 - 4.00]$	25
$[4.00 - 6.00]$	25

Table 2.1:

two signals are applied to the same filter we can use RII as Butterworth filter. Also because this process is for anlysis, we don't need to downsampling and/or to require perfect recontruction. Even more the bandpass filters can be overlap in the frequency domain. Although a decimation can be used to save computational time, indeed the bandwidths are lower than $F_s/2$, that operation is not considering in the following.

Chapter 3

Results on IS26 data

The results of this section reports the estimation for the eight SUT given in the table 1.1. The analysis is on the data from IS26. Each query on the DB is on 2 consecutive days from the 1st of June to the 9th of October, saying about 130 days. For each file a spectral analysis is processed leading to an SUT response estimate. From the spectral components we perform, in each frequency band and for each time interval, the response of the system under test (SUT) using the known response of the reference sensor. These responses includes both the sensor and the digitizer. These responses are averaged over the full set of time intervals over the all days. The MSC threshold is fixed to 0.98.

Let us remain the PTS specifications:

- IMS pass-band requirement $[0.02 - 4]$ Hz;
- $\pm 5\%$ on the response magnitude, i.e. ± 0.43 in dB scale;
- the calibration is required at least once a year;
- no requirement on the phase but ... $\pm 5^{\circ}$ as for seismic requirements.

1. Results of IS26 - sensor #1

In this section we only considered the sensor #1, but identical results have be obtained on aa seven other sensors.

The average estimate of the SUT response, in terms of gain in dB and phase in degree, are reported figure 3.1.The red dashed lines report the theoretical SUT response range requirements, which is ± 0.43 for the gain and $\pm 5°$ for the phase. They have be performed taking into account the SREF sensor response and using the model of NRS developped by the IMS.

1.1 SUT response by averaging on 124 days

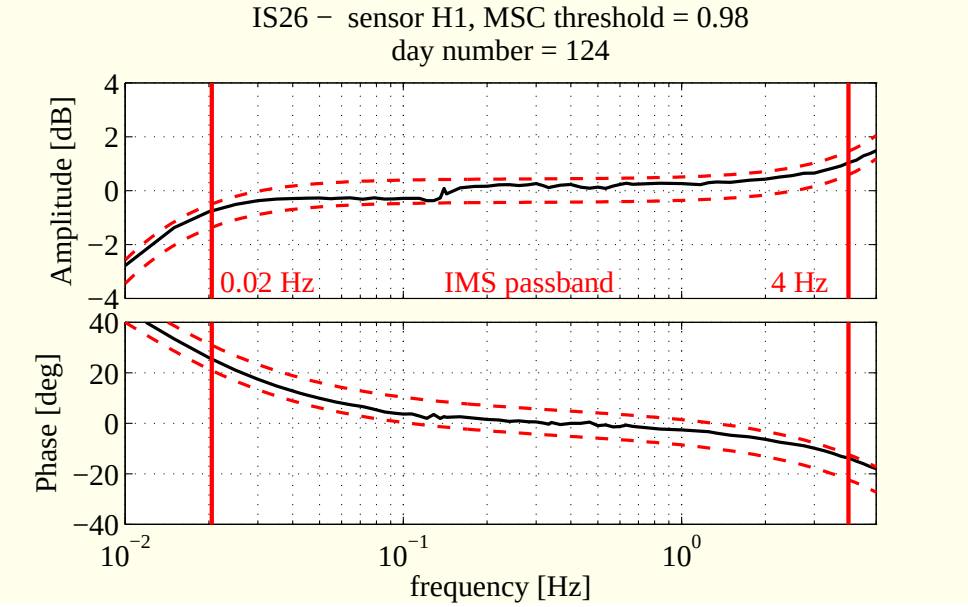

Figure 3.1: *The red dashed lines report the theoretical SUT response range requirement, which is* ± 0.43 *for the gain and* $\pm 5°$ *for the phase.*

1.2 Problem with some records

In this section we report a few issues we have observed in the recorded signals of the database `tesbed_archive` and that can be considered as "true" outliers. In figure 3.2 we have considered the same data than those of figure 3.1, adding two days more. These two days induce a burst in the performed SUT response. To identify the origin, we have plotted the signals of these two days figure 3.3. We clearly observed huge bursts in two samples locations. If we suppress these 2 bursts and re-introduce the two cleaned days to perform the SUT response, we obtain the curves reported 3.4, which are very similar to the curves figure 3.1. We may conclude than unusual values in the observed signals (outliers) lead to false measurement of the SUT response. Therefore it is very important to identify these outliers before any calibration.

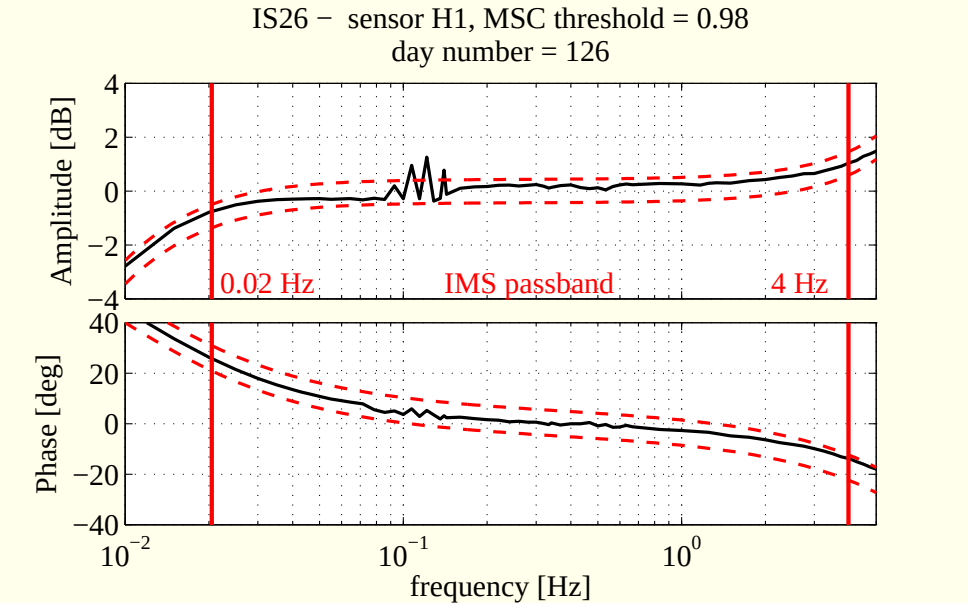

Figure 3.2: *Same data than those of figure 3.1 but adding two days more. We observe that a few values seem to be outliers. The data of these two days are reported figure 3.3. We see large bursts that are at the origin of the ouliers.*

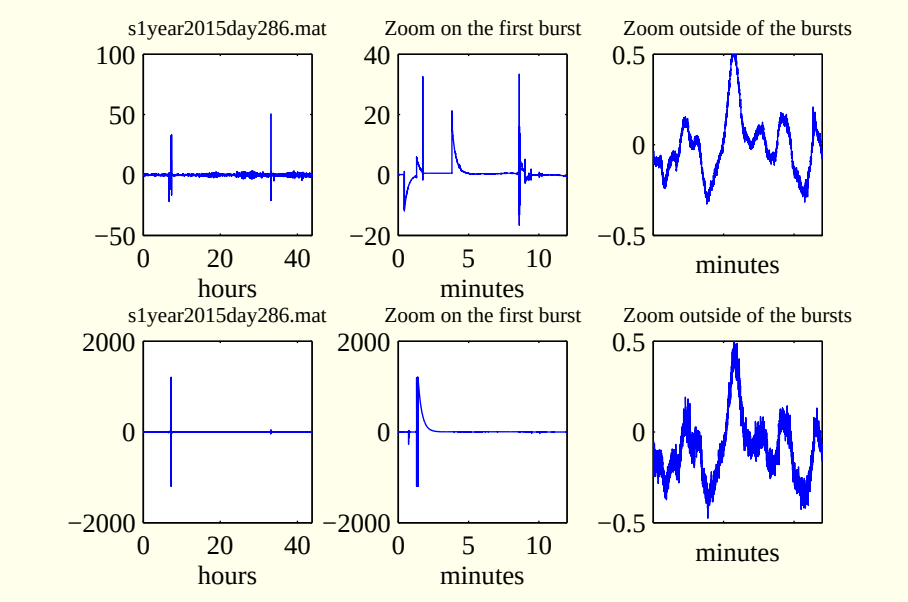

Figure 3.3: *Two days data added in the protocol to obtain the figure 3.1. We observe large bursts that can be considered as outliers, as they appear hugely different of the observed values outside of the bursts.*

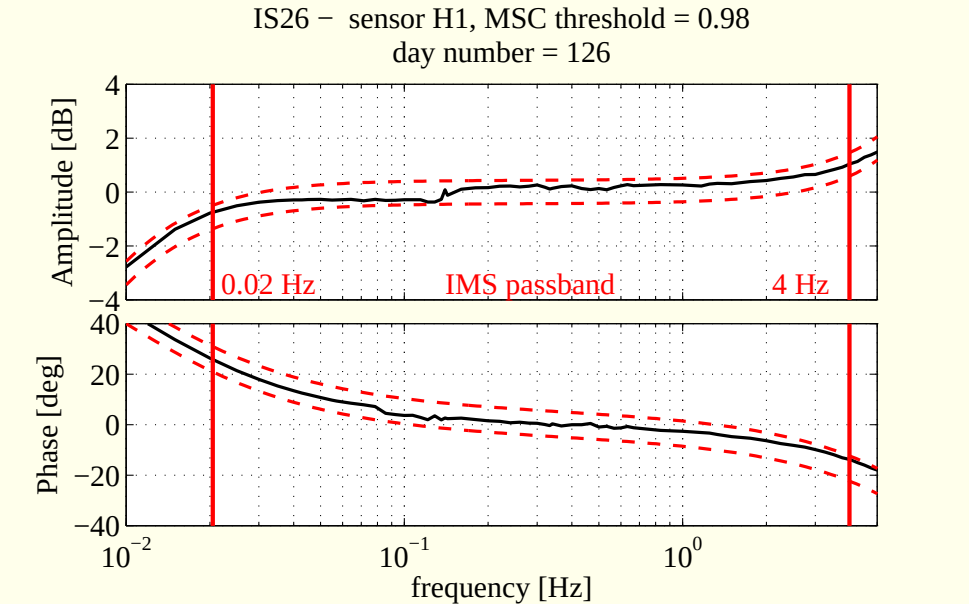

Figure 3.4: *Same data than those of figure 3.2 but suppressing the part of the added signals where bursts are located. The figure is very similar to the figure 3.1.*

1.3 SUT response at 1 Hz for all pairs of days

1.4 About 48 hours with different thresholds

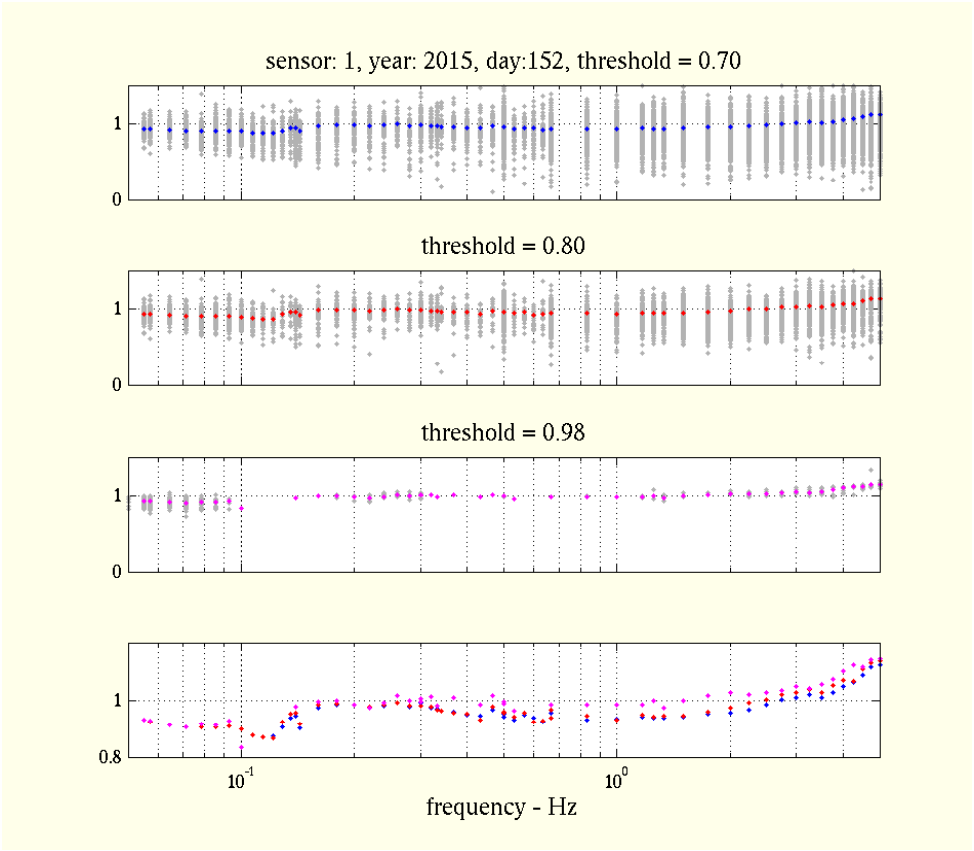

Figure 3.5: *Sup ratio in dB given by the expression* (2.8) *for 2 successive days into the* $P = 6$ *frequency bands for different values of the MSC threshold. The gray dots report the values associated to the MSC greater than the threshold. The color dots report the weighted averagings on the gray values. We see larger dispersion lower the MSC threshold. Also we see differences between the associated averagings, meanind that a bias is present. Theoretically (under regular assumptions) the bias is larger for lower MSC threshold.*

1.5 Full band on about 48 hours

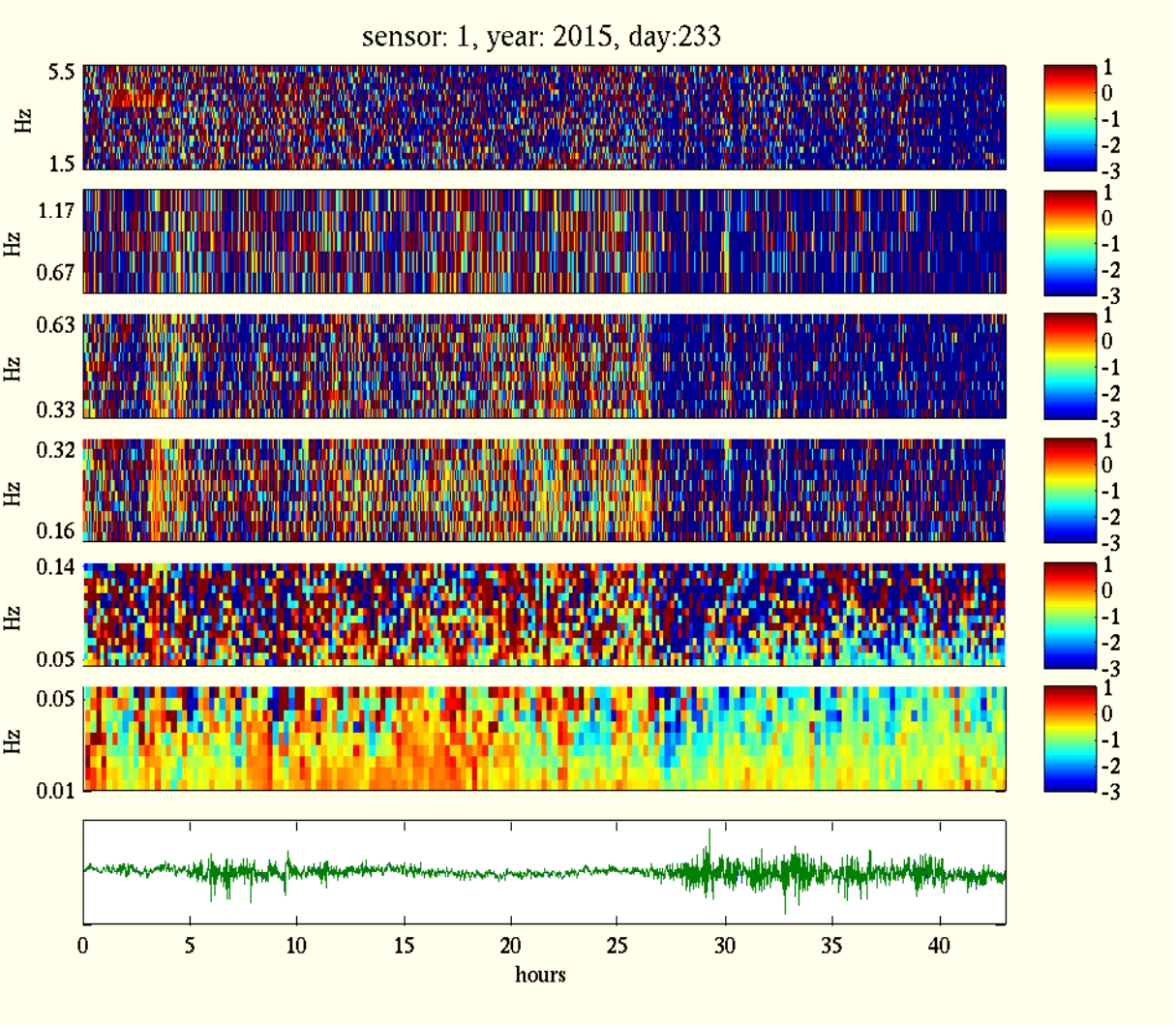

Figure 3.6: *Sup ratio in dB given by the expression* (2.8) *for 2 successive days into the* $P = 6$ *frequency bands.*

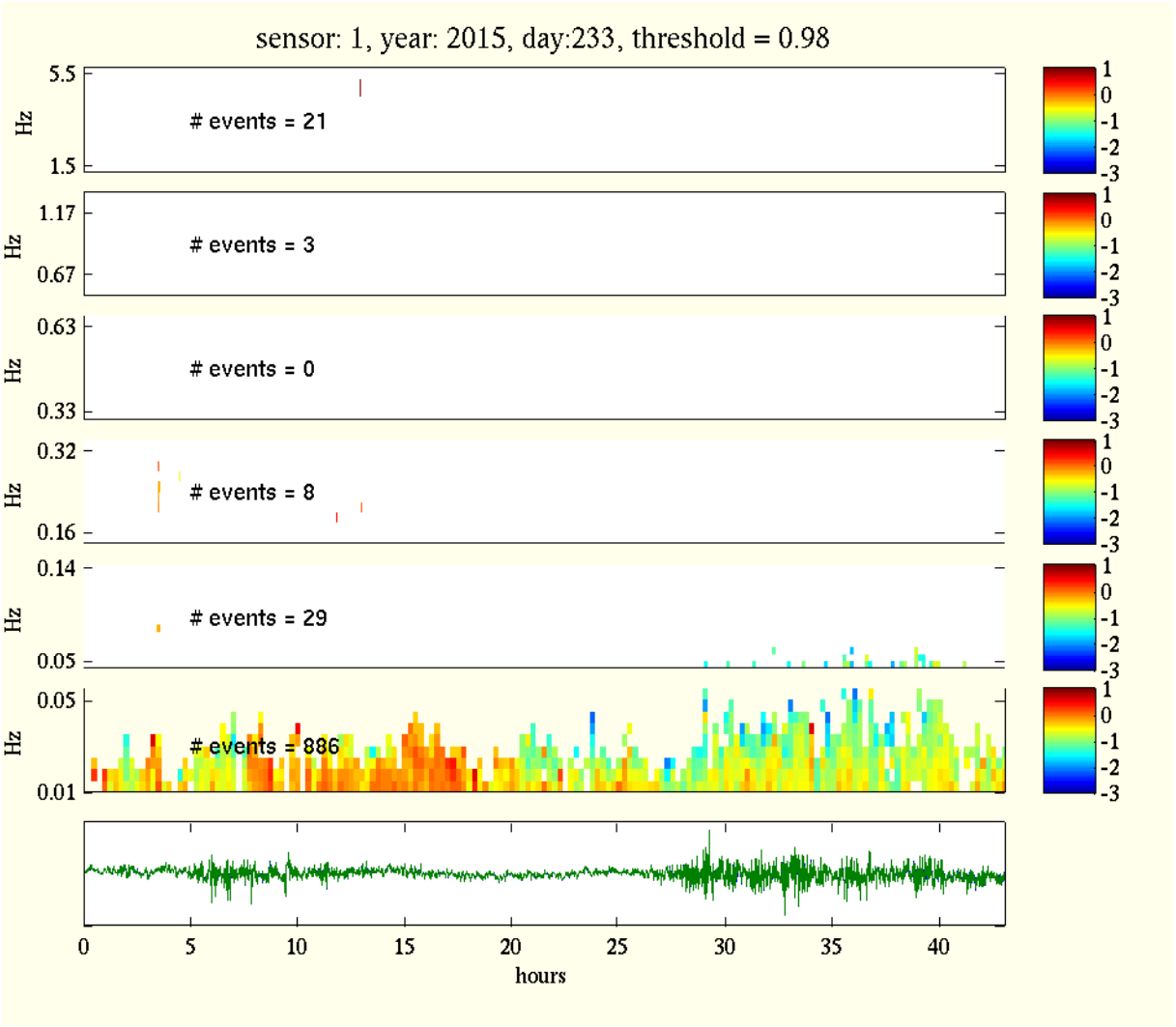

Figure 3.7: *Sup ratio in dB given by the expression* (2.8) *for 2 successive days into the* $P = 6$ *frequency bands for MSC above the threshold. We observe a very few events in the frequency mid-ranges, in this case we have to use a very long period of time to reduce the dispersion.*

1.6 Full band for 20 randomly selected days

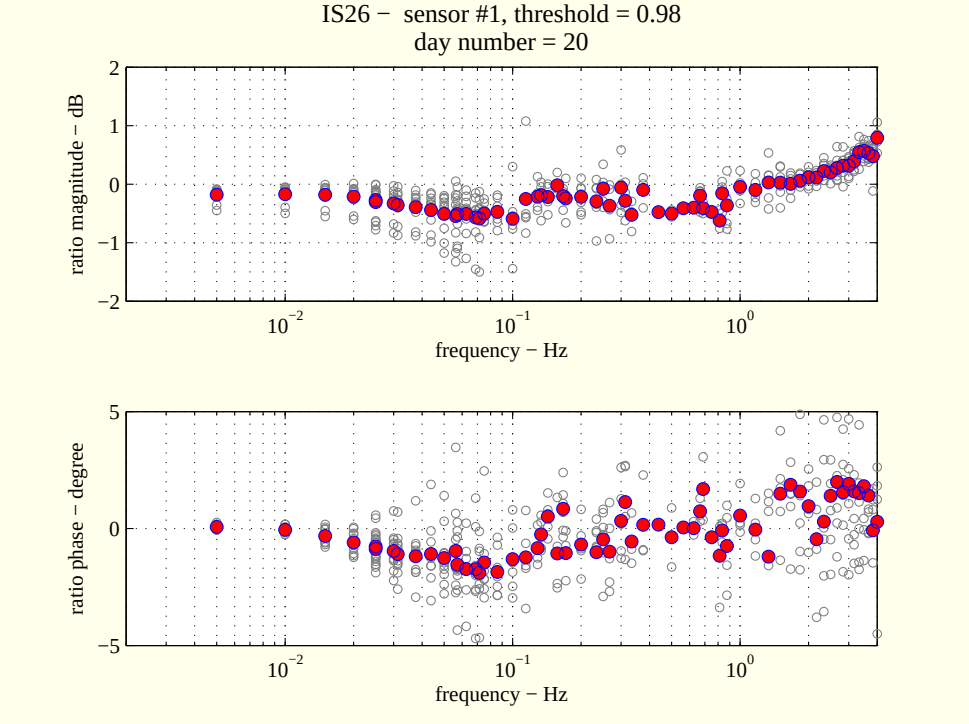

Figure 3.8: *Sup ratio given by the expression* (2.8). *The gray dots report the ratios averaged on a pair of 2 consecutive days, the red disks report the average on 10 pairs of days.*

1.7 Stability at 1 Hz

2. Comments

2.1 About the selected MSC threshold

We said that we can not choose a too low value for the MSC threshold because the underdetermination. That appears clearly on figure 3.9. When the MSC threshold is 0.7, there is a large discrepancy between the estimated ratios, see (2.15) and we outline that there is no way to solve the underdetermination. But for an MSC threshold of 0.95, the two curves are very close.

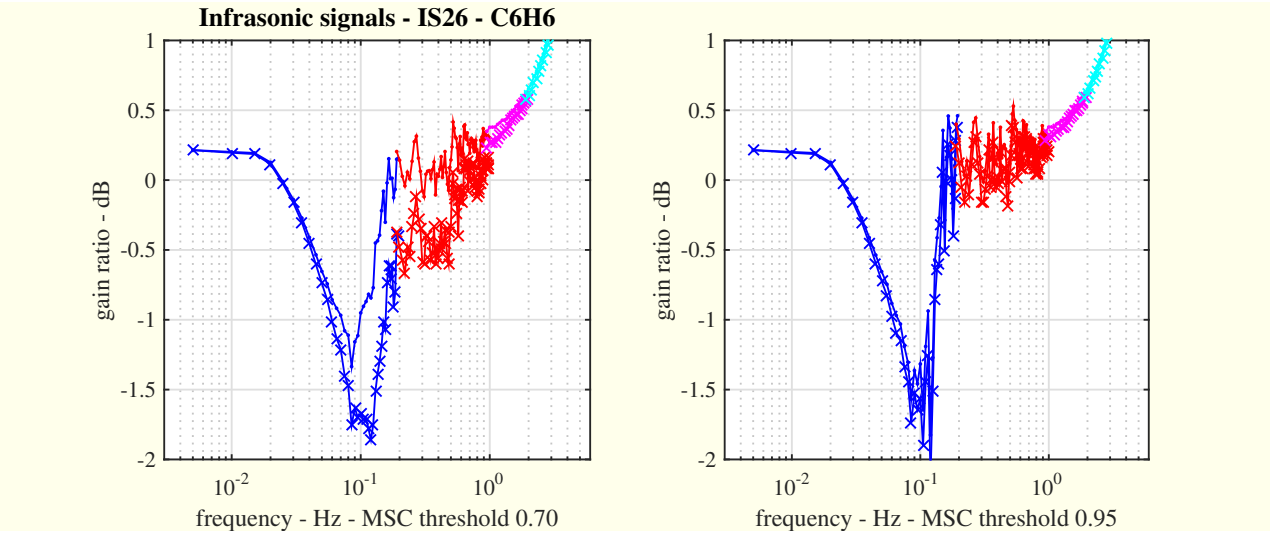

Figure 3.9: *Couple H6C6, for two MSC thresholds.*

On the other hand if the ratio between the two noise levels are perfectly known the indetermination is removed and we can use formula (2.7). You might be tempted to say that the two noises on the two sensors are identical except theirs levels and consider that the ratio is given by the number of inlets in the noise reduction system. However the figure 3.10 shows that the curve with a MSC threshold of 0.5 and the formula (2.7) lead to values different from the curves obtained with a a MSC threshold of 0.95.

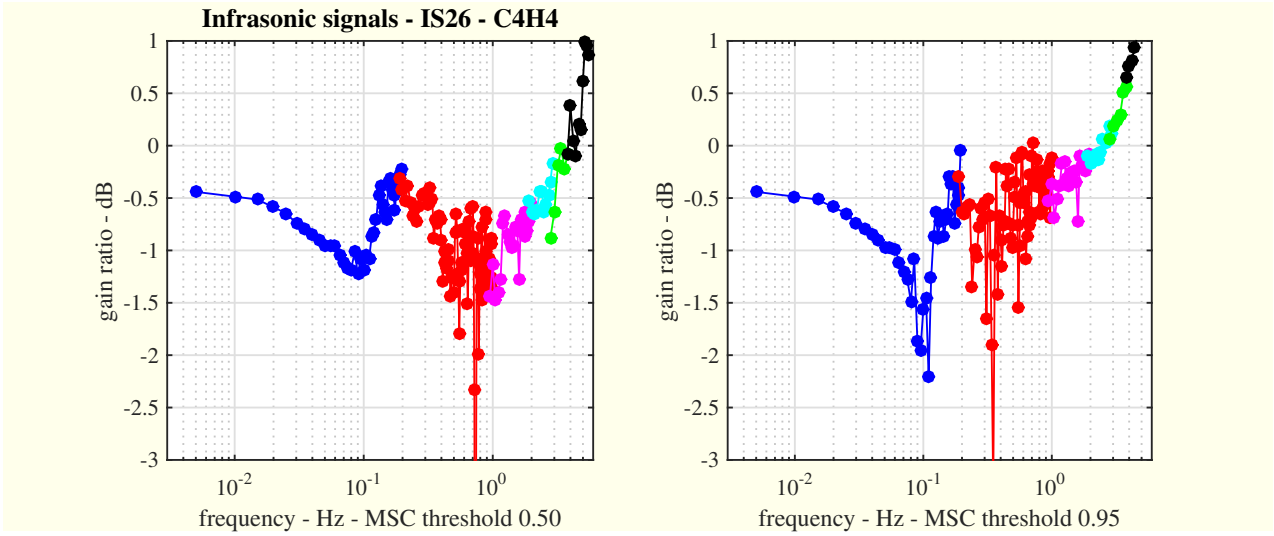

Figure 3.10: *Couple H4C4, for two MSC thresholds with formula* (2.7).

2.2 Dip on the curves

For the figures **??** to **??** associated to the MB2005, an important dip around 0.1 Hz is observed.

Also we have reported figure 3.11 a few days on the location H2C2. The different colors are for different days. The distribution seems to be uniform and does not depend on the day. The ratio seems to be different of 1.

If the MSC is about 0.99, meaning that the noises on the two sensors are negligible, and if the two sensors have the same response, the only reason to get a ratio less than 1, is that the acoustical SOI on the SUT is attenuated, may due to the noise system reduction. That would write: it exists $\alpha \in \mathbb{C}$ with $|\alpha| < 1$ s.t.:

$$\begin{cases} x_u(t) &= g_u \star (\alpha s(t)) \\ x_r(t) &= g_r \star s(t) \end{cases} \tag{3.1}$$

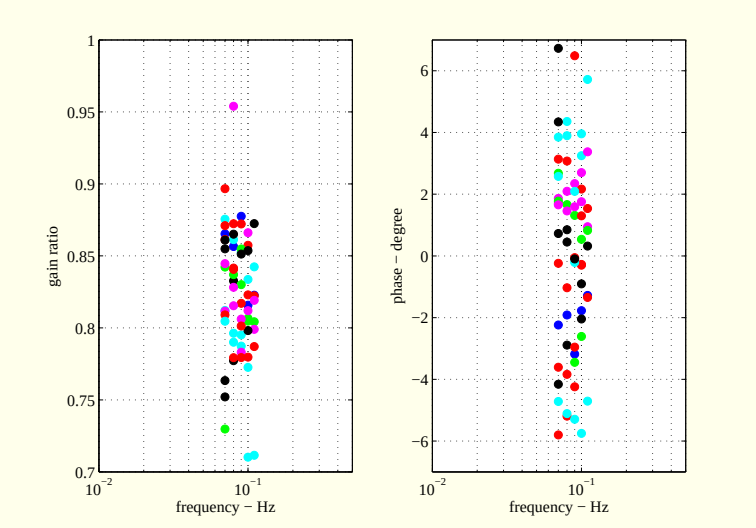

Figure 3.11: *a few days on H2C2. Only the band* $[0.08 - 0.12]$ *Hz is selected. The coherence is above* 0.99*. The different colors for the different days.*

Another way to see the response differences between the 2 sensors in the band $[0.08 - 0.12]$ Hz (gain ratio different of 1), appears figure 3.12. A zoom of the signals is plotted. It consists of about 1 minute, around a position where the observed coherence is above 0.99. We see that the signal on the SREF is bigger than this on the SUT. There is no way and no reason to reject this time window since the difference can be due either to the loss of gain or a unknown transfer function.

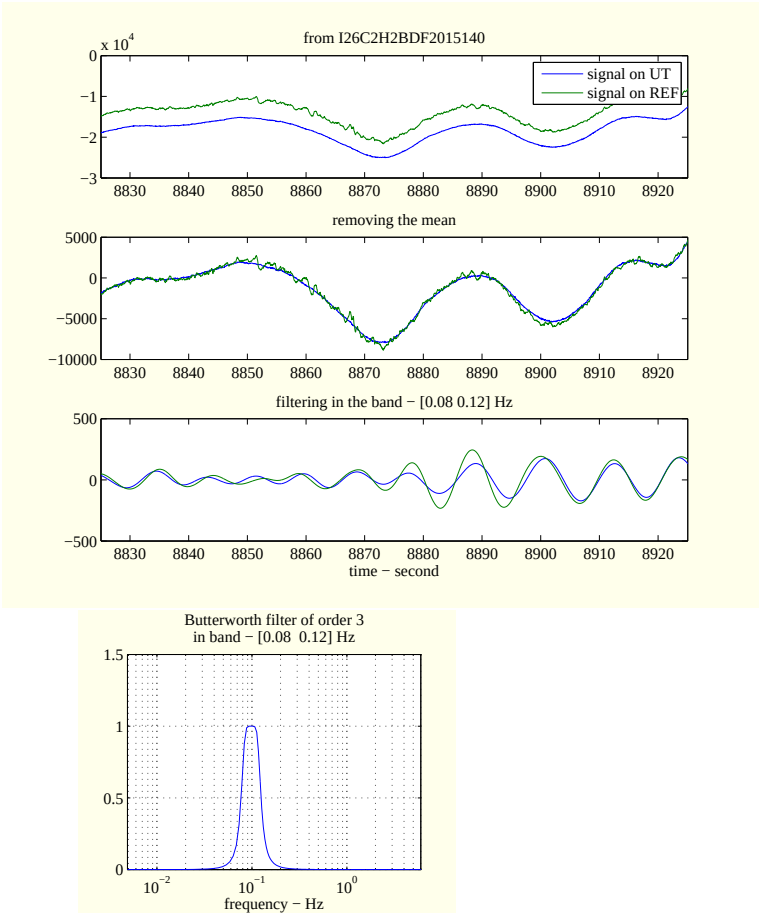

Figure 3.12: Filtered signals

Chapter 4

Wind effect on NRS

1. Introduction

The noise reduction system (NSR) commonly used in front of the infrasonic sensors in the IMS stations is based on the property that the wind noise appears as spatially uncorrelated at the air inlets of the NRS. Therefore by summation, the SNR is improved in the ratio of the number of inlets.

Unfortunately this gain is reduced in low frequencies as it has been observed by Alcoverro and al. [**?**]. This effect is characterized by the parameter $\zeta = v/f$ where v denotes the wind velocity and f the frequency. ζ can be interpreted as a "wavelength" and must be compared to the inlet inter-distances. If ζ is of the same order of magnitude of the inter distances, spatial coherence increases. It is not a good news, at first because the noise reduction is lower. Another drawback occurs when we want to calibrate the sensor by using a second sensor without NRS. In this case the levels of the coherent part will be different and we have no way to identify it.

As a quantitative illustration for wind velocity above 5 m/s, measurement is no more possible. But for velocity around 1 m/s and for frequency under 0.01 Hz, $\zeta > 100$ m and the wind is seen as coherent by the full NSR. In the middle range for velocity around 1 m/s and frequency around 0.1 Hz, $\zeta \approx 10$ m and a part of the wind noise is seen as coherent just because the NRS. That can induce artefact in the *in-situ* calibration because the coherence level close to 1 does not guaranty that the system to solve is well determined.

This article proposes a model for the spatial coherency, as it observed in [**?**] and presents simulated and experimental results. This model is similar to the model proposed in [**?**] which is derived from the pioneer approach of Mack and Flinn [**?**].

2. Model for wind turbulences

Wind turbulence is a very complex air movement. Here we assume that it can be described (as a gray box) by a stationary random process whose coherence matrix depends on the distance between two spatial locations. At first we just summarize useful definitions of stationary M-ary process.

2.1 Stationary M-ary process

An M-ary process $x(t) = \begin{bmatrix} x_1(t) & \dots & x_m(t) & \dots & x_M(t) \end{bmatrix}^T$ is associated to a spatial-time description. More specifically, t denotes the time and m an index associated with the 3D spatial locations, usually related to a sensor array. Therefore we have to consider spatial and temporal properties. For example, an M-ary process could be:

- temporally white, meaning that $\mathbb{E}\left[x_m(t)x_m(t')\right] = 0$ for $t \neq t'$,
- spatially white, meaning that $\mathbb{E}\left[x_m(t)x_{m'}(t)\right] = 0$ for $m \neq m'$.

There is no relationship between these two notions. In the general case $\mathbb{E}\left[x_m(t)x_{m'}(t')\right]$ does depend on (m, m', t, t'). Temporal stationarity refers to the property that the statistics do depend on $(t - t')$. Spatial "stationarity" is a littlebit more complicate because of the 3D coordinates. In some simple cases $\mathbb{E}\left[x_m(t)x_{m'}(t')\right]$ could depend only on the distance between the 2 points indexed by m, m'. A particular case is when $\mathbb{E}\left[x_m(t)x_{m'}(t')\right] = 0$ for $m \neq m'$ which is called spatially white.

Spectral description

When the process is a wide sense stationary (WSS) process a spectral representation can be associated to the covariances. A zero-mean WSS process is defined by the property that the covariance matrix $R(h) = \mathbb{E}\left[x(t+h)x^H(t)\right]$ does not depend on t. It is also characterized by the spectral matrix which is the Fourier transform of the covariance matrix sequence:

$$\Gamma(f) = \int e^{-2j\pi fh} R(h) dh \tag{4.1}$$

where $\Gamma(f)$ is a square matrix of size M.

The most fundamental property says that, for any value of the frequency f, the spectral matrix is a non-negative matrix. Moreover if $x(t)$ is real, $\Gamma(f) = \Gamma(-f)$.

If the spectral matrix is of rank 1, the process is said spatially coherent. If the spectral matrix is diagonal we say that the process is spatially white. If the spectral matrix is rank deficient we say that the process is partially-coherent. A process whose spectral matrix $\Gamma(f) = \sigma^2 I_M$ is both spatially and temporally white, because $\Gamma(f)$ does not depend on f.

It is also common to consider the function

$$C_{km}(f) \quad = \quad \frac{\Gamma_{km}(f)}{\sqrt{\Gamma_{kk}(f)\Gamma_{mm}(f)}} \tag{4.2}$$

where it is assumed that $\Gamma_{kk}(f)\Gamma_{mm}(f) \neq 0$. We can also use the matricial notation

$$C(f) = P^{-1}(f)\Gamma(f)P^{-1}(f) \tag{4.3}$$

or

$$\Gamma(f) = P(f)C(f)P(f) \tag{4.4}$$

where

$$P(f) = \begin{bmatrix} \Gamma_{11}^{1/2}(f) & 0 & \cdots & 0 \\ 0 & \Gamma_{22}^{1/2}(f) & \cdots & 0 \\ \vdots & & & \\ 0 & \cdots & 0 & \Gamma_{MM}^{1/2}(f) \end{bmatrix} \tag{4.5}$$

The M-ary matrix

$$C(f) = \begin{bmatrix} 1 & C_{12}(f) & \cdots & \\ C_{21}(f) & 1 & \cdots & \\ \vdots & & & \\ \cdots & \cdots & C_{M,M-1}(f) & 1 \end{bmatrix} \tag{4.6}$$

is called the coherence matrix. The function

$$\mathrm{MSC}_{km}(f) = |C_{km}(f)|^2 \tag{4.7}$$

is called the magnitude square coherence, in short MSC, between $x_k(t)$ and $x_m(t)$. It is easy to show that $\mathrm{MSC}_{km}(f) \leq 1$ and

$$\mathrm{MSC}_{km}(f) = 1, \quad \forall f \tag{4.8}$$

if and only if $x_k(t)$ and $x_m(t)$ are related by a linear filter. When $\mathrm{MSC}_{km}(f) = 1$ for all pairs (k, m) the spectral matrix is of rank 1, hence

$$C(f) = \mathbb{1}_M \mathbb{1}_M^T \tag{4.9}$$

and the process is fully coherent.

WSS filtering

A fundamental property concerns the filtering. Let us consider an M-ary zero-mean WSS process $x(t)$ with the spectral matrix $\Gamma_x(f)$. Let $y(t)$ the N-ary process on the output of a multiple input multiple output (MIMO) filter whose frequency response is the matrix $K(f)$ of size N by M. Then $y(t)$ is a WSS process with zero-mean and whose the spectral matrix writes:

$$\Gamma_y(f) \;\; = \;\; K(f)\Gamma_x(f)K^H(f) \tag{4.10}$$

2.2 Wind noise model

Wind turbulence is a very complex air movement. Here we assume that it can be described by a stationary random process whose coherence matrix as

the following entry form [**?**]:

$$C_{km}(f) = \int \exp(-2j\pi f(r_k - r_m)^T k)\, p_K(k) dk \tag{4.11}$$

where f is the frequency, r_k and r_m are any two 3D locations and k the slownwess vector. $p_K(k)$ is a probability distribution in $\mathbb{R}^3$ meaning that the slownwess vector is considered to be random. Let assume at first that k is deterministic, that writes $p_\Delta(u)du = \delta_0(du)$. Hence expression (4.11) can be written:

$$C_{km}(f) = e^{-2j\pi f(r_k - r_m)^T k_0}$$

That means that the displacement is a plane wave. It is worth to notice that the matrix C can be written:

$$C(f) = E(f)E^H(f), \quad \text{with} \quad E(f) = \begin{bmatrix} e^{-2j\pi r_1^T k_0} & \dots & e^{-2j\pi r_M^T k_0} \end{bmatrix}^T$$

Hence the multivariate process is fully coherent.

The model, derived from the pioneer work by Mack and Flinn [**?**] for the loss of coherence, considers that the azimuth a, the elevation e and velocity v are random variables. More specifically, we assume that the 3D vector $\mu = (a, e, v)$ writes:

$$\mu = \mu_0 + \nu \tag{4.12}$$

where $\mu_0 = (a_0, e_0, v_0)$ is a deterministic part and ν a zero-mean Gaussian random vector of dimension 3, whose covariance is denoted Σ_μ. It is shown in the appendix that:

$$C_{m,\ell}(f) = \underbrace{e^{-2j\pi f(r_m - r_\ell)^T k_0}}_{\text{pure delay}} \underbrace{e^{-2\pi^2 (f/v_0)^2 (r_m - r_\ell)^T \tilde{\Sigma}_k (r_m - r_\ell)}}_{\text{coherence effect}} \tag{4.13}$$

where $\tilde{\Sigma}_k$ is given by (8.3). We let $\zeta = v_0/f$ that appears as a "wavelength". Using the MSC definition (4.7) we derive the following expression:

$$\mathrm{MSC}_{km}(f) = e^{-4\pi^2 (r_m - r_\ell)^T \tilde{\Sigma}_k (r_m - r_\ell)/\zeta^2} \tag{4.14}$$

The same parameter ζ has been proposed by Alcoverro and al. in [**?**] to characterize the coherence of the wind disturbances for closely located sensors. The expression (8.5) has a clear meaning: the coherence between any two spatial locations decreases when the distance between them is large compared to the "wavelength" ζ.

It follows that when ζ is low, i.e. low wind velocity and/or high frequency, the wind signals on the different air inlets are almost uncorrelated hence M works as a reduction factor. On the other hand when ζ is high, i.e. high wind velocity and/or low frequency, the wind signals on the different air inlets are strongly correlated hence that allows good conditions to estimate the ratio between the sensor responses. In the mid-range, it appears that the common signal is present with two different amplitudes on the two sensors inducing a

ratio different from 1. Simulation below gives an illustration of that.

2.3 Coherence level as a function of v/d

To show that our model is in good agreement with the experimental results reported in [**?**] we consider the following simulation: we choose M random locations in such a way we obtain several different inter-distances. Using the formula (8.4) we compute for each pair of locations the frequency f_c associated with the coherence level of 0.5. This value has been chosen in [**?**] to define a cut-off for the MSC. We also compute the ratio v/d where d is the distance between the two locations. Finally the coordinate couples $(v/d, f_c)$ are reported figure 4.1. The obtained cloud shape figure is very similar to the observation values reported in figure 5 of [**?**]. It is worth to notice that the frequency f_c does not depend only on the distance. The reason is the coherence formula depends on the quadratic form $(r_m - r_k)^T \tilde{\Sigma}_k (r_m - r_k)$ and therefore on the angle between $(r_m - r_k)$ and the eigenvectors of the matrix $\tilde{\Sigma}_k$. This quadratic form reduces to the euclidian distance if $\tilde{\Sigma}_k \propto I_3$.

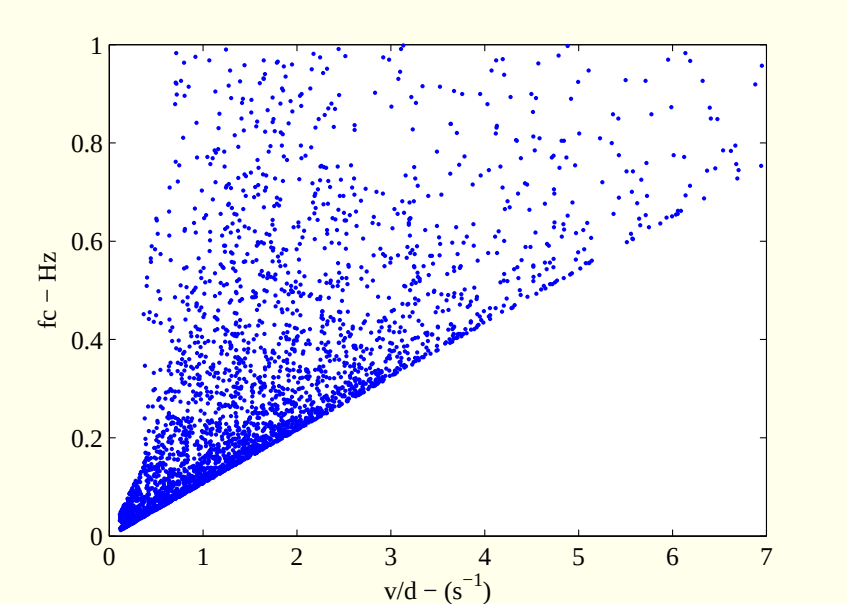

Figure 4.1: *Frequency f_c associated with the MSC level* 0.5 *as a function of v/d.*

3. Application to the *in-situ* calibration method

We consider a sensor with its NRS. This sensor is called sensor under test (SUT). We also consider an other sensor with no NRS, representing the reference sensor (Sref). The NRS consists of $M = 96$ air identical inlets located at the ends of pipes. The scheme is reported on the figure 4.2. Thanks to the Gabrielson's study [**?**], the transfer function $U(f)$ between any inlet and the cavity center of the NRS can be performed as a function of the geometrical structure of the NRS.

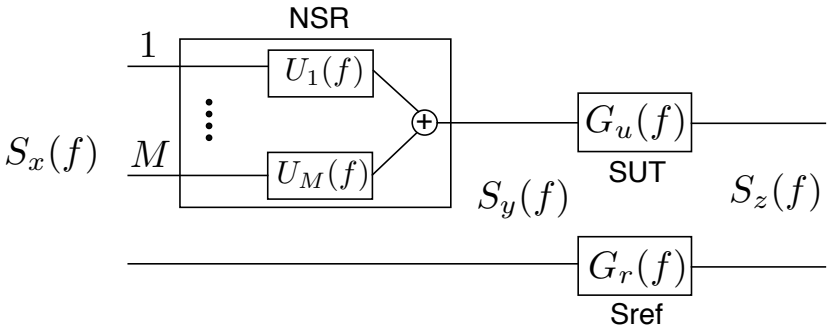

Figure 4.2: $S_x(f)$ *spectral matrix de dimension* $(M+1)$, $S_y(f)$ *spectral matrix de dimension* 2, $S_z(f)$ *spectral matrix de dimension* 2

The spectral matrix of the $(M+1)$-ary input signal writes:

$$S_x(f) = \gamma_s(f)\mathbb{1}_{M+1}\mathbb{1}_{M+1}^T + \Gamma(f) \tag{4.15}$$

where $\gamma_s(f)$ is the spectral density of the coherent part, which is generally acoustic. By acoustic we mean that the wave has a velocity c of the order of magnitude of 300 m/s. Let us notice that the associated wavelength of this acoustic part is greater than 75 meter for any frequencies below 4 Hz.

We assume that the spectral square matrix $\Gamma(f)$, whose size is $(M+1)$, is given by the expression (4.4) where $C(f)$ entries are given by (8.4) and

$$P(f) = \begin{bmatrix} \gamma_u^{1/2}(f) & 0 & \cdots & & 0 \\ 0 & \gamma_u^{1/2}(f) & \cdots & & 0 \\ \vdots & & & & \\ 0 & \cdots & 0 & \gamma_u^{1/2}(f) & 0 \\ 0 & \cdots & & 0 & \gamma_r^{1/2}(f) \end{bmatrix} \tag{4.16}$$

Here it is assumed that the noise has the same level $\gamma_u(f)$ on all NRS inlets but is different on the reference sensor denoted $\gamma_r(f)$.

It follows that the 2×2 spectral matrix associated to the signals at the

inputs of the two sensors writes:

$$S_y(f) = K(f)S_x(f)K^H(f) \tag{4.17}$$

where

$$K(f) = \begin{bmatrix} U_1^*(f) & \dots & U_M^*(f) & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}$$

where we assume that the transfer function of the pipe of the reference sensor is 1.

In the following we assume that $U_1(f) = \cdots = U_M(f) = U(f)$.

From (4.17) we derive that:

$$\begin{aligned} S_{y,11}(f) &= |U(f)|^2 \left(M^2\gamma_s(f) + \gamma_u(f)S_M(f)\right) \\ S_{y,22}(f) &= (\gamma_s(f) + \gamma_r(f)) \\ S_{y,12}(f) &= U^*(f)\left(M\gamma_s(f) + \gamma_u^{1/2}(f)\gamma_r^{1/2}(f)s_M(f)\right) \\ S_{y,21}(f) &= S_{y,12}^*(f) \end{aligned}$$

where

$$S_M(f) = \sum_{k=1}^{M}\sum_{k'=1}^{M} C_{k,k'} \quad \text{and} \quad s_M(f) = \sum_{k=1}^{M} C_{k,M+1}$$

Let us notice that for small values of ζ, the $(M+1)$ entries are spatially uncorrelated hence C is the identity leading to $S_M(f) = M$ and $s_M(f) = 0$.

We denote

$$S_z(f) = \begin{bmatrix} S_{UU}(f) & S_{UR}(f) \\ S_{RU}(f) & S_{RR}(f) \end{bmatrix}$$

with

$$\begin{aligned} S_{UU}(f) &= |G_u(f)|^2 |U(f)|^2 \left(M^2\gamma_s(f) + \gamma_u(f)S_M(f)\right) && (4.18) \\ S_{RR}(f) &= |G_r(f)|^2(\gamma_s(f) + \gamma_r(f)) && (4.19) \\ S_{UR}(f) &= G_u(f)G_r^*(f)U^*(f)\left(M\gamma_s(f) + \gamma_u^{1/2}(f)\gamma_r^{1/2}(f)s_M(f)\right) && (4.20) \\ S_{RU}(f) &= S_{UR}^*(f) && (4.21) \end{aligned}$$

Let us remark that a perfect NRS would have $U(f) = 1/M$ for all frequencies. Unfortunately that is not satisfied for high frequency values. Indeed it is known that the NRS presents a resonance above 2 Hz.

At first if we consider that $U(f) = 1/M$, $S_M(f) = M$ and $s_M(f) = 0$ then we obtain

$$\begin{aligned} S_{UU}(f) &= |G_u(f)|^2(\gamma_s(f) + \gamma_u(f)/M) && (4.22) \\ S_{RR}(f) &= |G_r(f)|^2(\gamma_s(f) + \gamma_r(f)) && (4.23) \\ S_{UR}(f) &= G_u(f)G_r^*(f)\gamma_s(f) && (4.24) \\ S_{RU}(f) &= S_{UR}^*(f) && (4.25) \end{aligned}$$

We recognize the classical expression which is the base of the *in-situ* calibration. We see also that the noise level is divided by M which is what we expect using an NRS. Unfortunately, since this ratio $\gamma_u(f)/\gamma_r(f)$ is unknown, the system is underdetermined. One way to circumvent this problem is to consider that the wind noises are very close to zero. Indeed if $\gamma_u(f) = \gamma_r(f) = 0$

$$G_u(f) = \frac{S_{UU}(f)}{S_{RU}(f)} G_r(f)$$

To test the zero-noise it is well-known that the MSC can be used but we have to keep in mind that the MSC equal 1 does not mean that the ratio between the sensor response is 1. It only means that the ratio is a transfer function as for example a pure number. It is the reason why the ratio presents a dip as we see below.

MSC

In this section we show that the MSC could be very close to 1 and the sensor frequency response ratio is not equal to 1. The MSC successively writes:

$$\begin{aligned} \mathrm{MSC}(f) &= \frac{|S_{UR}(f)|^2}{S_{UU}(f)S_{RR}(f)} \\ &= \frac{|M\gamma_s(f) + \gamma_u^{1/2}(f)\gamma_r^{1/2}(f)s_M(f))|^2}{(M^2\gamma_s(f) + \gamma_u(f)S_M(f))(\gamma_s(f) + \gamma_r(f))} \end{aligned} \tag{4.26}$$

We let $\rho_i(f) = \gamma_i(f)/\gamma_s(f)$. Then

$$\mathrm{MSC}(f) = \frac{|M + \rho_u^{1/2}(f)\rho_r^{1/2}(f)s_M(f))|^2}{(M^2 + \rho_u(f)S_M(f))(1 + \rho_r(f))} \tag{4.27}$$

It follows:

- For high frequencies $s_M(f) \approx 0$ and $S_M(f) \approx M$, hence

$$\mathrm{MSC}(f) \approx \frac{1}{(1 + \rho_u(f)/M)(1 + \rho_r(f))} \tag{4.28}$$

- For very low frequencies $s_M(f) = M$ and $S_M(f) = M^2$, hence:

$$\mathrm{MSC}(f) = \frac{(1 + \sqrt{\rho_u(f)\rho_r(f)})^2}{(1 + \rho_u(f))(1 + \rho_r(f))} \tag{4.29}$$

  the MSC could be close to 1 even if $\rho_u(f)$ and $\rho_r(f)$ are different of 0.

- For mid-range, $s_M(f) \approx \lambda_1 M$ and $S_M(f) \approx \lambda_2 M^2$, hence

$$\mathrm{MSC}(f) \approx \frac{(1 + \lambda_1\sqrt{\rho_u(f)\rho_r(f)})^2}{(1 + \lambda_2\rho_u(f))(1 + \rho_r(f))} \tag{4.30}$$

If λ_1 and λ_2 are close to 1, the MSC could be close to 1 even with $\rho_i(f) \neq 0$.

In conclusion MSC(f) close to 1 does not mean that $\rho_u(f)$ and $\rho_r(f)$ are close to 0 and in this case the system is still underdetermined.

Frequency response ratio

Assuming that $U(f) = 1/M$ for all frequencies, the frequency response ratio writes:

$$R(f) = \frac{S_{UU}(f)}{S^*_{UR}(f)} = \frac{G_u(f)}{G_r(f)} \times \frac{1 + \gamma_u(f) S_M(f)/M^2 \gamma_s(f)}{1 + \gamma_u^{1/2}(f)\gamma_r^{1/2}(f) s_M(f)/M\gamma_s(f)}$$

if $\gamma_u(f) = \gamma_r(f) = 1$, and $\gamma_u(f) = \rho\gamma_s(f)$:

$$R(f) = \frac{S_{UU}(f)}{S^*_{UR}(f)} = \frac{G_u(f)/M}{G_r(f)} \times \frac{M^2 + \rho S_M(f)}{M + \rho s_M(f)}$$

where the quantity $G_u(f)/M$ is provided by the Gabrielsson analysis and is very close to $1/M$, except above 2 Hz, due to the resonance effect. Numerical results are reported figure 4.3 for a given shape of $\rho(f)$.

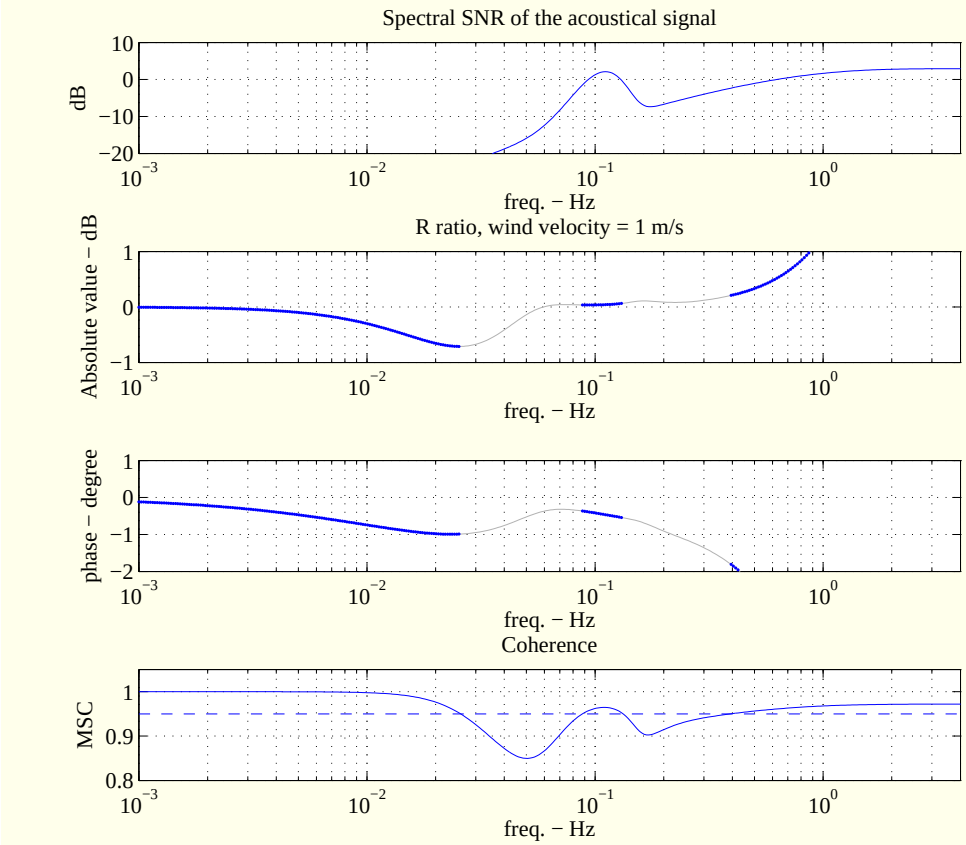

Figure 4.3: *The top curve represents the spectral ratio* $\rho(f)$*. The two following curves report the magnitude and the phase of the ratio The blue parts of corresponding to an MSC greater than* 0.95*. For frequencies over* 0.4 *Hz up to* 4 *Hz and for low noise levels, the curves reproduce the response of the NRS. In particular on the high frequency part the curves are related to a known resonance effect. For frequencies under* 0.08 *Hz and even for low values of* SNR*, the coherence is very high, because the wind turbulences appear as spatially coherent. In the mid frequency range we can have simultaneously a high MSC and a ratio less than* 1*.*

4. Results in I26

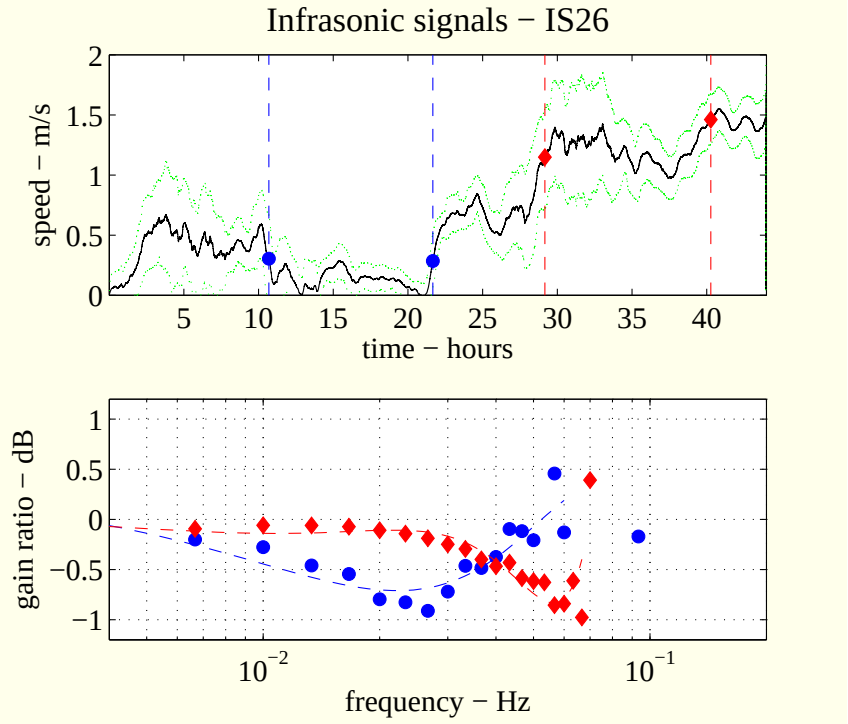

Figure 4.4: *Top figure: averaging of the wind speed on plain curve, standard deviation around the averaging in dotted curve. Markers indicate the two parts of interest. Bottom figure: curves associated to the two parts of interest. The shift between them has a ratio of about* 4 *which is about the ratio of the speed means od the two parts of interest.*

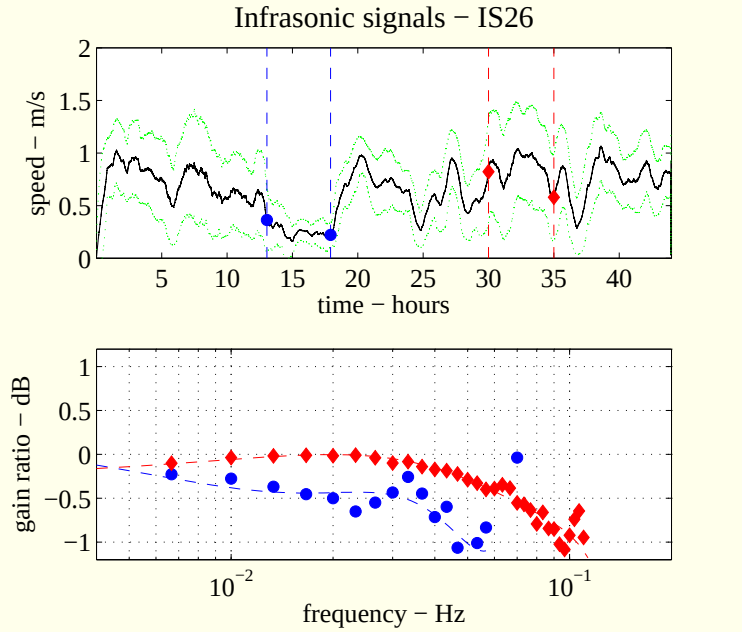

Figure 4.5: *see figure 4.4*

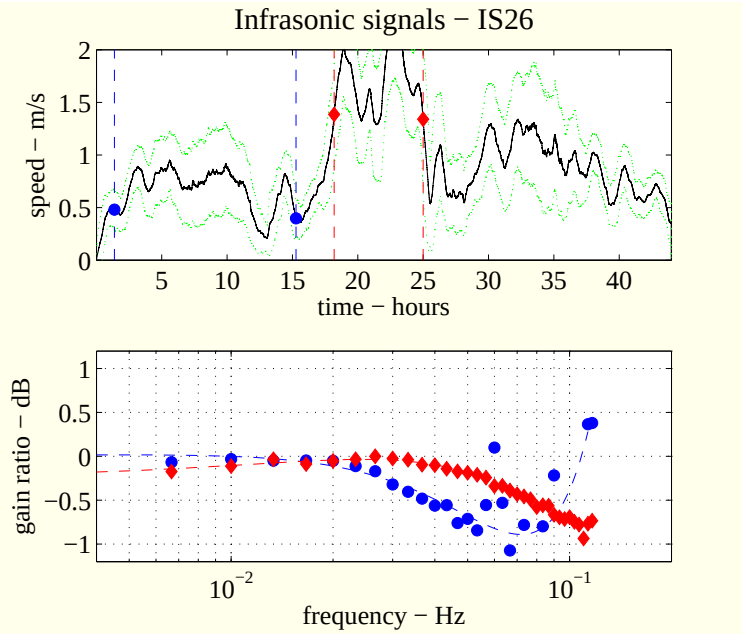

Figure 4.6: *see figure 4.4*

Chapter 5

Full process

1. 3 programs for test

The full process consists of

1. move data from the IDC, using `RUNextractfromDB.m`. The data with Matlab format is saved. This data consists of a structure called `records` with the following items:

   - `data`: sequence of samples,
   - `Fs_Hz`: sampling frequency,
   - `stime`, `time`: start and end times,
   - `station`: I26C1–I26C8 or I26H1–I26H8
   - `channel`: 'BDF' or 'LKO' or 'LWD' or 'LWS'

   Typically in the following we use BDF on H and C. The three other channels are only provided on H1. Channel LWD yields the wind speed measurement. The wind sensor is located at about 2 meters above the infrasonic sensor. The frequency sampling depends on the channel. Typically for infrasound signals, the frequency sampling is 20 Hz, and for wind features is 1 Hz.

2. using the filtercharacteristics, saved in `filtercharacteristics.mat` and the directory which contains the data from one of the 8 channels, run the program `estimationwithFB.m`. This program saves the useful elements for a display in a selected directory.

3. to display run the program `displaySUTresponse.m` for the file recorded in the selected directory of the previous item.

2. Input of the function `fbankanalysis.m`

- signals: an array of size $N \times 2$ where N is a number of samples, the first column is the signal from the SUT channel and the second for the SREF channel,

- filter bank settings: see section 2.,
- F_s: sampling frequency in Hz, typically 20 Hz,
- MSC threshold: threshold for the MSC, typically 0.98

Summary of the filter bank settings

```
%============================================================
%   filtercharact.designname: filter design (ex. 'butter')
%   filtercharact.Norder: order of the filter
%   filtercharact.Wlow_Hz:
%   filtercharact.Whigh_Hz:
%   filtercharact.SCPperiod_sec: duration in second
%                over which SPC is performed,
%                expected as stationarity time
%   filtercharact.windowshape: window shape for spectral
%                analysis (ex. 'hann' or 'hamming', ...)
%   filtercharact.overlapDFT: overlap rate for successive
%                DFT (typically 0.5)
%   filtercharact.overlapSCP: overlap rate for successive
%                spectral components (typically 0)
%   filtercharact.ratioDFT2SCP: ratio between period_sec and
%                DFT duration (typical integer value is 5).
%============================================================
```

- `designname` means that the name of the filter model. The Butterworth model is available in Matlab.
- `Norder` denotes the order of the filter. If 0 there is no filtering.
- `Wlow_Hz` denotes the low bound in Hz of the filter design,
- `Whigh_Hz` denotes the high bound in Hz of the filter design,
- `windowshape` denotes the weighted window. Many windows are available in Matlab. Window is used to do a compromise between the leakage and the bias. Hann's window is commonly used.
- `SCPperiod_sec` denotes the time duration expressed in second of the window used for the spectral estimation,
- `overlapDFT` overlapping rate for the spectral estimation,
- `overlapSCP` overlapping rate for the different spectral estimates,
- `ratioDFT2SCP` is an integer which denotes the ratio between the duration of the spectral estimation window and the duration of the DFT window.

Calculation scheme

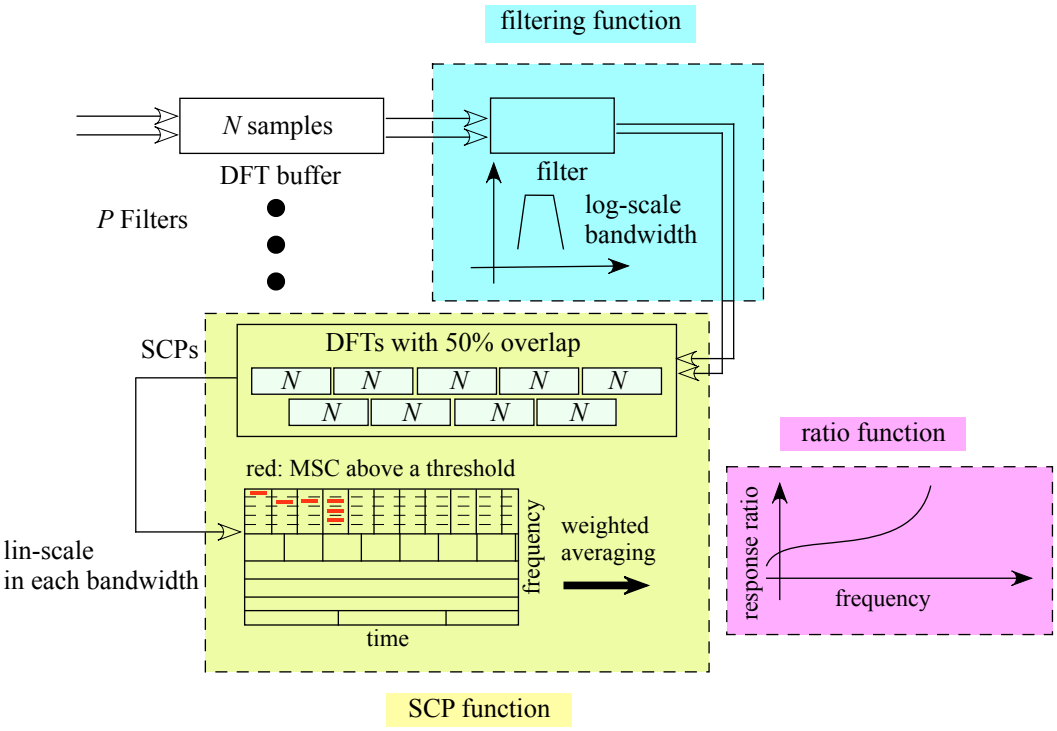

Figure 5.1: *FB analysis*

Let us report to the figure 5.1. The DFT buffer contains the data for a duration in accordance with the DFT analysis. Usually this duration is in the inverse of the bandwidth of the filter. The filter bandwidth is log-scaled in the frequency domain. If the SCP time window is 1000 seconds and the DFT buffer duration is a fifth i.e. 200 seconds with an overlap of 50%, the number of DFTs is 9. In this case the DFTs can be performed each time we progress in the buffer of 100 seconds. After 5 times the DFT duration, a weighted averaging is performed along the time in each filter bandwidth to obtain the SCPs taking into account the cells whose the MSC is above a given threshold, typically 0.98. The frequencies inside each bandwidth is uniformly spaced. With the current values of the filter bank, the number of frequency values for the full bandwidth estimation is less than 100. In the matlab function (see section 11) the averaging is obtained on two successive days.

The full calculation is based on the 3 following functions

- function `fbank.m`:
  - the function inputs are (1) the two signals, on SUT and SREF, (2) the filter characteristics and (3) the frequency sampling.
  - The filter outputs are the $P \times 2$ output signals.

- function `estimSCP.m`:
  - the inputs are the $P \times 2$ signals, the frequency response of the SREF, the DFT overlap, the SCP overlap, the frequency sampling and the smooth window.
  - the outputs are the SCPs which consist of P time-spectral structures, each of them consists of an array of size $K \times T$ where K is the number of frequency dots in the associated frequency band and T the number of SCP time slots during the full duration of the analysis.

  function `estimSUT.m`:
  - the inputs are the SCPs provided by the function `estimSCP.m` and the MSC threshold.
  - the outputs are the estimations of the ratio, expression (2.8), denoted `Rsup` which consists of P structures. Each structure contains many elements. The most important are:
    * the module of the ratio averaged on the different SCP time slots in the time frequency cells where the MSC is over the threshold. It is denoted `Rsup.modcst`, and the associated standard deviation.
    * the phase of the ratio averaged on the different SCP time slots in the time frequency cells where the MSC is over the threshold. It is denoted `Rsup.phasecst`, and the associated standard deviation

    It is worth to notice that the variables with the indication `tab` are for the different SCP time slots (without averaging) and the variables with the indication `cst` for the values associated to the time frequency cells where the MSC is over the threshold.

3. Output of the function `fbankanalysis.m`

```
% 1) SUTs: structures P x 1
%         xx.estimRsup: Rsup ratio
%         xx.estimRinf: Rinf ratio
%         xx.allMSCs: allMSCs;
%         xx.Nsupthreshold: count of the number of values over
%                   the threshold
%         xx.Nsupthresholdintheband: counts of the number of values
%                   over the threshold in the filter bandwidth
%         xx.frqsFFT_Hz: cell P x 1, each cell consists of
%                   frequency list in Hz of the DFTs
%         xx.SCP = all spectral components
%         xx.indexinsidefreqband = P x 1, indices of the
%                    frequency bounds of each filter
%                    in the xx.frqsFFT_Hz
% 2) filteredsignals,
% 3) allfrqsFFT_Hz, cell P x 1, each cell for each bank filter consists of
%             frequency list in Hz of the DFTs
% 4) alltimes_sec: cell  Px 1, each cell consists of the structure
%             yy.FFT:     time list in second of the DFTs
%             yy.SD:      time list in second of the SCPs
%             yy.signals: time list in second of the signals
% 5) filterbank
```

Remark for developper

It is advised to provide at the output of the FB analysis function, on one hand the response ratios in each of the P frequency bands with the associated period and on the other the MSC levels greater than the given threeshold. These two elements have the same size. An external function will perform the averaging over a given period of time with the weights as defined in expression 2.16.

It follows that

- an extra input must be considered to take into account the duration of the averaging. In our code this time is fixed to two consecutive days, i.e. 48 hours
- an extra output must be considered to provide the MSC levels associated to each spectral component. We only consider the time-frequency cells where the current MSC is above the given threshold.

Part II

Annexes

Chapter 6

Wide sense stationary process

A multivariate time series x_n is said to be a zero-mean wide-sense stationary (WSS) process of second order iif $\mathbb{E}[x_n] = 0$, trace $\left(\mathbb{E}\left[x_n x_n^H\right]\right) < +\infty$ and the sequence of covariance matrices[c0] $R(h) = \mathbb{E}\left[x_{n+h} x_n^H\right]$ does not depend on n. It follows that $R(h) = R^H(-h)$. Therefore if x_n is real $R(h) = R^T(-h)$.

Under very general conditions, its Fourier transform

$$\Gamma(f) = \sum_h R(h) e^{-2j\pi hf}, \quad \text{with} \quad f \in (0,1)$$

exists which is called the spectral matrix sequence. A fundamental property says that

$$\forall f, \quad \Gamma(f) \geq 0$$

The non-negativity of $\Gamma(f)$ implies that $\Gamma_{ij}(f) = \Gamma_{ji}^*(f)$ and $\Gamma_{ii}(f) \geq 0$. If x_n is real alors $\Gamma(f) = \Gamma(-f)$. For example for a bivariate process we have

$$\Gamma(f) = \begin{bmatrix} \Gamma_{11}(f) & \Gamma_{12}(f) \\ \Gamma_{21}(f) & \Gamma_{22}(f) \end{bmatrix}$$

where $\Gamma_{11}(f)$ and $\Gamma_{22}(f)$ are both positive functions. The coherence matrix is defined by

$$\begin{aligned} C(f) &= \begin{bmatrix} \Gamma_{11}^{-1/2}(f) & 0 \\ 0 & \Gamma_{22}^{-1/2}(f) \end{bmatrix} \begin{bmatrix} \Gamma_{11}(f) & \Gamma_{12}(f) \\ \Gamma_{21}(f) & \Gamma_{22}(f) \end{bmatrix} \begin{bmatrix} \Gamma_{11}^{-1/2}(f) & 0 \\ 0 & \Gamma_{22}^{-1/2}(f) \end{bmatrix} \\ &= \begin{bmatrix} 1 & C_{12}(f) \\ C_{21}(f) & 1 \end{bmatrix} \end{aligned}$$

with $C_{12}(f) = C_{2,1}^*(f)$. Let

$$\eta_{12}(f) = \frac{\Gamma_{12}(f)}{\sqrt{\Gamma_{11}(f)\Gamma_{22}(f)}} \tag{6.1}$$

[c0] The superscript H denotes the transposition-conjugaison, T the transposition and $*$ the conjugaison.

which is called the normalized cross spectral density and

$$\mathrm{MSC}(f) = \frac{|\Gamma_{12}(f)|^2}{\Gamma_{11}(f)\Gamma_{22}(f)} \tag{6.2}$$

which is called the Magnitude Squared Coherence (MSC) or shortly the coherence. A fundamental property says that

$$\forall f, \quad \mathrm{MSC}(f) \leq 1$$

and the equality occurs iff it exists a filter s.t. one signal of the bivariate process is the filtering of the other. In this case $\Gamma(f)$ is, up to a a multiplicative positive value, a projector of rank 1. A fundamental example is given by

$$\left\{ \begin{array}{rcl} x_{1,n} & = & g_{1,n} \star s_n \\ x_{2,n} & = & g_{2,n} \star s_n \end{array} \right. \quad \Leftrightarrow \quad x_n = \begin{bmatrix} x_{1,n} \\ x_{2,n} \end{bmatrix} = h_n \star s_n$$

where s_n is a monovariate WSS process with spectral density $\gamma_s(f)$. Then the spectral matrix of the bivariate process writes

$$\Gamma(f) = \gamma_s(f) G(f) G^H(f)$$

and

$$C(f) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix}$$

In the noisy case if the spectral matrix of the two noises is diagonal with respective spectral densities $\gamma_1(f)$ and $\gamma_2(f)$, the spectral matrix writes:

$$\Gamma(f) = \gamma_s(f) \begin{bmatrix} |G_1(f)|^2 & G_1(f)G_2^*(f) \\ G_1^*(f)G_2(f) & |G_2(f)|^2 \end{bmatrix} + \begin{bmatrix} \gamma_1(f)|G_1(f)|^2 & 0 \\ 0 & \gamma_2(f)|G_2(f)|^2 \end{bmatrix}$$

then

$$\begin{array}{rcl} \mathrm{MSC}(f) & = & \dfrac{|\Gamma_{1,2}(f)|^2}{\Gamma_{1,1}(f)\Gamma_{2,2}(f)} \\ & = & \dfrac{1}{(1+\mathrm{SNR}_1^{-1}(f))(1+\mathrm{SNR}_2^{-1}(f))} \end{array}$$

where

$$\mathrm{SNR}_1(f) = \frac{\gamma_s(f)}{\gamma_1(f)} \quad \text{and} \quad \mathrm{SNR}_2(f) = \frac{\gamma_s(f)}{\gamma_2(f)}$$

All these expressions can be generalized for more than 2 dimensions.

Chapter 7

Theoretical results on spectral estimation

1. Non parametrical spectral estimation

Let us consider $x_{n=0:N-1}$ N successive values of a real WSS bivariate process with zero-mean and spectral matrix $\Gamma(f)$. Its DFT writes

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n e^{-2j\pi kn/N}$$

It can be shown that, for large N, and for $k \neq 0$ and $k \neq N/2$:

$$X_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}_c(0, \Gamma_k)$$

where $\Gamma_k = \Gamma(k/N)$ and where $\mathcal{N}_c$ denotes the complex circular gaussian distribution. The periodogram is defined by

$$P_k = X_k X_k^H$$

In the following section 2., it is shown that the expectation of the periodogram P_k is Γ_k, but unfortunately the dispersion around Γ_k never goes to 0 when N goes to infinity. The periodogram is not a consistent estimate of Γ_k. A fundamental way to have consistency is to smooth the periodogram (smoothed peridogram). An other approach consists to segment the data in $(2M+1)$ 50%-overlapping blocks and averaging the $2M+1$ periodograms, applying a window on each block. That is called Welch's approach in the literature. The 2 approaches are very similar.

For the smoothed periodogram estimator at any frequency f writes:

$$\widehat{\Gamma}_k = \sum_{m=0}^{2M} W_m P_{k-M+m} \tag{7.1}$$

where k is an integer s.t. $\frac{k-1/2}{N} \leq f < \frac{k+1/2}{N}$. That leads to a mean square

error (MSE) that writes

$$\mathrm{MSE}_k \;=\; \mathcal{G}_k + \mathcal{B}_k\mathcal{B}_k^H \tag{7.2}$$

where the bias

$$\mathcal{B}_k = \Gamma_k - \sum_{k=0}^{2M} W_m \Gamma_{k-M+m} \tag{7.3}$$

and the covariance

$$\mathcal{G}_k = \sum_{k=0}^{2M} W_m^2 \mathrm{diag}\left(\Gamma_{k-M+m}\right) \mathrm{diag}\left(\Gamma_{k-M+m}\right)^H \tag{7.4}$$

The choice of the window and its length results from a compromise between bias and variance. Unfortunately this compromise is related to the shape of the spectrum which is *unknown*. Using (7.2) we derive a confidence interval replacing deterministic values by their estimates and assuming a gaussian distribution.

2. More periodogram properties

Let us consider $x_{n=0:N-1}$ a sequence of N consecutive values of a real wide-sense stationary bivariate process with zero-mean and spectral matrix $\Gamma(f)$. An fundamental property says that $\Gamma(f) \geq 0$ for any value of f. The DFT of the sequence writes

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n e^{-2j\pi kn/N}$$

It can be shown that, for large N, and for $k \neq 0$ and $k \neq N/2$:

$$X_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}_c(0, \Gamma_k)$$

where $\Gamma_k = \Gamma(k/N)$ and where $\mathcal{N}_c$ denotes the complex circular gaussian distribution whose the probability density writes:

$$p_{X_k}(x_k) = \frac{1}{\pi^2 |\Gamma_k|} e^{-x_k^H \Gamma_k^{-1} x_k}$$

Recall that for any complex gaussian random vector with zero-mean and covariance Γ we have by definition:

$$\mathbb{E}\left[XX^H\right] = \Gamma$$

and

$$\mathbb{E}\left[XX^T\right] = 0$$

and if $X_{1\cdots 4}$ denote 4 gaussian complex zero-mean r.v., we have

$$\begin{aligned}\mathbb{E}\left[X_1^{\beta_1}X_2^{\beta_2}X_3^{\beta_3}X_4^{\beta_4}\right] &= \mathbb{E}\left[X_1^{\beta_1}X_2^{\beta_2}\right]\mathbb{E}\left[X_3^{\beta_3}X_4^{\beta_4}\right] \\ &+\mathbb{E}\left[X_1^{\beta_1}X_3^{\beta_3}\right]\mathbb{E}\left[X_2^{\beta_2}X_4^{\beta_4}\right]+\mathbb{E}\left[X_1^{\beta_1}X_4^{\beta_4}\right]\mathbb{E}\left[X_2^{\beta_2}X_3^{\beta_3}\right]\end{aligned} \tag{7.5}$$

where β_i is either "star" or "not-star".

The formula (7.5) is very useful to perform the second order moments of the periodogram defined by

$$P_k = X_k X_k^H = \begin{bmatrix} X_1X_1^* & X_1X_2^* \\ X_1^*X_2 & X_2X_2^* \end{bmatrix} \tag{7.6}$$

Let us remark that if we multiply X_1 and X_2 by $e^{j\theta}$ the matrix P_k is unchanged. Then given P_k we can determine X_1 and X_2 up to a constant of modulus 1. It is possible to derive the distributions of P_k entries from the distribution of X_k. Alleviating the notations by omitting the index k, the distribution of these two complex r.v. X_1, and X_2 writes

$$\begin{aligned}p_{X_1,X_2}(x_1,x_2) &= \frac{1}{\pi^2(\Gamma_{1,1}\Gamma_{2,2}-|\Gamma_{1,2}|^2)} \\ &\exp\left(-\frac{|x_1|^2\Gamma_{2,2}+|x_2|^2\Gamma_{1,1}-2|x_1|\,|x_2||\Gamma_{2,1}|\cos(\alpha)}{(\Gamma_{1,1}\Gamma_{2,2}-|\Gamma_{1,2}|^2)}\right)\end{aligned}$$

where $\alpha = \arg x_1 - \arg x_2 - \arg \Gamma_{1,2}$.

In practical cases we often need only the second order moments.

Example 1 (Variance of $P_{1,1,k}$) *Let us recall that $P_{1,1,k} \geq 0$. It comes*

$$\mathbb{E}\left[P_{1,1,k}\right] = \mathbb{E}\left[X_1X_1^*\right] = \Gamma_{1,1} \geq 0$$

and

$$\mathbb{E}\left[P_{1,1,k}P_{1,1,k}^*\right] = \mathbb{E}\left[X_{1,k}X_{1,k}^*X_{1,k}^*X_{1,k}\right] = 2\mathbb{E}\left[X_{1,k}X_{1,k}^*\right]\mathbb{E}\left[X_{1,k}X_{1,k}^*\right]+0 = 2\Gamma_{1,1,k}^2$$

then

$$\mathrm{var}\left(P_{1,1,k}\right) = \mathbb{E}\left[P_{1,1,k}P_{1,1,k}^*\right] - \mathbb{E}\left[P_{1,1,k}\right]\mathbb{E}\left[P_{1,1,k}\right]^* = \Gamma_{1,1,k}^2$$

Example 2 (Variance of $P_{1,2,k}$) *Consider we want to determine the second order moments of $P_{1,2,k}$ which is complex. It comes*

$$\mathbb{E}\left[P_{1,2,k}\right] = \mathbb{E}\left[X_{1,k}X_{2,k}^*\right] = \Gamma_{1,2}$$

and

$$\begin{aligned}\mathbb{E}\left[P_{1,2,k}P_{1,2,k}^*\right] &= \mathbb{E}\left[X_{1,k}X_{2,k}^*X_{1,k}^*X_{2,k}\right] \\ &= \mathbb{E}\left[X_{1,k}X_{1,k}^*\right]\mathbb{E}\left[X_{2,k}X_{2,k}^*\right]+\mathbb{E}\left[X_{1,k}X_{2,k}^*\right]\mathbb{E}\left[X_{2,k}X_{1,k}^*\right]+0 \\ &= \Gamma_{1,1,k}\Gamma_{2,2,k}+\Gamma_{1,2}\Gamma_{1,2}^*\end{aligned}$$

then

$$\mathrm{var}\left(P_{1,2,k}\right) = \mathbb{E}\left[P_{1,2,k}P_{1,2,k}^*\right] - \mathbb{E}\left[P_{1,2,k}\right]\mathbb{E}\left[P_{1,2,k}^*\right] = \Gamma_{1,1,k}\Gamma_{2,2,k}$$

On the other hand

$$\mathbb{E}\left[P_{1,2,k}P_{1,2,k}\right] = \mathbb{E}\left[X_1X_2^*X_1X_2^*\right] = 2\Gamma_{1,2,k}^2$$

which is complex.

Example 3 *We want to determine the variance of $P_{1,2,k,R}$. To alleviate the notations we omit the index k and let $G = P_{1,2}$, $G = P_{1,2}$ and $G_R = P_{1,2,R}$. Then*

$$G_R = \frac{1}{2}(G + G^*)$$

at first

$$\mathbb{E}\left[G_R\right] = \frac{1}{2}(\Gamma_{1,2} + \Gamma_{1,2}^*) = \Gamma_{1,2,R}$$

then

$$\mathbb{E}\left[G_R^2\right] = \frac{1}{4}\mathbb{E}\left[GG + G^*G^* + 2GG^*\right]$$

at first

$$\begin{aligned}\mathbb{E}\left[G^2\right] &= \mathbb{E}\left[X_1X_2^*X_1X_2^*\right] \\ &= \mathbb{E}\left[X_1X_2^*\right]\mathbb{E}\left[X_1X_2^*\right] + 0 + \mathbb{E}\left[X_1X_2^*\right]\mathbb{E}\left[X_1X_2^*\right] \\ &= 2(\Gamma_{1,2})^2\end{aligned}$$

similarly

$$\mathbb{E}\left[G^*G^*\right] = 2(\Gamma_{1,2}^*)^2$$

and

$$\begin{aligned}2\mathbb{E}\left[GG^*\right] &= 2\mathbb{E}\left[X_1X_2^*X_1^*X_2\right] \\ &= 2|\Gamma_{1,2}|^2 + 2\Gamma_{1,1}\Gamma_{2,2} + 0\end{aligned}$$

then

$$\mathrm{var}\left(G_{1,2,R}\right) = \underbrace{\frac{1}{2}(\Gamma_{1,2})^2 + \frac{1}{2}(\Gamma_{1,2}^*)^2}_{\Gamma_{1,2,R}^2 - \Gamma_{1,2,I}^2} + \frac{1}{2}|\Gamma_{1,2}|^2 + \frac{1}{2}\Gamma_{1,1}\Gamma_{2,2} - \Gamma_{1,2,R}^2$$

Hence

$$\mathrm{var}\left(P_{1,2,k,R}\right) = \frac{1}{2}\Gamma_{1,1}\Gamma_{2,2} + \frac{1}{2}\Gamma_{1,2,R}^2 - \frac{1}{2}\Gamma_{1,2,I}^2$$

Similarly

$$\mathrm{var}\left(P_{1,2,k,I}\right) = \frac{1}{2}\Gamma_{1,1}\Gamma_{2,2} + \frac{1}{2}\Gamma_{1,2,I}^2 - \frac{1}{2}\Gamma_{1,2,R}^2$$

Properties of the periodogram

Similarly we do in the previous examples, we have

$$\mathbb{E}\left[P_{k,1,1}\right] = \Gamma_{k,1,1}, \qquad \mathrm{var}\left(P_{k,1,1}\right) = \Gamma_{k,1,1}^2 \tag{7.7}$$

$$\mathbb{E}\left[P_{k,2,2}\right] = \Gamma_{k,2,2}, \qquad \mathrm{var}\left(P_{k,2,2}\right) = \Gamma_{k,2,2}^2 \tag{7.8}$$

$$\mathbb{E}\left[P_{k,1,2}\right] = \Gamma_{k,1,2} \qquad \mathrm{var}\left(P_{k,1,2}\right) = \Gamma_{k,1,1}\Gamma_{k,2,2} \tag{7.9}$$

That means that the expectation of the periodogram P_k is Γ_k, that is well! But, unfortunately, the dispersion around Γ_k not only never goes to 0 when N goes to infinity, but it is of the order of magnitude of what we want to estimate. Consequently the periodogram is not a good estimator of Γ_k.

But all good estimators of Γ_k are built on the periodogram either by smoothing or by averaging. For example if we use a smoothing approach the estimator writes

$$\widehat{\Gamma}_k = \sum_{m=0}^{2M} W_m P_{k-M+m} \tag{7.10}$$

with the following properties:

- $\sum_{m=0}^{2M} W_m = 1$
- $\overline{W}_M^2 = \sum_{m=0}^{2M} W_m^2 \to 0, \quad M \to \infty$
- $M = N^\beta$ with $\beta \in (0,1)$, typically β is between 0.2 and 0.3. That induces that both M and N/M go to infinity when N goes to infinity.

It follows that

$$\mathbb{E}\left[\widehat{\Gamma}_k\right] = \sum_{k=0}^{2M} W_m \Gamma_{k-M+m}$$

and therefore the estimator is biased with bias given by

$$\mathcal{B}_k = \Gamma_k - \sum_{k=0}^{2M} W_m \Gamma_{k-M+m} \tag{7.11}$$

From the properties of P_k we derive that the variance of the 4 components of $\widehat{\Gamma}_k$ are given by the 4 elements of

$$\mathcal{G}_k = \sum_{k=0}^{2M} W_m^2 \mathrm{diag}\left(\Gamma_{k-M+m}\right) \mathrm{diag}\left(\Gamma_{k-M+m}\right)^H$$

It results that the mean square error (MSE) writes as the 2 by 2 matrix

$$\mathrm{MSE}_k \;=\; \mathcal{G}_k + \mathcal{B}_k \mathcal{B}_k^H$$

An estimate of the MSE can be obtained replacing the values of Γ_k by their estimates. Generally the variance decreases when the bias increases. There-

fore the choice of a smoothing window results in a trade-off between the bias and the variance.

It is worth to noting that $\mathcal{B}_k$, given by (7.11) represents the bias of the estimators. Naively you could imagine to subtract the bias to the estimator, replacing in (7.11) the unknown values by the observed values. Of course the expectation of this quantity is 0 but for one outcome it could large and therefore by subtracting it we will re-introduce more variance.

It is worth to noting that the spectrum estimator variance decreases in $1/(2M+1)$ where M is the length of the smoothing window for smoothed periodogram and the number of windows for Welch's approach, but it is not related to the data number N which is assumed to be *infinite*. In the practical cases, we derive the value of M and the window shape from a given accuracy. Then N is chosen to be huge w.r.t. M as for example $N = M^3$.

3. Second order moments for the smoothed periodogram

We omit the index k and let

$$\widehat{\Gamma} = \frac{1}{2M+1}\sum_{n=0}^{2M+1} X_n X_n^H = \begin{bmatrix} \widehat{\Gamma}_{1,1} & \widehat{\Gamma}_{1,2} \\ \widehat{\Gamma}_{1,2}^* & \widehat{\Gamma}_{2,2} \end{bmatrix} \to \Gamma = \begin{bmatrix} \Gamma_{1,1} & \Gamma_{1,2} \\ \Gamma_{1,2}^* & \Gamma_{2,2} \end{bmatrix} \tag{7.12}$$

where $X_{0\cdots 2M}$ are i.i.d. bivariate gaussian complex vector with zero-mean and covariance Γ. $\Gamma_{1,1}$ and $\Gamma_{2,2}$ are real. We let $\Gamma_{1,2} = \Gamma_{1,R} + j\Gamma_{1,2,I}$.

Remark 1 *Let us remark that the expression* (7.12) *is as an approximation of the smoothed periodogram. Indeed (i) the window is assumed to be the rectangular window and more important (ii) the r.v. X_n are assumed to have the same variance whereas in the equation* (7.10) *the r.v. have different variance, saying Γ_k. For non identically distributed r.v. the limit central theorem is a little bit more complicate.*

Here, for the two first order moments we have to perform the two first order moments of $X_n X_n^H$ and then divide by $1/2M+1$ for the second order moments. We have

$$\mathbb{E}\left[\widehat{\Gamma}\right] = \Gamma$$

$$\mathrm{var}\left(\widehat{\Gamma}_{1,1}\right) = \frac{1}{2M+1}\Gamma_{1,1}^2$$

$$\mathrm{var}\left(\widehat{\Gamma}_{2,2}\right) = \frac{1}{2M+1}\Gamma_{2,2}^2$$

$$\mathrm{var}\left(\widehat{\Gamma}_{1,2}\right) = \frac{1}{2M+1}\Gamma_{1,1}\Gamma_{2,2}$$

$$\mathrm{var}\left(\widehat{\Gamma}_{1,2,R}\right) = \frac{1}{2M+1}\frac{1}{2}\left(|\Gamma_{1,2}|^2 + \Gamma_{1,1}\Gamma_{2,2}\right)$$

and

$$\text{var}\left(\widehat{\Gamma}_{1,2,I}\right) = \frac{1}{2M+1}\frac{1}{2}\left(|\Gamma_{1,2}|^2 + \Gamma_{1,1}\Gamma_{2,2}\right)$$

It can be shown that $\Gamma_{1,2,R}$ and $\Gamma_{1,2,R}$ are correlated with

$$\text{cov}\left(\widehat{\Gamma}_{1,2,R}, \widehat{\Gamma}_{1,2,I}\right) = \Gamma_{1,2,R}\Gamma_{1,2,I}$$

4. Approximate variances

δ-method

The so called δ-method allows to derive the covariance of

$$Z = f(Y)$$

from the covariance of Y where $f : \mathbb{R}^m \mapsto \mathbb{R}^q$. We let $\mu_Y = \mathbb{E}[Y]$. Using the first order Taylor expansion of f in the neighborhood of μ_Y, we write

$$Z \approx f(\mu_Y) + J_{\mu_Y}(Y - \mu_Y)$$

where J_{μ_Y} is the Jacobian of f, which is a matrix of size $m \times q$, performed in $y = \mu_Y$. Therefore at first order

$$\mathbb{E}[Z] \approx f(\mu_Y) + J_{\mu_Y}\mathbb{E}[Y - \mu_Y] = f(\mu_Y) + 0$$

then

$$Z - \mathbb{E}[Z] \approx J_{\mu_Y}(Y - \mu_Y)$$

therefore, according to the definition $\text{cov}(Z) = \mathbb{E}\left[(Z - \mathbb{E}[Z])(Z - \mathbb{E}[Z])^H\right]$, we have

$$\text{cov}(Z) \approx J_{\mu_Y}\text{cov}(Y)\, J_{\mu_Y}^H$$

Approximate variance of $\arg\widehat{\lambda}_k$

$$\tan(\phi) = \frac{\Gamma_{1,2,I}}{\Gamma_{1,2,R}}$$

Then

$$d\phi = \frac{\Gamma_{1,2,R}d\Gamma_{1,2,I} - \Gamma_{1,2,I}d\Gamma_{1,2,R}}{|\Gamma_{1,2}|^2}$$

Therefore

$$\text{var}\left(\widehat{\phi}\right) = \frac{1}{|\Gamma_{1,2}|^4}\left(\Gamma_{1,2,R}^2\text{var}\left(\widehat{\Gamma}_{1,2,I}\right) + \Gamma_{1,2,I}^2\text{var}\left(\widehat{\Gamma}_{1,2,R}\right) - 2\Gamma_{1,2,R}\Gamma_{1,2,I}\text{var}\left(\widehat{\Gamma}_{1,2,R}, \widehat{\Gamma}_{1,2,I}\right)\right)$$

Using that

$$\text{var}\left(\widehat{\Gamma}_{1,2,R}\right) = \frac{1}{2}\Gamma_{1,1}\Gamma_{2,2} + \frac{1}{2}\Gamma_{1,2,R}^2 - \frac{1}{2}\Gamma_{1,2,I}^2$$

$$\text{var}\left(\widehat{\Gamma}_{1,2,I}\right) = \frac{1}{2}\Gamma_{1,1}\Gamma_{2,2} + \frac{1}{2}\Gamma_{1,2,I}^2 - \frac{1}{2}\Gamma_{1,2,R}^2$$

and

$$\text{cov}\left(\widehat{\Gamma}_{1,2,R}, \widehat{\Gamma}_{1,2,I}\right) = \Gamma_{1,2,R}\Gamma_{1,2,I}$$

we get

$$\text{var}\left(\widehat{\phi}\right) = \frac{1-\mu}{2\mu}$$

Now if we smooth on a rectangular window with weights $1/(2M+1)$ see equation (**??**), the variance is given by

$$\text{var}\left(\widehat{\phi}\right) = \frac{1}{2M+1}\frac{1-\mu}{2\mu}$$

We can approximate the distribution of $\widehat{\phi}$ by a gaussian with mean ϕ and variance $\frac{1}{2M+1}\frac{1-\mu}{2\mu}$. The green curve of the third figure of **??** uses this approximation.

Approximate variance of $|\widehat{\lambda}_k^{(1)}|$

We have

$$R_1 = \frac{\sqrt{\Gamma_{1,2,R}^2 + \Gamma_{1,2,I}^2}}{\Gamma_{2,2}}$$

Then

$$\begin{aligned} dR_1 &= \frac{\Gamma_{1,2,R}d\Gamma_{1,2,R} + \Gamma_{1,2,I}d\Gamma_{1,2,I}}{\Gamma_{2,2}\sqrt{\Gamma_{1,2,R}^2 + \Gamma_{1,2,I}^2}} - \frac{\sqrt{\Gamma_{1,2,R}^2 + \Gamma_{1,2,I}^2}}{\Gamma_{2,2}^2}d\Gamma_{2,2} \\ &= \frac{\Gamma_{2,2}\Gamma_{1,2,R}d\Gamma_{1,2,R} + \Gamma_{2,2}\Gamma_{1,2,I}d\Gamma_{1,2,I} - |\Gamma_{1,2}|^2 d\Gamma_{2,2}}{\Gamma_{2,2}^2|\Gamma_{1,2}|} \end{aligned}$$

Therefore we have to determine the expression of the variances of $\widehat{\Gamma}_{1,2,R}$, $\widehat{\Gamma}_{1,2,I}$, $\widehat{\Gamma}_{2,2}$, and the 3 covariances $\text{cov}\left(\widehat{\Gamma}_{1,2,R}, \widehat{\Gamma}_{1,2,I}\right)$, $\text{cov}\left(\widehat{\Gamma}_{1,2,R}, \widehat{\Gamma}_{2,2}\right)$ and $\text{cov}\left(\widehat{\Gamma}_{1,2,I}, \widehat{\Gamma}_{2,2}\right)$.

$$\mathbb{E}\left[\widehat{\Gamma}_{1,2,R}, \widehat{\Gamma}_{2,2}\right] = \frac{1}{2}\mathbb{E}\left[(\widehat{\Gamma}_{1,2} + \widehat{\Gamma}_{1,2}^*)\widehat{\Gamma}_{2,2}\right] = \frac{1}{2}\mathbb{E}\left[(X_1X_2^* + X_1^*X_2)X_2X_2^*\right]$$

$$\frac{1}{2}\mathbb{E}\left[X_1X_2^*X_2X_2^*\right] + \frac{1}{2}\mathbb{E}\left[X_1^*X_2X_2X_2^*\right] = \Gamma_{1,2}\Gamma_{2,2} + \Gamma_{1,2}^*\Gamma_{2,2} = 2\Gamma_{1,2,R}\Gamma_{2,2}$$

$$\operatorname{cov}\left(\widehat{\Gamma}_{1,2,R}, \widehat{\Gamma}_{2,2}\right) = \Gamma_{1,2,R}\Gamma_{2,2}$$

Similarly

$$\operatorname{cov}\left(\widehat{\Gamma}_{1,2,I}, \widehat{\Gamma}_{2,2}\right) = \Gamma_{1,2,I}\Gamma_{2,2}$$

Then

$$\operatorname{var}(R_1) = \frac{1}{\Gamma_{2,2}^4 |\Gamma_{1,2}|^2}\left(A + B + C + D + E + F\right) \tag{7.13}$$

$$A = \Gamma_{2,2}^2\Gamma_{1,2,R}^2\left(\frac{1}{2}\Gamma_{1,1}\Gamma_{2,2} + \frac{1}{2}\Gamma_{1,2,R}^2 - \frac{1}{2}\Gamma_{1,2,I}^2\right)$$

$$B = \Gamma_{2,2}^2\Gamma_{1,2,I}^2\left(\frac{1}{2}\Gamma_{1,1}\Gamma_{2,2} + \frac{1}{2}\Gamma_{1,2,I}^2 - \frac{1}{2}\Gamma_{1,2,R}^2\right)$$

$$C = |\Gamma_{1,2}|^4\Gamma_{2,2}^2$$

$$D = 2\Gamma_{2,2}^2|\Gamma_{1,2}|^2\Gamma_{1,2,R}\Gamma_{1,2,I}$$

$$E = -2|\Gamma_{1,2}|^2\Gamma_{2,2}^2\Gamma_{1,2,R}^2$$

$$F = -2|\Gamma_{1,2}|^2\Gamma_{2,2}^2\Gamma_{1,2,I}^2$$

$$E + F = -2\Gamma_{2,2}^2|\Gamma_{1,2}|^4$$

$$A + B = \Gamma_{2,2}^2\left(\frac{1}{2}\Gamma_{1,1}\Gamma_{2,2}|\Gamma_{1,2}|^2 + \frac{1}{2}|\Gamma_{1,2}|^4 - 2\Gamma_{1,2,R}^2\Gamma_{1,2,I}^2\right)$$

$$A + B + D + E + F = \Gamma_{2,2}^2\left(\frac{1}{2}\Gamma_{1,1}\Gamma_{2,2}|\Gamma_{1,2}|^2 + \frac{1}{2}|\Gamma_{1,2}|^4 - 2|\Gamma_{1,2}|^4\right)$$

$$\operatorname{var}\left(|\Gamma_{1,2}|/\widehat{\Gamma}_{2,2}\right) = \frac{1}{2}\left(\frac{\Gamma_{1,1}\Gamma_{2,2} - |\Gamma_{1,2}|^2}{\Gamma_{2,2}^2}\right)$$

Approximate variance of $|\widehat{\lambda}_k^{(2)}|$

From the previous section we have

$$\operatorname{var}\left(|\widehat{\Gamma}_{2,1}|/\widehat{\Gamma}_{1,1}\right) = \frac{1}{2}\left(\frac{\Gamma_{1,1}\Gamma_{2,2} - |\Gamma_{1,2}|^2}{\Gamma_{1,1}^2}\right)$$

then

$$\operatorname{var}\left(\widehat{\Gamma}_{1,1}/|\widehat{\Gamma}_{2,1}|\right) = \frac{1}{2}\left((\Gamma_{1,1}\Gamma_{2,2} - |\Gamma_{1,2}|^2)\frac{\Gamma_{1,1}^2}{|\Gamma_{1,2}|^4}\right)$$

Summary

We can rewritten the 2 last expressions using

$$\mu = \frac{|\Gamma_{1,2}|^2}{\Gamma_{1,1}\Gamma_{2,2}}$$

and the averaging on $(2M+1)$:

$$\text{var}\left(|\widehat{\Gamma}_{1,2}|/\widehat{\Gamma}_{2,2}\right) = \frac{1}{2(2M+1)}\frac{\Gamma_{1,1}}{\Gamma_{2,2}}(1-\mu) \tag{7.14}$$

and

$$\text{var}\left(\widehat{\Gamma}_{1,1}/|\widehat{\Gamma}_{2,1}|\right) = \frac{1}{2(2M+1)}\frac{\Gamma_{1,1}}{\Gamma_{2,2}}\frac{1-\mu}{\mu^2} \tag{7.15}$$

Where μ is close to 1, the 2 formulas are equivalent.

Estimator based on eigen-decomposition (uncompleted)

$$\lambda_k^{(3)} = \frac{\Gamma_{1,1} - \Gamma_{2,2} + ((\Gamma_{1,1} - \Gamma_{2,2})^2 + 4|\Gamma_{1,2}|^2)^{1/2}}{2\Gamma_{1,2}^*}$$

5. Some distributions related to the smoothed periodogram

Wiskart's distribution

We consider the sequence of $2M+1$ bivariate random gaussian complex independent variables

$$X_m \overset{\text{i.i.d.}}{\sim} \mathcal{N}_c(0, \Gamma)$$

We let

$$\Gamma = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 e^{j\theta} \\ \rho\sigma_1\sigma_2 e^{-j\theta} & \sigma_2^2 \end{bmatrix}$$

where $\sigma_1 \geq 0$, $\sigma_2 \geq 0$, $\rho \geq 0$ and $\theta \in (0, 2\pi)$, and

$$\widehat{\Gamma} = \frac{1}{2M+1}\sum_{m=1}^{2M+1} X_m X_m^H = \begin{bmatrix} A_1 & Re^{j\Phi} \\ Re^{-j\Phi} & A_2 \end{bmatrix}$$

where $A_1 \geq 0$, $A_2 \geq 0$, $R \geq 0$ and $\Phi \in (0, 2\pi)$. We also have $R^2 \leq A_1 A_2$ thanks to the positivity of $\widehat{\Gamma}$.

We let $K = 2M + 1$. The distribution of (A_1, A_2, R, Φ) writes:

$$
\begin{aligned}
p_{A_1A_2R\Phi}(a_1, a_2, r, \phi) &= \frac{r}{\pi(K-1)!(K-2)!|\Gamma|^K}(a_1a_2 - r^2)^{K-2} && (7.16)\\
&\exp\left\{-\frac{1}{|\Gamma|}\left(\sigma_2^2 a_1 + \sigma_1^2 a_2 - 2r\rho\sigma_1\sigma_2\cos(\theta-\phi)\right)\right\} && (7.17)\\
&\mathbb{1}(a_1 \geq 0) \times \mathbb{1}(a_2 \geq 0) \times \mathbb{1}(\sqrt{a_1a_2} \geq r \geq 0) \times \mathbb{1}(\phi \in (0, 2\pi)) && (7.18)
\end{aligned}
$$

Ratio distributions

We consider at first the ratio $|\widehat{\Gamma}_{1,2}|/\widehat{\Gamma}_{2,2}$, expression (**??**). Integrating expression (7.23) w.r.t. ϕ gives

$$
\begin{aligned}
p_{A_1A_2R}(a_1, a_2, r) &= \frac{2r}{(K-1)!(K-2)!|\Gamma|^K}(a_1a_2 - r^2)^{K-2}\\
&\exp\left\{-\frac{1}{|\Gamma|}\left(\sigma_2^2 a_1 + \sigma_1^2 a_2\right)\right\} I_0(2r\rho\sigma_1\sigma_2/|\Gamma|)
\end{aligned}
$$

It is remarkable to notice that this expression does not depend on θ. We let

$$
\left\{\begin{array}{rcl} Y_1 &=& A_1\\ Y_2 &=& A_2\\ T &=& R/A_2 \end{array}\right. \Leftrightarrow \left\{\begin{array}{rcl} A_1 &=& Y_1\\ A_2 &=& Y_2\\ R &=& TY_2 \end{array}\right. \qquad (7.19)
$$

whose Jacobian is Y_2 which is positive. Because $R^2 \leq A_1A_2$ it follows that $Y_1 - T^2Y_2 \geq 0$. Then

$$
\begin{aligned}
p_{Y_1Y_2T}(y_1, y_2, t) &= \frac{2ty_2^2}{(K-1)!(K-2)!|\Gamma|^K}(y_1y_2 - t^2y_2^2)^{K-2}\mathbb{1}(y_1 - t^2y_2 \geq 0)\\
&\exp\left\{-\frac{1}{|\Gamma|}\left(\sigma_2^2 y_1 + \sigma_1^2 y_2\right)\right\} I_0(2ty_2\rho\sigma_1\sigma_2/|\Gamma|)
\end{aligned}
$$

Integrating w.r.t. y_1 writes

$$
\begin{aligned}
p_{Y_2T}(y_2, t) &= \int_0^{+\infty} \frac{2ty_2^K}{(K-1)!(K-2)!|\Gamma|^K}(y_1 - t^2y_2)^{K-2}\mathbb{1}(y_1 - t^2y_2 \geq 0)\\
&\exp\left\{-\frac{1}{|\Gamma|}\left(\sigma_2^2 y_1 + \sigma_1^2 y_2\right)\right\} I_0(2ty_2\rho\sigma_1\sigma_2/|\Gamma|)dy_1
\end{aligned}
$$

We let $u = y_1 - t^2y_2$, it get

$$
\begin{aligned}
p_{Y_2T}(y_2, t) &= \int_0^{+\infty} \frac{2ty_2^K}{(K-1)!(K-2)!|\Gamma|^K}u^{K-2}\\
&\exp\left\{-\frac{1}{|\Gamma|}\left(\sigma_2^2 u + (t^2\sigma_2^2 + \sigma_1^2)y_2\right)\right\} I_0(2ty_2\rho\sigma_1\sigma_2/|\Gamma|)du
\end{aligned}
$$

and

$$p_{Y_2T}(y_2,t) = \frac{2ty_2^K}{(K-1)!(K-2)!|\Gamma|^K}\exp\left\{-\frac{1}{|\Gamma|}(t^2\sigma_2^2+\sigma_1^2)y_2\right\} I_0(2ty_2\rho\sigma_1\sigma_2/|\Gamma|) \int_0^{+\infty} u^{K-2}\exp\left\{-\frac{1}{|\Gamma|}\sigma_2^2 u\right\} du$$

and

$$p_{Y_2T}(y_2,t) = \frac{2ty_2^K}{(K-1)!(K-2)!|\Gamma|^K}\times(K-2)!\frac{|\Gamma|^{K-1}}{\sigma_2^{2(K-1)}} \exp\left\{-\frac{1}{|\Gamma|}(t^2\sigma_2^2+\sigma_1^2)y_2\right\} I_0(2ty_2\rho\sigma_1\sigma_2/|\Gamma|)$$

$$p_{Y_2T}(y_2,t) = \frac{2ty_2^K}{(K-1)!|\Gamma|\,\sigma_2^{2(K-1)}}\exp\left\{-\frac{1}{|\Gamma|}(t^2\sigma_2^2+\sigma_1^2)y_2\right\} I_0(2ty_2\rho\sigma_1\sigma_2/|\Gamma|)$$

We let $x = 2ty_2\rho\sigma_1\sigma_2/|\Gamma|$ then $dx = 2t\rho\sigma_1\sigma_2 dy_2/|\Gamma|$ and

$$p_T(t) = \frac{1}{(K-1)!}\frac{\lambda}{\rho}\frac{1}{(2t)^K}\frac{1}{\lambda^K}\frac{(1-\rho^2)^K}{\rho^K} \int_0^{+\infty} x^K \exp\left\{-\frac{1}{|\Gamma|}\frac{t^2\sigma_2^2+\sigma_1^2}{2t\rho\sigma_1\sigma_2}x\right\} I_0(x)dx \tag{7.20}$$

rewritten as

$$p_T(t) = \frac{\lambda}{\rho}\int_0^{+\infty}(\zeta x)^K e^{-\xi x} I_0(x)dx \tag{7.21}$$

where

$$\zeta = \frac{1-\rho^2}{2t\rho\lambda}\frac{1}{((K-1)!)^{1/K}}$$

$$\xi = \frac{1+t^2\lambda^2}{2t\rho\lambda}$$

The last integration w.r.t. x has been done numerically. Also we remark that

$$\zeta^K = \exp\left(K\log(\mu) - \sum_{k=1}^{K-1}\log(k)\right)$$

where

$$\mu = \frac{1-\rho^2}{2t\rho\lambda}$$

which is more accurate to use than factorial function. Also it is more accurate to use $\tilde{I}_0(x) = e^{-x}I_0(x)$ which is usually proposed in several numerical toolboxes.

We consider the ratio $\widehat{\Gamma}_{1,1}/|\widehat{\Gamma}_{1,2}|$, expression (**??**).

For the ratio $V = A_1/R$, we compute at first the distribution of $W = R/A_2$ which can be obtained using the expression (7.21) reversing the role of A_1 and A_2. Then using the transformation $V = 1/W$ whose the Jacobian is $1/V^2$ we get

$$p_V(v) = \frac{\lambda}{\rho} v^{-2} \int_0^{+\infty} (\zeta x)^K e^{-\xi x} I_0(x) dx \tag{7.22}$$

where

$$\zeta = \frac{v(1-\rho^2)}{2\rho\lambda} \frac{1}{((K-1)!)^{1/K}}$$
$$\xi = \frac{v^2+\lambda^2}{2v\rho\lambda}$$

Phase distribution (not completed)

We consider the sequence of $2M+1$ bivariate random gaussian complex independent variables

$$X_m \overset{\text{i.i.d.}}{\sim} \mathcal{N}_c(0, \Gamma)$$

We let

$$\Gamma = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 e^{j\theta} \\ \rho\sigma_1\sigma_2 e^{-j\theta} & \sigma_2^2 \end{bmatrix}$$

where $\sigma_1 \geq 0$, $\sigma_2 \geq 0$, $\rho \geq 0$ and $\theta \in (0, 2\pi)$, and

$$\widehat{\Gamma} = \frac{1}{2M+1} \sum_{m=1}^{2M+1} X_m X_m^H = \begin{bmatrix} A_1 & Re^{j\Phi} \\ Re^{-j\Phi} & A_2 \end{bmatrix}$$

where $A_1 \geq 0$, $A_2 \geq 0$, $R \geq 0$ and $\Phi \in (0, 2\pi)$. We also have $R^2 \leq A_1 A_2$ thanks to the positivity of $\widehat{\Gamma}$.

We let $K = 2M+1$. The distribution of (A_1, A_2, R, Φ) writes:

$$\begin{aligned} p_{A_1A_2R\Phi}(a_1, a_2, r, \phi) &= \frac{r(a_1a_2 - r^2)^{K-2}}{\pi(K-1)!(K-2)!|\Gamma|^K} \\ &\quad \exp\left\{-\frac{1}{|\Gamma|}\left(\sigma_2^2 a_1 + \sigma_1^2 a_2 - 2r\rho\sigma_1\sigma_2\cos(\theta-\phi)\right)\right\} \\ &\quad \mathbb{1}(a_1 \geq 0) \times \mathbb{1}(a_2 \geq 0) \times \mathbb{1}(\sqrt{a_1a_2} \geq r \geq 0) \times \mathbb{1}(\phi \in (0, 2\pi)) \end{aligned} \tag{7.23}$$

We have

$$\int_0^{\sqrt{a_1a_2}} P(r) e^{-\mu r} dr = \left[\frac{-1}{\mu} e^{-\mu r} \sum_{s=0}^{2K-3} \frac{1}{\mu^s} P^{(s)}(r)\right]_0^{\sqrt{a_1a_2}}$$

Here $\mu = 2\rho\sigma_1\sigma_2\cos(\theta-\phi)/|\Gamma|$ and

$$\begin{aligned}
P(r) &= r(a_1a_2 - r^2)^{K-2} = \sum_{q=0}^{K-2}(-1)^q(a_1a_2)^{K-2-q}r^{2q+1}C_{K-2}^q \\
P^{(1)}(r) &= (2q+1)\sum_{q=1}^{K-2}(-1)^q(a_1a_2)^{K-2-q}r^{2q}C_{K-2}^q \\
P^{(2)}(r) &= (2q+1)2q\sum_{q=1}^{K-2}(-1)^q(a_1a_2)^{K-2-q}r^{2q-1}C_{K-2}^q \\
&\vdots \\
P^{(2K-4)}(r) &= -(K-2)(a_1a_2)^{K-3}r^3 \\
P^{(2K-3)}(r) &= (-1)^{K-2}
\end{aligned}$$

$$\int_0^{+\infty} a_1^{m+1/2} e^{-\alpha_1 a_1} da_1$$

and

$$\int_0^{+\infty} a_1^{m} e^{-\alpha_1 a_1} da_1$$

We use

$$\int_0^{+\infty} x^{t-1} e^{-\mu x} dx = \Gamma(t) \frac{1}{\mu^t}$$

...

(uncompleted ...)

6. MSC distribution

For the MSC defined in (6.2) an estimator is:

$$\widehat{\mathrm{MSC}}_k = \frac{|\widehat{\Gamma}_{k,1,2}|^2}{\widehat{\Gamma}_{k,1,1}\widehat{\Gamma}_{k,2,2}} \tag{7.24}$$

Its statistics have been given in [**?**]. The density probability writes

$$\begin{aligned} p(c, \mathrm{MSC}) &= (N_d - 1) \times \left[\frac{(1-c)(1-\mathrm{MSC})}{(1-c\times\mathrm{MSC})^2}\right]^{N_d} \times \frac{1-c\times\mathrm{MSC}}{(1-c)^2} \\ &\quad {}_2F_1(1-N_d, 1-N_d; 1; c\times\mathrm{MSC}) \end{aligned} \tag{7.25}$$

ant the cumulative function writes:

$$\begin{aligned} F(c, \mathrm{MSC}) &= \mathbb{P}\left(\widehat{\mathrm{MSC}}_k \leq c; \mathrm{MSC}\right) \\ &= c \times \left(\frac{1-\mathrm{MSC}}{1-c\times\mathrm{MSC}}\right)^{N_d} \times \\ &\quad \sum_{k=0}^{N_d-2} \left(\frac{1-c}{1-c\times\mathrm{MSC}}\right)^{k} {}_2F_1(-k, 1-N_d; 1; c\times\mathrm{MSC}) \end{aligned} \tag{7.26}$$

where ${}_2F_1$ is the hypergeometric function.

The integer N_d is either the size of the window for smoothed periodogram or the number of blocks for the averaged peridograms (Welch method). We will call it the degree of smoothing, in short d.o.s.

Fortunately the expression (7.26) is easy to compute using that

$${}_2F_1(-k, 1-N_d; 1; c\times\mathrm{MSC}) = \sum_{m=0}^{k} T_m$$

where $T_0 = 1$ and the recursion:

$$T_m = T_{m-1} \frac{(m-1-k)(m-N_d)\times\mathrm{MSC}}{m^2}\, c \tag{7.27}$$

To show the accuracy of this expression we did a simple simulation. We generate a sequence of the bivariate process

$$x_n = Gw_n$$

where w_n is a bivariate process with 0 mean and covariance I_2. Therefore x_n is a bivariate process with 0 mean and covariance $\Gamma = GG^H$. Therefore the MSC of the 2 components is

$$\frac{|\Gamma_{1,2}|^2}{\Gamma_{1,1}\Gamma_{2,2}}$$

Cumulative function has been performed from Monte-Carlo simulation and compared to the expression (7.26). The results are reported figure 7.1. The theoretical values are in very good agreement with those obtained from Monte-Carlo simulation.

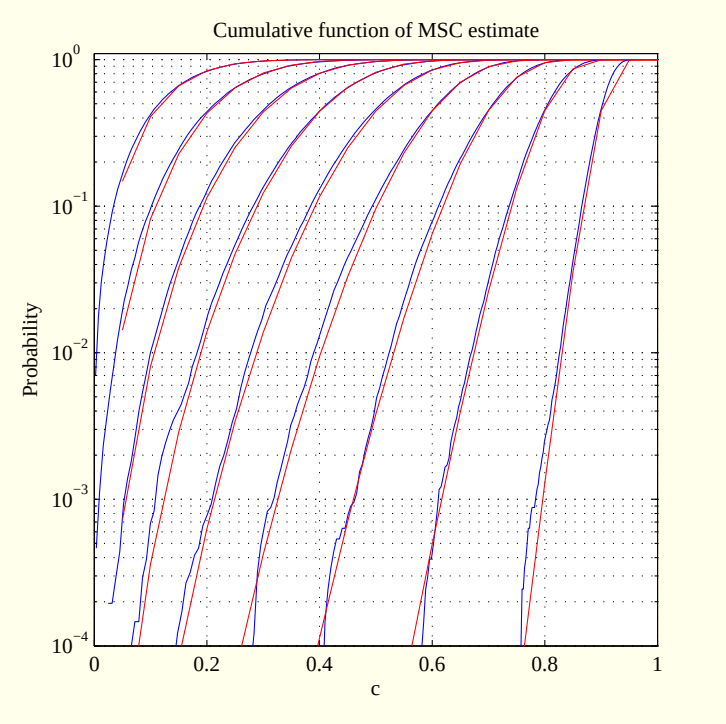

Figure 7.1: *Cumulative function of the MSC estimate given by expression* (7.24)*. The true values of the MSC are from* 0.1 *to* 0.9 *by step* 0.1 *from the left to the right. In blue, the values obtained by simulation on* 40960 *samples. In red the theoretical expression* (7.26)*.*

7. MSC detection

The analytical expressions of the probability density functions (7.25) and the cumulative function (7.26) of the MSC estimator. Therefore we have a way to perform numerically the inverse of the cumulative function and then determine a detection threshold. Indeed, a simple dichotomy can be applied because of the monotony of the cumulative function. Then we can derive a confidence interval and attest on the MSC value.

We consider the statistical model

$$\{p(\mu, \mathrm{MSC}_0), \quad \mathrm{MSC}_0 \in (0,1)\}$$

where p is the probability density function of $\widehat{\mathrm{MSC}}$ for a given value of the d.o.s. N_d. Its expression is given in equation (7.25). We want to test the hypothesis $H_0 = \{\mathrm{MSC}_0 > C\}$ against the counter hypothesis $H_1 = \{\mathrm{MSC}_0 \leq C\}$ with $C \in (0,1)$. The GLRT writes

$$\Lambda(\mu) = \frac{\max_{\mathrm{MSC}_0 \in (0,1)} p(\mu, \mathrm{MSC}_0)}{\max_{\mathrm{MSC}_0 \in H_0} p(\mu, \mathrm{MSC}_0)}$$

The property that $\Lambda(\mu)$ is a monotonic increasing function of c is not proven but we admit that $\widehat{\mathrm{MSC}}$ is a good function of test. In this case we can fix the threshold η such that

$$\mathbb{P}\left(\widehat{\mathrm{MSC}} \leq \eta; \mathrm{MSC}_0\right) = 0.95$$

Figure 7.2 we have reported the η values at 95% for $\mathrm{MSC}_0 \in (0,1)$ for $N_d = 20, 30, 40, 50$. It is worth noticing that the curves is largely over the first bissectrix even for large N_d. For example for $\mathrm{MSC} = 0.8$ the threshold is given on the zoom of the figure for different values of N_d.

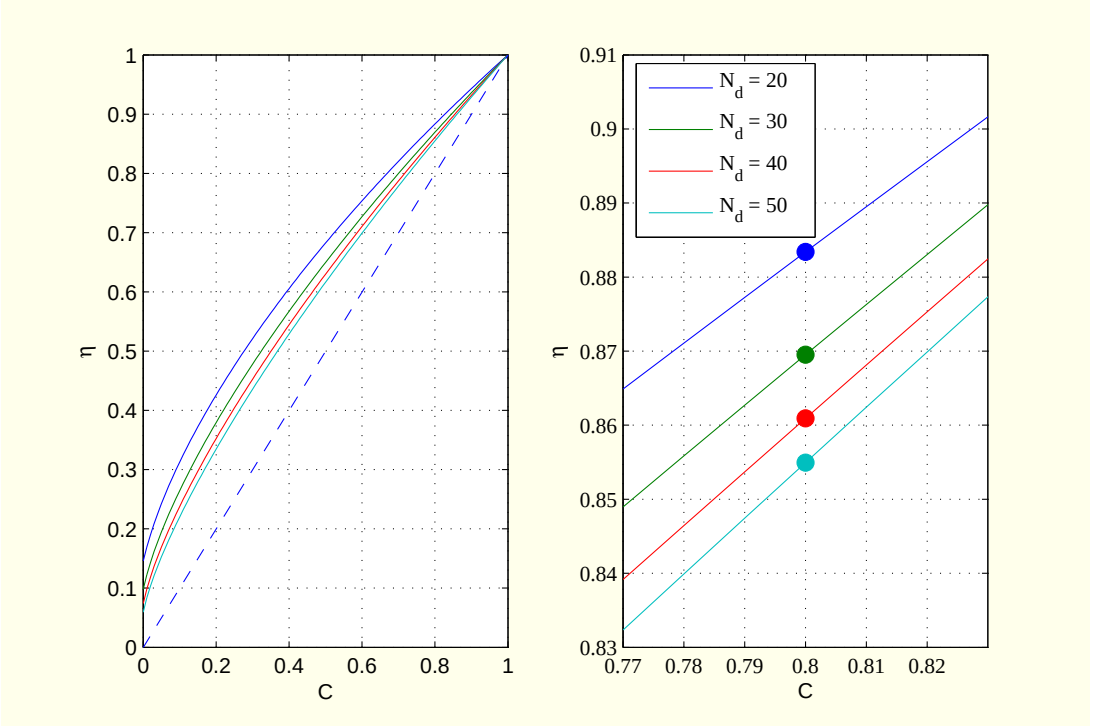

Figure 7.2: *Threshold at* 95% *and* $N_d = 20, 30, 40, 50$ *as a function of the value on test.*

A simulation has been done with a white SOI leading to a level of MSC of 0.7. We have reported figure 7.3 the numerical estimates of the MSC and the threshold at 95%. We verified that only 5% are over the threshold. That means that, if we observe a value of MSC greater than around 0.8, the probability that the MSC is under 0.7 is less than 5%. That is a very conservative attitude.

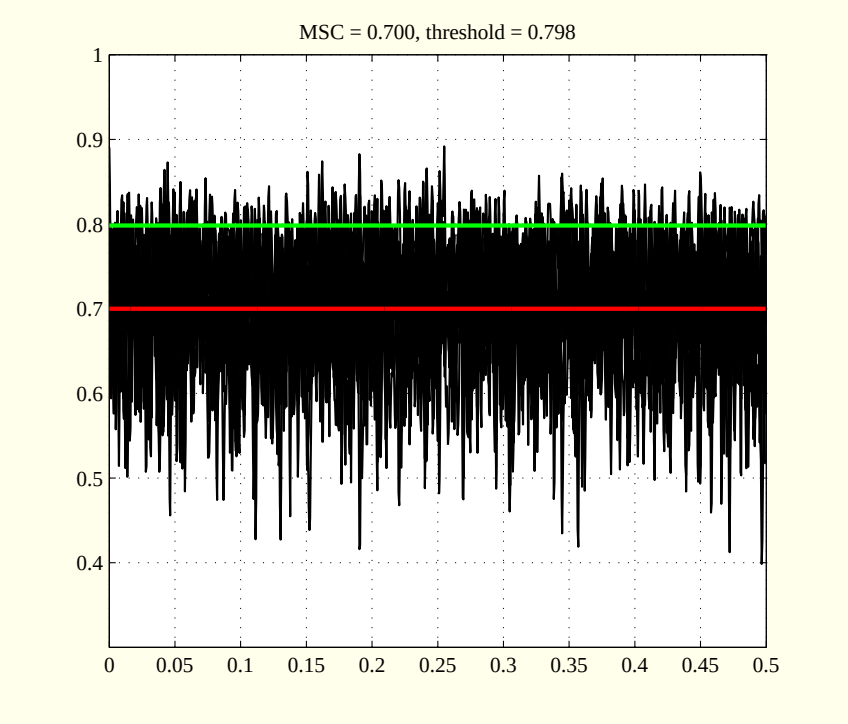

Figure 7.3: *Monte-Carlo simulations on 1000 runs. In red the true value of the MSC. In green, the threshold derived from the analytical expression in such a way that the probability to be over this value is less than* 5%*. In black the Monte-Carlo estimates. It has been verified that* 5% *are over the threshold.*

Chapter 8

LOC

Let us consider the equation:

$$Y_t = g_t \star X_t$$

where g_t is the M-ary impulse response of the single input multiple output (SIMO) linear filter and X_t is a scalar WSS process, with zero mean and spectral matrix (of size 1) $\gamma_x(f)$. It is easy to show that the spectral matrix of Y_t writes

$$\Sigma_y(f) = \gamma_x(f) G(f) G^H(f)$$

where $G(f)$ is the discrete time Fourier transform of g_t. We notice that $\Sigma_y(f)$ is a matrix of rank 1. A particular case is a pure delay propagation filter as for example in the planar propagation where the vector entry of $G(f)$ writes:

$$G_m(f) = e^{-2j\pi f r_m^T \theta}$$

where r_m is the 3D location of the sensor m and θ the wave number of the planar wave. It follows that the coherence matrix of the M-ary observation has the following entry expression

$$\begin{aligned} C_{y,m\ell}(f) &= e^{-2j\pi f (r_m - r_\ell)^T \theta} \\ &= \int_{\mathbb{R}^3} e^{-2j\pi f (r_m - r_\ell)^T u} \delta_\theta(u) du \end{aligned} \tag{8.1}$$

where du is the infinitesimal Lebesgue measure in $\mathbb{R}^3$ and δ_θ the Dirac's distribution located in θ. The expression (8.1) can be seen as an expectation of $e^{-2j\pi f (r_m - r_\ell)^T \theta}$ assuming that θ is deterministic.

Coherence model

In a pioneer work Mack and Flinn [**?**] propose to model the loss of coherence by considering that the azimuth, the elevation and velocity are uncertain. More specifically, authors in [**?**] assume that the 3D vector $\mu = (a, e, c)$ where a denotes the azimuth, e the elevation and c the sound velocity writes:

$$\mu = \mu_0 + \nu \tag{8.2}$$

where $\mu_0 = (a_0, e_0, c_0)$ is a deterministic value and ν a zero-mean Gaussian random vector of dimension 3, whose covariance is denoted Σ_μ. We know that μ is related to the slowness vector k by the one-to-one mapping

$$f:\ \mu = \left\{ \begin{array}{ll} a = \arg(k_2 + jk_1) & a \in (0, 2\pi) \\ e = \arg\sin(ck_3) & e \in (-\pi/2, \pi/2) \\ v = (k_1^2 + k_2^2 + k_3^2)^{1/2} & v \in \mathbb{R}^+ \end{array} \right. \iff$$

$$k = \left\{ \begin{array}{ll} k_1 = -v^{-1}\sin(a)\cos(e) \\ k_2 = v^{-1}\cos(a)\cos(e) \\ k_3 = v^{-1}\sin(e) \end{array} \right. \in \mathbb{R}^3$$

Using a first order Taylor's expansion, it follows that $k \approx f(\mu_0) + J(\mu_0)(\mu - \mu_0)$ where the Jacobian writes

$$\begin{array}{rcl} J(\mu_0) & = & v_0^{-1} \times \tilde{J}(\mu_0) \\ \text{with} \qquad \tilde{J}(\mu_0) & = & \begin{bmatrix} -\cos(a_0)\cos(e_0) & \sin(a_0)\sin(e_0) & v_0^{-1}\sin(a_0)\cos(e_0) \\ -\sin(a_0)\cos(e_0) & -\cos(a_0)\sin(e_0) & -v_0^{-1}\cos(a_0)\cos(e_0) \\ 0 & \cos(e_0) & -v_0^{-1}\sin(e_0) \end{bmatrix} \end{array}$$

Therefore from (8.2) we derive $k \approx k_0 + \varepsilon$ where $\varepsilon = J(\mu_0)\,\nu$ appears as a zero-mean Gaussian random vector whose the covariance is:

$$\Sigma_k \approx v_0^{-2} \underbrace{\tilde{J}(\mu_0)\Sigma_\mu \tilde{J}^T(\mu_0)}_{\tilde{\Sigma}_k} \tag{8.3}$$

Hence the coherence matrix entry of the model with random wavefront writes:

$$C_{m,\ell}(f) \ = \ e^{-2j\pi f (r_m - r_\ell)^T k_0} \Phi_\epsilon(2\pi f(r_m - r_\ell))$$

where $\Phi_\epsilon : u \in \mathbb{R}^3 \mapsto \mathbb{E}\left[e^{ju^T\varepsilon}\right] \in \mathbb{C}$ is the characteristic function of ε. Since ε is Gaussian distributed with zero-mean and covariance matrix Σ_k, hence we have

$$C_{m,\ell}(f) = \underbrace{e^{-2j\pi f (r_m - r_\ell)^T k_0}}_{\text{pure delay}} \underbrace{e^{-2\pi^2 (f/v_0)^2 (r_m - r_\ell)^T \tilde{\Sigma}_k (r_m - r_\ell)}}_{\text{LOC}} \tag{8.4}$$

We let $\zeta = v_0/f$ that can be interpreted as a "wavelength". Using the MSC definition (4.7) we derive the following expression:

$$\text{MSC}_{km}(f) = e^{-\frac{4\pi^2}{\zeta^2}(r_m - r_\ell)^T \tilde{\Sigma}_k (r_m - r_\ell)} \tag{8.5}$$

It follows that

- when ζ is large w.r.t. the sensor inter-distances, the matrix $C \approx I$, hence the components of $x(t)$ appear as non spatially coherent,
- when ζ is small w.r.t. the sensor inter-distances, the matrix $C \approx \mathbb{1}\mathbb{1}^T$ which is a projector, hence the components of $x(t)$ appear as spatially coherent.

Chapter 9

Remark on deterministic representation

Remark on a deterministic approach

We have seen that, in the absence of noises, we can use the DFT of each trajectory and perform the ratio without any averaging. But we have also to express a deterministic property which can be the equivalent of the MSC condtion. For that we can work frequency by frequency and therefore omit the dependence in frequency. We denote $X_{u,1}, \ldots X_{u,M}$ the DFT of the signal of the SUT and $X_{r,1}, \ldots X_{r,M}$ the DFT of the signal of the SREF.

Now to define in the determistic case a notion equivalent to the MSC, we can say that the values of the ratio must be almost constant along the M windows. That writes: it exists 2 coefficients not depending on m, verifying $\alpha_u^2 + \alpha_r^2 = 1$ and s.t.

$$\alpha_u X_{u,m} + \alpha_r X_{r,m} \approx 0 \tag{9.1}$$

With obvious matricial notations, we have:

$$X\alpha = \epsilon$$

$$X = \begin{bmatrix} X_{u,1} & X_{r,1} \\ \vdots & \\ X_{u,M} & X_{r,M} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \alpha_u \\ \vdots \\ \alpha_r \end{bmatrix}$$

The minimization of the norm of ϵ under the constraint $\|\alpha\|^2 = 1$ leads to take for α the eigenvector associated to the lowest eigenvalue of denoted $\lambda_{\min}$. The matrix $M^{-1}X^TX$ writes:

$$\Gamma = \frac{1}{M} \begin{bmatrix} \sum_{m=1}^{M} |X_{u,m}|^2 & \sum_{m=1}^{M} X_{u,m} X_{r,m}^* \\ \sum_{m=1}^{M} X_{u,m}^* X_{r,m} & \sum_{m=1}^{M} |X_{r,m}|^2 \end{bmatrix}$$

The minimum value is then given by:

$$\epsilon_{\min} = \lambda_{\min}$$

Lower $\lambda_{\min}$ better the approximation (9.1). On the other hand, lower $\lambda_{\min}$ lower the determinant of Γ which writes:

$$\begin{aligned}\Delta &= M^{-2}\sum_{m=1}^{M}|X_{u,m}|^2\sum_{m=1}^{M}|X_{r,m}|^2 - M^{-2}\left|\sum_{m=1}^{M}X_{u,m}X_{r,m}^*\right|^2 \\ &= M^{-2}\sum_{m=1}^{M}|X_{u,m}|^2\sum_{m=1}^{M}|X_{r,m}|^2\,(1-\mathrm{MSC})\end{aligned}$$

In conclusion nearer the MSC to 1, better the approximation (9.1).

Part III
Library

Chapter 10

Programs

1. RMSE as function of coherence

```
%============================================================
% estimHanalysis.m
%============================================================
% Determine the level of MSC useful to reach a given level of
% accuracy for the ratio HUT/HREF.
% Simulation is conducted on synthetical or semi-synthetical
% signals.
%============================================================
%
% used function: fbankanalysis.m
% with no filterbank
%
% We draw randomly and use the H estimate
% Models of signals
% xREF = BGfield + sqrt(sigma2REF)*wREF;
% xUT  = BGfield + sqrt(sigma2UT)*wUT;
% yREF = filter(bREF,aREF,xREF);
% yUT  = filter(bUT,aUT,xUT);
%
% Results have to be compared to these
% of program STATSONH11H12.m, plotted with allprintstatsonHest.m
%============================================================
% N is divided in segments of length equal to Tfft_sec*Fs_Hz
% with Tfft_sec=300.
% To form the data block on which the spectrum is performed,
% we use M consecutive segments.
% The simulation is iterated in such a way the duration of the
% total observation is 5 hours. This value is obvioulsy not crucial.
% It is just to perform the RMSE on enough simulations.
% The RMSE is computed by averaging on the full frequency band.
%
%============================================================
clear
addpath  ../../../ZZtoolbox/
synth            = 1;
Fs_Hz   = 20;
FFT_sec = 300;
Lfft    = fix(FFT_sec*Fs_Hz);
duration_hour    = 5;
N           = duration_hour*3600*Fs_Hz;
frqs           = (0:Lfft-1)*Fs_Hz/Lfft;
selectbandave_Hz = 10;
idbband          = find(frqs<selectbandave_Hz,1,'last');
bandave          = 1:idbband;


%======== filter REF ==========
FREF_Hz      = 0;
rhoREF       = 0.7;
costhetaREF  = (1+rhoREF*rhoREF)*cos(2*pi*FREF_Hz/Fs_Hz)/2/rhoREF;
bREF         = 0.5*[1+rhoREF*rhoREF;-4*rhoREF*costhetaREF;1+rhoREF*rhoREF];
aREF         = [1 -2*rhoREF*costhetaREF rhoREF*rhoREF]';
HREF         = fft(bREF,Lfft) ./ fft(aREF,Lfft);
HREF         = HREF(:);
%======== filter UT ==========
FUT_Hz       = 0;
rhoUT        = 0.7;
costhetaUT   = (1+rhoUT*rhoUT)*cos(2*pi*FUT_Hz/Fs_Hz)/2/rhoUT;
bUT          = 0.5*[1+rhoUT*rhoUT; -4*rhoUT*costhetaUT; 1+rhoUT*rhoUT];
aUT          = [1 -2*rhoUT*costhetaUT rhoUT*rhoUT]';
HUT          = fft(bUT,Lfft) ./ fft(aUT,Lfft);
HUT          = HUT(:);

% randn('seed',0)
if synth
    % for synthetical signals
    % the BGfield is white
    randnN = randn(N,1);
    randnN = randnN/std(randnN);
else
    % the BGfield is drawn
    % from the database
    directorydata = '../../../../DATA_IS/I26/';
    % directorydata = './';
    filesmat = dir([directorydata '*.mat']);
    nbmats   = length(filesmat);
    ifile    = 2;
    commandload = sprintf('load %s%s',directorydata,filesmat(ifile).name);
    eval(commandload)
    id1=fix(rand*1e6);
    data_pa = records{2}.data(id1+(1:N));
```

```
    data_pa = data_pa-mean(data_pa);
    randnN = data_pa/std(data_pa);
end
% theoretically gamma has no effect
gamma          = 5;
BGfield        = sqrt(gamma)*randnN;

overlapFFT     = 0.5;
overlapAVE     = 0;

sqrtLfft       = sqrt(Lfft);
frqs           = (0:Lfft-1)*Fs_Hz/Lfft;

listM          = [5 10];
LM             = length(listM);
listMSC        = 0.8:0.02:0.99;
LMSC           = length(listMSC);
% root mean square error on the ratio SUU/abs(SUR)
RMSEUUUR       = zeros(LMSC,LM);
RMSEAB2        = zeros(LMSC,LM);

%==============================
% Because airinlet is given the MSC implies
% the level of white noise (not true for non white)
%==============================
for iMSC=1:LMSC
    MSCtrue   = listMSC(iMSC);
    %==============================
    % airinlet induces lower noise on UT than on REF
    % airinlet value is of order of magnitude of the
    % number of airinlets, typically a few tens
    % but has little consequence because we fix the MSC
    % by taking into  account this number.
    %==============================
    airinlet    = 48;
    sigma2UT    = max(roots([airinlet ...
        (airinlet+1)*gamma ...
        gamma*gamma*(1-1  ./ MSCtrue)]));
    sigma2REF   = airinlet*sigma2UT;
    % gamma*gamma/(gamma+sigma2UT)/(gamma+sigma2REF)

    wREF  = randn(N,1);
    wUT   = randn(N,1);

    xREF  = BGfield  + sqrt(sigma2REF)*wREF;
    xUT   = BGfield  + sqrt(sigma2UT)*wUT;

    yREF    = filter(bREF,aREF,xREF);
    yUT     = filter(bUT,aUT,xUT);
    % U = 1, REF =2
    signals = [yUT, yREF];

    for im=1:LM
        M  = listM(im);
        P=1;
        filtercharacteristics(P).designname         = '';
        filtercharacteristics(P).Norder             = 0;
        filtercharacteristics(P).Wlow_Hz            = 0.0001;
        filtercharacteristics(P).Whigh_Hz           = Fs_Hz;
        filtercharacteristics(P).SCPperiod_sec      = FFT_sec*M;
        filtercharacteristics(P).windowshape        = 'hann';
        filtercharacteristics(P).overlapDFT         = 0.5;
        filtercharacteristics(P).overlapSCP         = 0;
        filtercharacteristics(P).ratioDFT2SCP       = M;

        [SUTs, filteredsignals, allfrqsFFT_Hz, alltimes_sec, filterbank] = ...
            fbankanalysis(signals, ...
            filtercharacteristics, ...
            Fs_Hz,...
            0.5);

        nbruns = size(SUTs.estimRsup.tabmod,2);
        rmse_integrated = zeros(nbruns,1);
        for irun=1:nbruns
            HestUUUR_irun = SUTs.estimRsup.tabmod(:,irun) .* abs(HREF);
            mseHestUUUR_irun = (HestUUUR_irun(bandave)-abs(HUT(bandave))) .^2;
            mseHestUUUR_irun_rel = mseHestUUUR_irun ./ abs(HUT(bandave));
            rmse_integrated(irun)=sqrt(nanmean(mseHestUUUR_irun));
        end
        RMSEUUUR(iMSC,im) = nanmean(rmse_integrated);
    end
end
%%
figure(4)
leg = cell(LM,1);
for it=1:LM
    leg{it}=sprintf('M = %3i',listM(it));
end
plot(listMSC,RMSEUUUR,'x-')
% hold on
% plot(listMSC,RMSEAB2,'o-')
set(gca,'fontsize',12,'fontname','times')
grid on
hl=legend(leg);
set(hl,'fonts',12,'fontn','times')
xlabel('coherence','fontsize',12,'fontname','times')
ylabel('RMSE on SUT','fontsize',12,'fontname','times')
hold on
plot([0.8 1],0.05*ones(2,1),'--')
hold off
set(gca,'ylim',[0 0.2])
%==========================================
%%
HorizontalSize = 12;
VerticalSize   = 8;
set(gcf,'units','centimeters');
set(gcf,'paperunits','centimeters');
set(gcf,'PaperType','a4');
set(gcf,'position',[0 5 HorizontalSize VerticalSize]);
set(gcf,'paperposition',[0 0 HorizontalSize VerticalSize]);

set(gcf,'color', [1,1,0.92]);%0.7*ones(3,1))
set(gcf, 'InvertHardCopy', 'off');

% figure(4); print -depsc -loose ../../textes/6distConjointHMSC/figures/allHest
```

```
% print -depsc -loose ../../textes/6distConjointHMSC/figures/allHest
% !epstopdf ../../textes/6distConjointHMSC/figures/allHest.eps
% !rm ../../textes/6distConjointHMSC/figures/allHest.eps

return
%%
% save signals signals Fs_Hz
% return
figure(4)
subplot(411)
semilogx(frqs,10*log10(sRRf))
set(gca,'xlim',[0 6])
set(gca,'ylim',[-20 20])
ylabel('dB')
grid on
%==
subplot(412)
semilogx(frqs,10*log10(sUUf))
set(gca,'xlim',[0 6])
set(gca,'ylim',[-20 20])
ylabel('dB')
grid on
%==
subplot(413)
semilogx(frqs,20*log10(mean(HestUUUR,2)))
hold on
semilogx(frqs,20*log10(abs(HUT)),'r')
hold off
set(gca,'ylim',20*log10(abs(HUT(1)))+[-15,5])
% set(gca,'ylim',[-20 40])
set(gca,'xlim',[0 6])
ylabel('dB')
grid on
%==
subplot(414)
semilogx(frqs,MSC)
set(gca,'xlim',[0 6])
grid on


%
HorizontalSize = 25;
VerticalSize   = 18;
set(gcf,'units','centimeters');
set(gcf,'paperunits','centimeters');
set(gcf,'PaperType','a4');
% set(gcf,'position',[0 5 HorizontalSize VerticalSize]);
set(gcf,'paperposition',[0 0 HorizontalSize VerticalSize]);
set(gca,'fontn','times','fonts',10)

set(gcf,'color', [1,1,0.92]);
set(gcf, 'InvertHardCopy', 'off');
```

2. Test of coherence

```
%=================================================
% SigniLevelTestCoherence.m
%=================================================
% Performances of test on the hypothesis
% H0 = { MCS > cintest} with M d.o.f
% The significance level is denoted signifiancelevel
% Used function: cumulFunctionMSC.m
%=================================================
clear
addpath  /Users/maurice/etudes/ctbto/allJOBs2015/myjob/1TaskOnSensors/textes/6distConjointHMSC/fullprocess/ZZtoolbox/
signifiancelevel = 0.9;
cintest = 0.97; NE=100;
E = linspace((0.9),(0.999),NE)';
allM = 5:2:10;
LM = length(allM);
P = zeros(NE,LM);
thres = zeros(LM,1);
leg=[];
for id=1:LM
    N = allM(id);
    P(:,id) = cumulFunctionMSC(E,cintest,N);
    thres(id) = invcumulFunctionMSC(signifiancelevel,cintest,N);
    leg = [leg;{sprintf('M = %2i, th = %4.3f',N,thres(id))}];
end
plot(E,P,'.-')
hold on
plot([0, 1],[signifiancelevel * [1 1]])
ht = plot(thres,signifiancelevel*ones(LM,1),'o');
set(ht,'markerfacec','b')
hold off
grid
set(gca,'ylim',[0.8 1])
set(gca,'xlim',[0.97 1])

ht=legend(leg);
set(ht,'fontsize',12)
ht=title(sprintf('H_0 = %s MSC>%4.2f %s at level %4.2f','\{',cintest,'\}',signifiancelevel));
set(ht,'fontname','times','fontsize',12)

HorizontalSize = 12;
VerticalSize   = 8;
set(gcf,'units','centimeters');
set(gcf,'paperunits','centimeters');
set(gcf,'PaperType','a4');
set(gcf,'position',[0 5 HorizontalSize VerticalSize]);
set(gcf,'paperposition',[0 0 HorizontalSize VerticalSize]);
set(gca,'fontname','times','fontsize',12)

set(gcf,'color', [1,1,0.92]);
set(gcf, 'InvertHardCopy', 'off');
%

% figure(4); print -depsc -loose  ../../textes/6distConjointHMSC/figures/MSCtestthreshold

% figure(4)
% print -dpdf -loose ../../textes/6distConjointHMSC/figures/MSCtestthreshold
% print -depsc -loose ../../textes/6distConjointHMSC/figures/allHest
% !epstopdf ../../textes/6distConjointHMSC/figures/allHest.eps
% !rm ../../textes/6distConjointHMSC/figures/allHest.eps
```

3. Statistical distribution of the ratios

```
%======================================================================
%   CIHestimate.m
%======================================================================
% We compar theoretical and simulated distributions
% the thoeretical values are performed in the function
%                 statsRatiosHbis.m (see toolbox)
%======================================================================
% statistics by drawing vectors under a given spectral matrix GAMMA
% WARNING: that means that the frequency value is fixed and hence can be
% omitted.
%===
% Comparison of Monte-Carlo simulation results and
% theoretical values for the two ratios used to estimate
% HU from HR. The results do depend mainly on the
% true ratio denoted
%
%       HUonHR_k = HU_k / HR_k
%
% which is a complex number.
% To obtain consistent estimates we mix 2M+1 values, either
% by smoothing the periodogram or by averaging several periodograms
% (Welch's approach).
% The statitstics of the modulus of HUonHR_k are given in the report.
% For the phase we used an approximative formula based on delta-method
% and gaussian shape.
%
%%
% Important remark:
% Below we also compute the conditional bias
% w.r.t. a given MSC threshold. Therefore when we reduce "threshold2"
% the bias decreases but that does not simulate what we have in
% real situation. Indeed in this simulaiton the true MSC stays equal
% to "MSC_true"
% Therefore we are not in the case where we threshold a real MSC
% which does change.
%======================================================================

% Reference =1, Under test=2
%       REF UxR
%       UxR UT
%======================================================================
%
clear
addpath  /Users/maurice/etudes/ctbto/allJOBs2015/myjob/1TaskOnSensors/textes/6distConjointHMSC/fullprocess/ZZtoolbox/

%=====================================================
% R=1, U=2
% we loock for statistics of ratioUUonUR, ratioURonRR,
% then HU = HR x ratioUUonUR OR HU = HR x ratioURonRR
%=====================================================

% global factor for the spectral matrices(no effect)
gammafactor            = 5;
% true HUonHR
absHUonHR              = 1;
phase_HUminusHR_degree = 20;
argHUonHR_rad          = phase_HUminusHR_degree*pi/180;
HRonHU                 = absHUonHR*exp(1j*argHUonHR_rad);
twoMminus1              = 9;
% noise on the sensor under test is currently lower
% than noise on the reference sensor, ie
%      sigmaR = g x sigmaU, with g about 10
% C =1/(1+sigmaU2)(1+sigmaR2)
%
% (1+sigmaU2)(1+g^2 sigmaU2)=1/C
% [1-1/C 1+g^2 g^2]
%%
MSC_true               = 0.96;
Ninlets                = 96;
noiseratioEff          = sqrt(Ninlets);
sigmaU2                = roots([noiseratioEff^2  1+noiseratioEff^2 1-1/MSC_true]);
sigmaU2                = max(sigmaU2);
sigmaU                 = sqrt(sigmaU2);
sigmaR                 = sigmaU*noiseratioEff;
sigmaR2                = sigmaR ^2;
%%
%=========================
%====== spectralmatrix
spectralmatrix         = gammafactor * ...
    [(1+sigmaR2)/absHUonHR  exp(-1j*argHUonHR_rad); ...
    exp(1j*argHUonHR_rad) (1+sigmaU2)*absHUonHR];
%=========================
% for histograms
nbbins = 500;

absHR2 = gammafactor/absHUonHR;
HR     = sqrt(absHR2);
% Monte-Carlo simulations
Lruns  = 1000000;
sqrtG  = sqrtm(spectralmatrix);
SUU_MC = zeros(Lruns,1);
SRR_MC = zeros(Lruns,1);
SUR_MC = zeros(Lruns,1);

for in = 1:twoMminus1
    W  = (randn(Lruns,2)+1j*randn(Lruns,2))/sqrt(2);
    X        = W * sqrtG;
    % REF is on RR=1, and UT is on UU=2
    SRR_MC    = SRR_MC + (X(:,1) .* conj(X(:,1)));
    SUU_MC    = SUU_MC + (X(:,2) .* conj(X(:,2)));
    SUR_MC    = SUR_MC + (X(:,1) .* conj(X(:,2)));
end
SUU_MC           = SUU_MC/twoMminus1;
SRR_MC           = SRR_MC/twoMminus1;
SUR_MC           = SUR_MC/twoMminus1;
absSUR_MC        = abs(SUR_MC);

Rsup_MC          = SUU_MC ./ SUR_MC;
realRsup_MC      = real(Rsup_MC);
imagRsup_MC      = imag(Rsup_MC);
phaseRsup        = -atan2(imagRsup_MC,realRsup_MC);
phaseRsup_degree = phaseRsup*180/pi;
```

```
Rinf_MC          = conj(SUR_MC) ./ SRR_MC;
realRinf_MC      = real(Rinf_MC);
imagRinf_MC      = imag(Rinf_MC);
phaseRinf        = -atan2(imagRinf_MC,realRinf_MC);
phaseRinf_degree = phaseRsup*180/pi;

%================================================
absRsup_MC = abs(Rsup_MC);
absRinf_MC = abs(Rinf_MC);

absRmid_MC = sqrt(SUU_MC ./ SRR_MC);

%============== compare mean/median ================
medianrealRsup         = nanmedian(realRsup_MC);
medianimagRsup         = nanmedian(imagRsup_MC);
absmedianRsup          = sqrt(medianrealRsup .^2 + ...
    medianimagRsup .^2);
phaseRmediansup        = atan2(medianimagRsup,medianrealRsup);
phaseRmediansup_degree = phaseRmediansup*180/pi;

meanrealRsup         = nanmean(realRsup_MC);
meanimagRsup         = nanmean(imagRsup_MC);
absmeanRsup          = sqrt(meanrealRsup .^2 + ...
    meanimagRsup .^2);
phaseRmeansup        = atan2(meanimagRsup,meanrealRsup);
phaseRmeansup_degree = phaseRmeansup*180/pi;
%================================================

hatsigmau   = sigmaU*(1+randn(Lruns,1)/sqrt(twoMminus1));
hatsigmar   = sigmaR*(1+randn(Lruns,1)/sqrt(twoMminus1));

hatsigmau2  = hatsigmau .^2;
hatsigmar2  = hatsigmar .^2;

AK               = (SUU_MC .* absHR2) ./ (absSUR_MC .^2);
hatgammaSOI      = 0.5*(1+sqrt(1+4*hatsigmau2 .* AK)) ./ AK;
hatabslambda_MC  = absSUR_MC ./ (hatgammaSOI .* absHR2);

%================================================
noiseratioEff2=noiseratioEff^2;
if noiseratioEff>1
    Hgknown_MC = (((noiseratioEff2-1)/(2*noiseratioEff2)) ...
        .*(absRsup_MC)) .*...
        (1+sqrt(1+4*noiseratioEff2/((noiseratioEff2-1)^2)*...
        absRinf_MC ./ absRsup_MC));
end

MSC_MC = real((absSUR_MC .^2) ./ (SUU_MC .*SRR_MC));

%=========================
% histograms
[hRinf,binRinf] = hist(absRinf_MC,nbbins);
pdfRinf_MC      = hRinf /Lruns/(binRinf(2)-binRinf(1));

[hRsup,binRsup] = hist(absRsup_MC,nbbins);
pdfRsup_MC      = hRsup /Lruns/(binRsup(2)-binRsup(1));

[hRmid,binRmid] = hist(absRmid_MC,nbbins);
pdfRmid_MC      = hRmid /Lruns/(binRmid(2)-binRmid(1));

[harg,binarg]  = hist(phaseRsup_degree,nbbins);
pdfarg_MC      = harg /Lruns/(binarg(2)-binarg(1));

if noiseratioEff>1
    [hgknown,bingknown] = hist(Hgknown_MC,nbbins);
    pdfgknown_MC        = hgknown /Lruns/(bingknown(2)-bingknown(1));
end

mean2expH        = -(absRinf_MC - absRsup_MC);
[hmean,binmean]  = hist(mean2expH,nbbins);
pdfmean_MC       = hmean /Lruns/(binmean(2)-binmean(1));

[hmym,binmym]    = hist(hatabslambda_MC,100);
pdfmym_MC        = hmym /Lruns/(binmym(2)-binmym(1));

%============ theoretical expressions
allT.TUUonUR     = binRsup;
allT.TURonRR     = binRinf;
allT.MSC         = 0;
allT.phase       = binarg;
[hatpdfGsup, hatpdfGinf, hatMSC, hatPhase] = ...
    statsRatiosHbis(allT,spectralmatrix,twoMminus1,0.05);

%%
stdapMLEdeltamethod                = ...
    sqrt(absHUonHR * absHUonHR * (1-MSC_true)/sqrt(MSC_true)/(2*twoMminus1));
hatpdfappMLE                       = ...
    (1/sqrt(2*pi)/stdapMLEdeltamethod)* ...
    exp(-(binmym-1) .^2/(2 * stdapMLEdeltamethod^2));
%=========== conditional expectation
% E(Rsup|C\in(c1,c2))
LLlistind = 200;
listind   = linspace(0.1,1,LLlistind);
Econd     = zeros(LLlistind-1,1);
for ib=2:LLlistind
    condindex   = and(MSC_MC>listind(ib-1), MSC_MC<listind(ib));
    Econd(ib-1) = nanmean(absRsup_MC(condindex));
end
%%
if 1
    figure(1)
    clf
    subplot(2,1,1)
    bar(binRsup,pdfRsup_MC)
    hold on
    plot(binRsup,hatpdfGsup.pdf,'r','linew',2)
    hold off


    %== title
    title(sprintf('sensor gain ratio = %5.1f, true MSC = % 4.2f,\nnoise ratio = %2i, M = %i',...
        absHUonHR, MSC_true, Ninlets, (twoMminus1+1)/2),'fontsize',12)

   %=========================================
```

```
    subplot(2,1,2)
    bar(binRinf,pdfRinf_MC)
    hold on
    plot(binRinf,hatpdfGinf.pdf,'r','linew',2)
    hold off

    %     %=========================================
    %
    %     subplot(3,1,3)
    %     %     bar(bingknown,pdfgknown_MC)
    %     %     set(gca,'xlim',[0.2 1.8])
    %     bar(binarg,pdfarg_MC)
    %     hold on
    %     plot(binarg,hatPhase,'r','linew',2)
    %     hold off
    %     grid on
    %     % text('string',txt,'interpreter','latex','pos',[0.75,3],'fontsize',16)
    %     set(gca,'fontname','times','fontsize',12)
    %=========================================
    txtsupinf    = cell(2,1);
%     textEsup = sprintf('mean-sup = %4.5f', nanmean(absRsup_MC));
%     textEinf = sprintf('mean-inf = %4.5f', nanmean(absRinf_MC));
    textEsup = sprintf('mean-sup = %4.3f', hatpdfGsup.mean);
    textEinf = sprintf('mean-inf = %4.3f', hatpdfGinf.mean);

    txtsupinf{1} = '$\widehat{R}_{\sup}$';
    txtsupinf{2} = '$\widehat{R}_{\inf}$';

    textR{1} = sprintf('%s\n%s',txtsupinf{1},textEsup);
    textR{2} = sprintf('%s\n%s',txtsupinf{2},textEinf);

for ii=1:2
    subplot(2,1,ii)
    set(gca,'fontname','times','fontsize',12)
    text('string',textR{ii},...
        'interpreter','latex','pos',[1.08 4.8],'fontsize',12)
    set(gca,'xlim',nanmean(absRsup_MC) + 0.3*[-1 1])
    set(gca,'ylim',[0 10])
    grid on
end

    HorizontalSize = 12;
    VerticalSize   = 8;
    set(gcf,'units','centimeters');
    set(gcf,'paperunits','centimeters');
    set(gcf,'PaperType','a4');
%     set(gcf,'position',[0 5 HorizontalSize VerticalSize]);
    set(gcf,'paperposition',[0 0 HorizontalSize VerticalSize]);

    set(gcf,'color', [1,1,0.92]);%0.7*ones(3,1))
    set(gcf, 'InvertHardCopy', 'off');
end

% figure(1); print -depsc -loose ../../textes/6distConjointHMSC/figures/theoreticaldistribratios.eps

%=========================================
if 0
    threshold1             = MSC_true;
    threshold2             = 0.98;
    figure(2)
    clf
    subplot(121)
    plot(MSC_MC(1:Lruns),(absRsup_MC(1:Lruns)),'.')
    set(gca,'ylim',[0 2.5])
    set(gca,'xlim',[0 1])
    grid on
    hold on
    plot([0,1],absHUonHR*ones(2,1),'k','linew',2)
    hold off
    set(gca,'fontname','times','fontsize',18)
    xlabel('MSC','fontsize',18)
    ylabel('Rsup','fontsize',18)

    % subplot(223)
    % plot(listind(2:LLlistind)-0.5* (listind(2)-listind(1)),...
    %     20*log10(Econd))
    % grid on
    % set(gca,'fontn','times','fonts',10)
    % ylabel('Conditional expectation - dB')

    text00 = sprintf('Simulation based on spectral matrix');
    text0 = sprintf('True ratio HUT/HREF = %i,\nTrue MSC = % 4.2f\nNoise power ratio = %2i (%4.1f dB)\nSmoothing window number = %i\nLruns = %i',  ...
        absHUonHR,MSC_true, Ninlets, 20*log10(noiseratioEff), twoMminus1, Lruns);
    text1 = sprintf('E(Rsup|MSC>%4.2f) - 1 = %4.5f', threshold1, ...
        nanmean(absRsup_MC(MSC_MC>threshold1))-1);
    text2 = sprintf('E(Rsup|MSC>%4.2f) - 1 = %4.5f', threshold2, ...
        nanmean(absRsup_MC(MSC_MC>threshold2))-1);%, sum(MSC_MC>threshold2));
%     text3 = sprintf('E(Rsup) = %4.5f', nanmean(absRsup_MC));
    text4 = sprintf('median(Rsup) = %4.5f', nanmedian(absRsup_MC));
%     text5 = sprintf('E(Rinf) = %4.5f', nanmean(absRinf_MC));
    txts = sprintf('%s\n%s\n%s\n%s\n%s\n%s',text00,text0,text1,text2,textEsup,textEinf);

    subplot(122)
    set(gca,'ylim',[0 0.2])
    set(gca,'xtick',[],'ytick',[],'box','off')
    set(gca,'color', [1,1,0.92])
    set(gca,'xcolor', [1,1,0.92])
    set(gca,'ycolor', [1,1,0.92])
    text(0,0.1,txts,'fontname','times','fontsize',18)
    hpos = get(gca,'position');
    set(gca,'position',[0.52 hpos(2:4)])

    set(gcf,'color', [1,1,0.92]);%0.7*ones(3,1))
    set(gcf, 'InvertHardCopy', 'off');

end
%%
printdirectory  = ' ../../textes/6distConjointHMSC/slidesITW2015/';
fileprintepscmd = sprintf('print -depsc -loose %stheoreticaldistribratios.eps',printdirectory);
fileeps2pdfcmd  = sprintf('/Library/TeX/texbin/epstopdf %stheoreticaldistribratios.eps',printdirectory);
filermcmd       = sprintf('!rm %stheoreticaldistribratios.eps',printdirectory);
saveflag=0;
%
if saveflag
```

```
    eval(fileprintepscmd)
%     eval(fileeps2pdfcmd)
    eval(filermcmd)
end
```

4. Extraction from DB

Settings for the query to the database

```
%============== RUNextractfromDB.m ====================================
clear

addpath   ZZtoolbox/
addpath   ZZtoolbox/00pierrick/

directorysave2daysignals  = '../../../AAdataI26calib/';

%=== temporary files
temporary_gparse_dir = 'ZZtoolbox/00pierrick/tempfiles/';
if not(exist(temporary_gparse_dir,'dir'))
    rmdir(temporary_gparse_dir,'s')
     mkdir(temporary_gparse_dir)
else
    commandrmparse = sprintf('!rm %s.*',temporary_gparse_dir);
    eval(commandrmparse);
end
if exist('gparse.wfdisc','file')
    !rm gparse*.*;
end
%========= source of data
data_source = 'testbed_archive';
user        = 'charbit';
password    = 'sqlmomo';
channel     = '(''BDF'',''BDF'',''LWS'',''LWD'',''LKO'')';

yearstart   =  '2015';
monthstart  =  '08';
HMSstart    = '00:00:10';
yearend     =  '2015';
monthend    =  '08';
HMSend      = '23:50:10';

for ihc=4%:8
    stations    = sprintf(' (''I26H%i'',''I26C%i'') ',ihc,ihc);
    for daystart_num    =  23:2:29
        if daystart_num<10
            daystart    = ['0' num2str(daystart_num)];
            if daystart_num==9
                dayend  = '10';
            else
                dayend  = ['0' num2str(daystart_num+1)];
            end
        else
            daystart    = num2str(daystart_num);
            dayend      = num2str(daystart_num+1);
        end
        %=== clean temporary files
        commandclean = sprintf('!rm %s/*.*',temporary_gparse_dir);
        eval(commandclean)
        %=== extract data from the database
        h_starttime   = sprintf('%s/%s/%s %s',yearstart,monthstart,daystart, HMSstart);
        h_endtime     = sprintf('%s/%s/%s %s',yearend,monthend,dayend, HMSend);
        [~,starttime] = unix(['h2e ',h_starttime,' ofmt="%#"']);
        [~,endtime]   = unix(['h2e ',h_endtime,' ofmt="%#"']);
        starttime     = str2double(starttime);
        endtime       = str2double(endtime);
        wlength       = endtime-starttime;
        extractfromDB
        if nowfdiscflag
            display('***** .wfdisc does not exist');
        else
            savesignals
        end
        if exist(sprintf('%sgparse.wfdisc',temporary_gparse_dir),'file')
            commandrmparse = sprintf('!rm %s.*',temporary_gparse_dir);
            eval(commandrmparse);
        end
    end
end
%=================================================================
```

Query to the database

```
%============= extractfromDB.m ========================================
%==== Write the query
% this programm is called by
%       - RUNextractfromDB.m which provides the settings
% used functions:
%       - convertCSStomatlab
%
%
fid           = fopen(sprintf('%sgparse_temp.par',temporary_gparse_dir),'w');
fprintf(fid, 'open data_source=%s user=%s password=%s\n', data_source, user, password);
fprintf(fid, '%s\n',['query wfdisc select * from sel3.wfdisc where sta in ', ...
    stations, 'and chan in ', channel,'  and time between ',num2str(starttime),...
    '  and ',num2str(starttime+wlength),' order by sta,chan,time']);
fprintf(fid, '%s\n','read waveforms');
fprintf(fid, '%s\n','write waveforms');
fclose(fid);
%====
disp('************************************************************')
disp('****************** query to data base **********************')
%==== Gparse run
unix('setenv ORACLE_HOME /cots/oracle/oracle-10.2;');
unix('setenv D_LIBRARY_PATH $ORACLE_HOME/lib:$ORACLE_HOME/lib32;');
commandunix = ...
    sprintf('unix(''/ctbto/ims/sm/local/linux/Geotool++/2.3.10/bin/gparse < %sgparse_temp.par;'');',...
    temporary_gparse_dir);
eval(commandunix);
if exist('gparse.wfdisc','file')
    nowfdiscflag = 0;
    commandmove = sprintf('!mv gparse*.* %s.',temporary_gparse_dir);
else
```

```
    nowfdiscflag = 1;
    return
end

eval(commandmove);
%====
disp('************************************************************')
disp('***************** convert to Matlab format ******************')
%====================== Convert to Matlab
filewfdisc      = sprintf('%sgparse.wfdisc',temporary_gparse_dir);
[records, samplerate] = convertCSStomatlab(filewfdisc);
%====================== end ============================
```

Conversion to format `.mat`

```
function [records, samprate] = convertCSStomatlab(filewfdisc)
%=================================================================
% convert data from wfdisc into Matlab format .mat
%
%=================================================================
fid   = fopen(filewfdisc,'r');
tline = fgetl(fid);
iline = 0;
while ischar(tline)
    iline           = iline+1;
    station{iline}  = strtrim(tline(1:5));
    chan{iline}     = strtrim(tline(6:13));
    stime(iline)    = str2double(tline(14:34));
    wfid(iline)     = str2double(tline(36:43));
    chanid(iline)   = str2double(tline(44:52));
    jdate(iline)    = str2double(tline(53:61));
    etime(iline)    = str2double(tline(62:79));
    nsamp(iline)    = str2double(tline(80:88));
    samprate(iline) = str2double(tline(89:100));
    calib(iline)    = str2double(tline(101:117));
    calper(iline)   = str2double(tline(118:135));
    instype{iline}  = tline(135:136);
    segtype{iline}  = tline(137:144);
    datatype{iline} = tline(144:146);
    clip{iline}     = tline(147:148);
    dir1{iline}     = strtrim(tline(149:209));
    dfile{iline}    = strtrim(tline(210:247));
    foff(iline)     = str2double(tline(248:257));
    commid(iline)   = str2double(tline(258:266));
    lddate{iline}   = tline(267:276);
    tline           = fgetl(fid);
end
fclose(fid);
% DY              = num2str(jdate(1));
% NaNflag = sprintf('%ssta%s_Y%s_D%s.mat',...
%     dirname,station{1}(5),DY(1:4),DY(5:end));
% ratiorates    = samprate / min(samprate);
length_record   = fix(etime-stime) .* samprate;
% Read waveforms
fid             = fopen(filewfdisc(1:end-5),'r','b');
signal_temp     = fread(fid,'float32');
fclose(fid);
%========= save data
records = cell(length(wfid),1);
for is = 1 : length(wfid)
    offset              = foff(is)/4;
    signal_is           = signal_temp(offset + 1:offset+length_record(is));
    records{is}.data    = signal_is*calib(is);
    records{is}.Fs_Hz   = samprate(is);
    records{is}.stime   = stime(is);
    records{is}.etime   = etime(is);
    records{is}.station = station{is};
    records{is}.channel = chan{is};
end
%=================================================================
```

5. Full process

5.1 Example of filter bank description

5.2 Main program for estimation

```
%====================== estimationwithFB.m ==============================
% program evaluates some parameters from the signals
% located in directorysignals. These parameters are
% saved in directoryresults.
% Then for display use programs in the directory
% progs2display.
%
%========================================================================
% used function
%     - fbankanalysis.m
%
%========================================================================
% Rk:
% The 3 following quantities do not depend on the
% index ifile. Therefore they could be performed
% outside of the loop on ifile. Indeed
% SUTs(ip).indexinsidefreqband can be performed outside
% of the function fbankanalysis.m
% idipinf       = zeros(Pfilter,1);
% idipsup       = zeros(Pfilter,1);
% cumsumnbfq_ip = zeros(Pfilter,2);
% In the following program they are performed at the output
% of the function fbankanalysis.m
%========================================================================
%
%=================== Warning ===============
% we have observed huge outliers in the following files:
% ihc==1, date==2015/10/07 ,from sample index 2.4e6
% ihc==1, date==2015/10/09 ,from sample index 2.4e6
% ihc==2, date==2015/08/07 ,from sample index 2.4e6
% ihc==2, date==2015/10/05 ,from sample index 2.4e6
% ihc==5, date==2015/10/05
% ihc==6, date==2015/10/07
% ihc==8, date==2015/10/07 , from sample index 2.5e6
% in this version we remove them
%
%========================================================================

clear
allcolors = ['b.';'r.';'m.';'c.';'g.';'k.';'rx';'yx';'mx';'rx';'kx';...
    'c.';'k.';'r.';'c.';'m.';'g.';'b.';'k.';'r.';'c.';'m.';'g.';'k.'];
%====================
MSCthreshold = 0.98;
%====================

FLAGsaveall   = 0;
FLAGsavesmall = 1;
addpath ZZtoolbox/

%=== directory of input signals
directorysignals    = '../../../AAdataI26calib/';
%=== directory of output results
% if FLAGsaveall=1
directoryresultsALL = 'BBresults';
% if FLAGsavesmall=1
directoryresults    = sprintf('AAresultswithFB%ibis',fix(MSCthreshold*100));

%============== load the filter bank characteristics ====================
%  the useful variable is FILTERCHARACT
%======
% if generated by geneFB.m, use LOAD; if it is a .m program use RUN
% filtercharactfilename = 'filtercharacteristics/filtercharacteristics';
%cmdloadfilter        = sprintf('load(''%s'')',filtercharactfilename);
filtercharactfilename = 'filtercharacteristics/filtercharacteristics1.m';
cmdloadfilter         = sprintf('run(''%s'')',filtercharactfilename);
eval(cmdloadfilter);

%======
nofilterflag = 0;

if nofilterflag
    clear filtercharact;
    filtercharact(1).designname     = 'butter';
    filtercharact(1).Norder         = 2;
    filtercharact(1).Wlow_Hz        = 0.01;
    filtercharact(1).Whigh_Hz       = 5;
    filtercharact(1).SCPperiod_sec  = 300;
    filtercharact(1).windowshape    = 'hann';
    filtercharact(1).overlapDFT     = 0.5;
    filtercharact(1).overlapSCP     = 0;
    filtercharact(1).ratioDFT2SCP   = 5;
end

%====================
Pfilter = length(filtercharact);

% if and(Pfilter==1, filtercharact(Pfilter).Norder==0)
%     filtercharact(Pfilter).Wlow_Hz  = 0.001;
%     filtercharact(Pfilter).Whigh_Hz = 10;
% end

for ihc = 1, ihc
    %==================== read data ========================
    fileswithdotmat              = dir(sprintf('%ss%i/s%iy*.mat',...
        directorysignals,ihc,ihc));
    nbmats                       = length(fileswithdotmat);
    allfrqsPfilters              = zeros(10000,nbmats);
    allRatioSupPfilters          = zeros(10000,nbmats);
    allSTDmodRatioSupPfilters    = zeros(10000,nbmats);
    allSTDphaseRatioSupPfilters  = zeros(10000,nbmats);

    allRatioInfPfilters          = zeros(10000,nbmats);
    allSTDmodRatioInfPfilters    = zeros(10000,nbmats);
    allSTDphaseRatioInfPfilters  = zeros(10000,nbmats);
```

```
allmeanMSCcstPfilters        = zeros(10000,nbmats);
nbofvaluesoverthreshold      = zeros(10000,nbmats);
allScpPfilters               = zeros(3,10000,nbmats);
%================================================
for ifile=1:nbmats, ifile,tic
    fullfilename_i      = fileswithdotmat(ifile).name;
    dotlocation         = strfind(fullfilename_i,'.');
    underscorelocation  = strfind(fullfilename_i,'_');
    filenameonly        = fullfilename_i(...
        setdiff(1:dotlocation-1,...
        underscorelocation));
    commandload         = sprintf('load %ss%i/%s',...
        directorysignals,ihc,fullfilename_i);
    eval(commandload)
    Ts_sec = 1/Fs_Hz;
    Ntotal = size(signals_centered,1);

    date_i = sprintf('%s/%s/%s',fullfilename_i(7:10),...
        fullfilename_i(16:17),fullfilename_i(21:22));

    % ihc==1, date==2015/10/07 ,from sample index 2.4e6
    % ihc==1, date==2015/10/09 ,from sample index 2.4e6
    % ihc==2, date==2015/08/07 ,from sample index 2.4e6
    % ihc==2, date==2015/10/05 ,from sample index 2.4e6
    % ihc==5, date==2015/10/05
    % ihc==6, date==2015/10/07
    % ihc==8, date==2015/10/07 , from sample index 2.5e6

    %============ Warning ======================
    %===========================================
    %===========================================
    %== some manual checks
    if and(ihc==1,date_i=='2015/10/07')
        signals_centered = signals_centered(0.6e6:2.2e6,:);
    end
    if and(ihc==1,date_i=='2015/10/09')
        signals_centered = signals_centered(0.6e6:Ntotal,:);
    end
    if and(ihc==2,date_i=='2015/08/07')
        signals_centered = signals_centered(1:2.4e6,:);
    end
    if and(ihc==2,date_i=='2015/10/05')
        signals_centered = ...
            signals_centered([1:0.5e6 1e6:Ntotal],:);
    end
    if and(ihc==2,date_i=='2015/10/13')
        signals_centered = ...
            signals_centered([1e6:2.2e6 2.3e6:Ntotal],:);
    end

    if and(ihc==5,date_i=='2015/10/05')
        signals_centered = signals_centered(1:2.4e6,:);
    end

    if and(ihc==6,date_i=='2015/10/07')
        signals_centered = signals_centered(1:2.14e6,:);
    end

    if and(ihc==8,date_i=='2015/10/07')
        signals_centered = signals_centered(1:2.5e6,:);
    end
    %===========================================
    %===========================================
    % notice that the SUTs is not saved, therefore we have only the
    % last associated the laxt index which is NBMATS

    [SUTs, filteredsignals, allfrqsFFT_Hz, ...
        alltimes_sec, filterbank] = ...
        fbankanalysis(signals_centered,...
        filtercharact,Fs_Hz,MSCthreshold);
    %================================================
    % These 3 following quantities do not depend on the
    % index ifile. Therefore they could be performed
    % outside of the loop on ifile. Indeed
    % SUTs(ip).indexinsidefreqband can be performed outside
    % of the function fbankanalysis.m
    idipinf       = zeros(Pfilter,1);
    idipsup       = zeros(Pfilter,1);
    cumsumnbfq_ip = zeros(Pfilter,2);
    id1     = 1;
    for ip=1:Pfilter
        cumsumnbfq_ip(ip,1) = id1;
        idipinf(ip) = SUTs(ip).indexinsidefreqband(1);
        idipsup(ip) = SUTs(ip).indexinsidefreqband(2);
        id2         = id1+(idipsup(ip)-idipinf(ip));
        cumsumnbfq_ip(ip,2) = id2;
        id1         = id2+1;
    end
    %======
    for ip=1:Pfilter
        id1               = cumsumnbfq_ip(ip,1);
        id2               = cumsumnbfq_ip(ip,2);
        allRatioSupPfilters(id1:id2,ifile) = ...
            SUTs(ip).estimRsup.modcst(idipinf(ip):idipsup(ip)) .* ...
            exp(1i*SUTs(ip).estimRsup.phasecst(idipinf(ip):idipsup(ip)));
        allSTDmodRatioSupPfilters(id1:id2,ifile) = ...
            SUTs(ip).estimRsup.stdmodcst(idipinf(ip):idipsup(ip));
        allSTDphaseRatioSupPfilters(id1:id2,ifile) = ...
            SUTs(ip).estimRsup.stdphasecst(idipinf(ip):idipsup(ip));

        allRatioInfPfilters(id1:id2,ifile) = ...
            SUTs(ip).estimRinf.modcst(idipinf(ip):idipsup(ip)) .* ...
            exp(1i*SUTs(ip).estimRinf.phasecst(idipinf(ip):idipsup(ip)));
        allSTDmodRatioInfPfilters(id1:id2,ifile) = ...
            SUTs(ip).estimRinf.stdmodcst(idipinf(ip):idipsup(ip));
        allSTDphaseRatioInfPfilters(id1:id2,ifile) = ...
            SUTs(ip).estimRinf.stdphasecst(idipinf(ip):idipsup(ip));

        allmeanMSCcstPfilters(id1:id2,ifile) = ...
            nanmean(SUTs(ip).allMSCs.tabcst(idipinf(ip):idipsup(ip),:),2);
        allfrqsPfilters(id1:id2,ifile) = ...
            SUTs(ip).frqsFFT_Hz(idipinf(ip):idipsup(ip))';
        nbofvaluesoverthreshold(id1:id2,ifile) = ...
            sum(not(isnan(SUTs(ip).allMSCs.tabcst(idipinf(ip):idipsup(ip),:))),2);
        allScpPfilters(:,id1:id2,ifile) = ...
```

```
                squeeze(SUTs(ip).spectralmatrix(:,idipinf(ip):idipsup(ip)));
        end
%         if FLAGsaveall
%             comsave = ...
%                 sprintf('save %s/s%i/resultssta26sensor%s', ...
%                 directoryresultsALL,ihc,filenameonly);
%             eval(comsave);
%         end
%        toc
    end

    allRatioSupPfilters         = allRatioSupPfilters(1:id2,:);
    allSTDmodRatioSupPfilters   = allSTDmodRatioSupPfilters(1:id2,:);
    allSTDphaseRatioSupPfilters = allSTDphaseRatioSupPfilters(1:id2,:);

    allRatioInfPfilters         = allRatioInfPfilters(1:id2,:);
    allSTDmodRatioInfPfilters   = allSTDmodRatioInfPfilters(1:id2,:);
    allSTDphaseRatioInfPfilters = allSTDphaseRatioInfPfilters(1:id2,:);

    allfrqsPfilters             = allfrqsPfilters(1:id2,1);
    allmeanMSCcstPfilters       = allmeanMSCcstPfilters(1:id2,:);
    nbofvaluesoverthreshold     = nbofvaluesoverthreshold(1:id2,:);

    allScpPfilters              = allScpPfilters(:,1:id2,:);

    if FLAGsavesmall
        comsave = ...
            sprintf('save %s/resultssta26sensor%i',...
            directoryresults,ihc);
        clear signals_centered
        clear filteredsignals
        clear alltimes_sec
        clear SUTs
        eval(comsave);
    end
end
%========================== END =======================================
%%
% ip=1;
% NaverageFFTs = filtercharact(1).ratioDFT2SCP;
% allT.TUUonUR = linspace(0.7,1.3,100);
% allT.TURonRR = linspace(0.7,1.3,100);
% allT.MSC     = linspace(0.5,1,100);
% allT.phase   = linspace(0,2*pi,100);
%
%
% listifq = cumsumnbfq_ip(ip,1):cumsumnbfq_ip(ip,2);
% Llistifq = length(listifq);
% STDmodtheo_ip = zeros(Llistifq,1);
%
% for indfq=1:Llistifq
%     ifq=listifq(indfq);
%     SCP_ip_ifq = nanmean(allScpPfilters(:,ifq,:),3);
%     if any(isnan(SCP_ip_ifq))
%         STDmodtheo_ip(indfq)=NaN;
%     else
%         RR=[SCP_ip_ifq(1) SCP_ip_ifq(3);SCP_ip_ifq(3)' SCP_ip_ifq(2)];
%         [statUUonUR, statURonRR, statMSC]    = ...
%             statsRatiosHbis(allT, RR, NaverageFFTs, 0.3);
%         STDmodtheo_ip(indfq) = diff(statUUonUR.CI)/2;
%     end
% end
% nbofvalues_ip = sum(nbofvaluesoverthreshold(cumsumnbfq_ip(ip,1):cumsumnbfq_ip(ip,2),:),2);
% STDmodempiric_ip = nanmean(...
%     allSTDmodRatioSupPfilters(cumsumnbfq_ip(ip,1):cumsumnbfq_ip(ip,2),:),2);
% corrlevel = corr(STDmodtheo_ip(and(not(isnan(STDmodtheo_ip)),not(STDmodempiric_ip==0))), STDmodempiric_ip(and(not(isnan(STDmodtheo_ip)),not(STDmodempiric
%
% [STDmodtheo_ip./(STDmodempiric_ip) nbofvalues_ip]
% corrlevel
%
% figure(1)
% subplot(121)
% plot( allT.TUUonUR,statURonRR.pdf,'.-')
% hold on
% plot( allT.TUUonUR,statUUonUR.pdf,'.-r')
% hold off
% subplot(122)
% plot(STDmodtheo_ip, STDmodempiric_ip,'.')
%
```

6. Application program

The first following program extracts the final ratio between the SUT and the Sref sensor response, using the data available in the format `.mat`. The data consist of many files for each of the 8 sensors of IS26. All the ratios are saved in a selected directory.

6.1 temporalevolution.m

6.2 displaySensorResponse.m

Chapter 11

Toolbox

1. Main analysis function: `fbankanalysis.m`

```
%=========================================================================
function [SUTs, filteredsignals, allfrqsFFT_Hz, alltimes_sec, ...
    filterbank] = ...
    fbankanalysis(sigin, ...
    filtercharacteristics, ...
    Fs_Hz,...
    MSCthreshold)
%=========================================================================
% synopsis:
%    SUTs = fbankanalysis...
%          (sigin,filtercharacteristics,Fs_Hz,MSCthreshold)
% DFT for discrete Fourier transform
% SCP for spectral components
%     SCP_k = MEAN_{m=1}^M wDFT_{m,k}
%     wDFT_{m,k} = DFT( window .* signal_m, k, Lfft)
%          where signal_m is the m-th segment with overlap,
%          Lfft the DFT length
%          and k the frequency index associated to the
%          frequency k*Fs_Hz/Lfft
%=========================================================================
% Remark:
% The 3 parameters, M, Lfft, and stationarity-duration are
% related by:
%              stationarity-duration ~ M * Lfft / Fs_Hz
%
% The choice of M is related to the accuracy of the estimator
% In the following we focus on the stationarity-duration value,
% depending on the frequency band, and on M:
%    stationarity-duration is defined by xx.SCPperiod_sec
%    M is defined by xx.ratioDFT2SCP (overlap not applied)
%    therefore we derive Lfft, regarding the overlap
%    xx.overlapDFT
%=========================================================================
% Inputs:
%    sigin: input signals T x 2
%    filtercharacteristics: structure Px1
%         P: number of frequency bands
%         xx.Norder: order of the filter
%                if xx.Norder=0, no filtering
%         xx.Wlow_Hz: lower bound of the frequency band in Hz
%         xx.Whigh_Hz: upper bound of the frequency band in Hz
%         xx.SCPperiod_sec: duration in second
%                 over which SPC is performed,
%                 expected as stationarity time
%         xx.windowshape: window shape for spectral
%                 analysis (ex. 'hann' or 'hamming', ...)
%         xx.overlapDFT: overlap rate for successive
%                 DFT (typically 0.5)
%         xx.overlapSCP: overlap rate for successive
%                 spectral components (typically 0)
%         xx.ratioDFT2SCP: ratio between period_sec and
%                 DFT duration (typical integer value is 5).
%     Fs_Hz: sampling frequency in Hz
%     MSCthreshold: MSC threshold (advised value > 0.99)
%
%=========================================================================
% Rk: the spectral components are performed on successive DFTs with
%     overlapping. If M denotes the ratioFFT2DSP and
%     if overlapFFT is 0.5, the number of DFTs is 9
%=========================================================
% Example of filtercharacteristics structure:
%=========================================================================
%         xx.Norder: 4
%         xx.Wlow_Hz: 0.02
%         xx.Whigh_Hz: 0.2
%         xx.SCPperiod_sec: 250
%         xx.windowshape: 'hann'
%         xx.overlapDFT: 0.5
%         xx.overlapSCP: 0
%         xx.ratioDFT2SCP: 5.
%=========================================================================
% filtercharact(1).Norder           = 4;
% filtercharact(1).Wlow_Hz          = 0.02;
% filtercharact(1).Whigh_Hz         = 0.2;
% filtercharact(1).SCPperiod_sec    = 500;
% filtercharact(1).windowshape      = 'hann';
% filtercharact(1).overlapDFT       = 0.5;
% filtercharact(1).overlapSCP       = 0;
% filtercharact(1).ratioDFT2SCP     = 5;
%=========================================================================
```

```
%=========================================================================
% Outputs:
%     SUTs: structures P x 1
%         xx.estimRsup: Rsup ratio
%         xx.estimRinf: Rinf ratio
%         xx.allMSCs: allMSCs;
%         xx.Nsupthreshold: count of the number of values over
%                 the threshold
%         xx.Nsupthresholdintheband: counts of the number of values
%                 over the threshold in the filter bandwidth
%         xx.SCP = all spectral components
%         xx.indexinsidefreqband = P x 1, indices of the
%                  frequency bounds of each filter
%                  in the xx.frqsFFT_Hz
%         alltimes_sec: cell  Px 1, each cell consists of
%             yy.FFT: time list in second of the DFTs
%             yy.SD: time list in second of the SCPs
%             yy.signals: time list in second of the signals
%
%         frqsFFT_Hz: cell P x 1, each cell consists of
%                  frequency list in Hz of the DFTs
%=========================================================================
% included functions: estimSCP, estimSUT, fbank
%=========================================================================
%=========================================================================


%=== for theoretical STD computation
allT.TUUonUR = linspace(0.6,2,100);
allT.TURonRR = linspace(0.6,2,100);
allT.MSC     = linspace(0.6,1,100);
allT.phase   = linspace(0,2*pi,100);

%=== for SCP analysis
overlapDFT       = filtercharacteristics.overlapDFT;
overlapSCP       = filtercharacteristics.overlapSCP;
ratioDFT2SCP     = filtercharacteristics.ratioDFT2SCP;
P                = length(filtercharacteristics);
allfrqsFFT_Hz    = cell(P,1);
alltimes_sec     = cell(P,1);
SUTs             = struct;
T                = size(sigin,1);
%===== filtering SIGIN ==> SIGOUT
[sigout, filterbank]  = fbank(sigin,filtercharacteristics,Fs_Hz);
filteredsignals       = zeros(T,2,P);
for ifilter = 1:P
    taustationary_sec = filtercharacteristics(ifilter).SCPperiod_sec;
    windowingshape    = filtercharacteristics(ifilter).windowshape;
    signals   = sigout(:,:,ifilter);
    [T,d]     = size(signals);
    signals   = signals-ones(T,1)*mean(signals);
    filteredsignals(:,:,ifilter) = signals;
    Tfft_sec  = taustationary_sec/ratioDFT2SCP;
    Lfft      = fix(Tfft_sec*Fs_Hz);
    %==================== spectral analysis ========================
    [allSDs, time_temp, allfrqsFFT_Hz{ifilter}] = ...
        estimSCP(signals(:,1),signals(:,2),...
        ones(Lfft,1), overlapDFT, ...
        ratioDFT2SCP,overlapSCP, Fs_Hz, windowingshape);
    alltimes_sec{ifilter} = time_temp;
    %=================================================================
    [estimRsup, estimRinf, spectralmatrix, allMSCs, Nsupthreshold] ...
        = estimSUT(allSDs, MSCthreshold);
    SUTs(ifilter).spectralmatrix = spectralmatrix;
    SUTs(ifilter).estimRsup = estimRsup;
    SUTs(ifilter).estimRinf = estimRinf;
    SUTs(ifilter).allMSCs   = allMSCs;
    SUTs(ifilter).Nsupthreshold = Nsupthreshold;
    SUTs(ifilter).frqsFFT_Hz = allfrqsFFT_Hz{ifilter};
    SUTs(ifilter).SCP = allSDs;
    SUTs(ifilter).indexinsidefreqband = [ ...
        ceil((filtercharacteristics(ifilter).Wlow_Hz/Fs_Hz)*length(SUTs(ifilter).frqsFFT_Hz)); ...
        ceil((filtercharacteristics(ifilter).Whigh_Hz/Fs_Hz)*length(SUTs(ifilter).frqsFFT_Hz))];
    if and(P==1, filtercharacteristics(ifilter).Norder==0)
                SUTs(ifilter).Nsupthresholdintheband = ...
            nansum(SUTs(ifilter).allMSCs.indexcst,2);
    else
        SUTs(ifilter).Nsupthresholdintheband = ...
            nansum(SUTs(ifilter).allMSCs.indexcst(SUTs(ifilter).indexinsidefreqband(1):...
            SUTs(ifilter).indexinsidefreqband(2),:),2);
    end
    indexfrqinside = SUTs(ifilter).indexinsidefreqband(1):SUTs(ifilter).indexinsidefreqband(2);
    Lindexfrqinside = length(indexfrqinside);
    for indfreq_ii = 1:Lindexfrqinside
        ifreq_ii = indexfrqinside(indfreq_ii);
        elementofR_ii = spectralmatrix(:,ifreq_ii);
        if any(isnan(elementofR_ii))
            SUTs(ifilter).theoreticalmodstd(indfreq_ii) = NaN;
        else
            RR_ii = [elementofR_ii(1) elementofR_ii(3) ; ...
                elementofR_ii(3)' elementofR_ii(2)];
            [statUUonUR, statURonRR, statMSC] = statsRatiosHbis(allT, RR_ii,ratioDFT2SCP , 0.7);
            SUTs(ifilter).theoreticalmodstd(indfreq_ii) = diff(statUUonUR.CI)/2;
        end
    end
end
%=========================================================================
function [sigout, filterbank] = fbank(sigin,filtercharact,Fs_Hz)
%=========================================================================
P           = length(filtercharact);
[T,d]       = size(sigin);
sigout      = zeros(T,d,P);
filterbank = cell(P,1);
for ifilter = 1:P
    if not(filtercharact(ifilter).Norder==0)
        flow = filtercharact(ifilter).Wlow_Hz/Fs_Hz;
        fhigh = filtercharact(ifilter).Whigh_Hz/Fs_Hz;
        fname = filtercharact(ifilter).designname;
        forder = filtercharact(ifilter).Norder;
        switch fname
            case 'fir1'
                fdesign = sprintf('filnum = %s(%i,[%5.8f,%5.8f]);',...
                    fname,forder,2*flow,2*fhigh);
                filden = 1;
            case 'butter'
```

```
                fdesign = sprintf('[filnum,filden] = %s(%i,[%5.8f %5.8f]);',...
                    fname,forder,2*flow,2*fhigh);
            case 'cheby1'
                fdesign = sprintf('[filnum,filden] = %s(%i,%i,[%5.8f %5.8f]);',...
                    fname,forder,0.02,2*flow,2*fhigh);
        end
        eval(fdesign);
        sigout(:,:,ifilter) = filter(filnum,filden,sigin);
        filterbank{ifilter}.num = filnum;
        filterbank{ifilter}.den = filden;

    else
        sigout(:,:,ifilter) = sigin;
        filterbank{ifilter}.num = NaN;
        filterbank{ifilter}.den = NaN;

    end
end
%=========================================================================
function [allSDs, time_sec, frqsFFT_Hz] = ...
    estimSCP(xU,xR,GREF,overlapFFT, ...
    NaverageFFTs, overlapSD, Fs_Hz, smoothwindow)
%=========================================================================
% Perform the spectral components of the two signals xU et xR.
%=========================================================================
% The code uses the Welch's approach. The signal is shared into DFT
% windows of which the length is the length of GREF, with the
% overlap rate of OVERLAPFFT. Then the specral components is averaged
% on NaverageFFTs DFT blocks. Therefore each spectral block corresponds
% to a time period reported in TIME_SEC.
%
% Inputs:
%    xU: signal observed on the SUT (T x 1)
%    xR: signal observed on the SREF (T x 1)
%    GREF: frequency response of the SREF (Lfft x 1)
%    overlapFFT: between 0 and 1, overlap on the FFT block
%    NaverageFFTs: number of FFT-length to averaging
%    overlapSD: between 0 and 1, overlap on the averaging
%                Spectral Density block
%    Fs_Hz: sampling frequency in Hz
%    smoothwindow: character word as 'rect', 'hann', ...
%            [default: rectangular]
% Outputs:
%    allSDs.RR = auto-spectrum of SREF
%    allSDs.UU = auto-spectrum of SUT
%    allSDs.UR = cross-spectrum of (SUT,SREF)
%    allSDs.MSC = Magnitude Square Coherence
%    allSDs.Rsup = SUU/SUR
%    allSDs.Rinf = SUR/SRR
%    allSDs.det = determinant of the spectral matrix
%
%    time_sec.FFT: grid of time for the FFT blocks (in second)
%    time_sec.SD: grid of time for the Spectral Density blocks (in second)
%    time_sec.signals: grid of time for the samplig period (in second)
%    frqsFFT_Hz: grid of frequency for the interval [0, Fs_Hz] (in Hz)
%=========================================================================
%=========================================================================
xU   = xU(:);
xR   = xR(:);
N    = length(xU);
Lfft = length(GREF);
sqrtLfft    = sqrt(Lfft);
shiftSignal = fix((1-overlapFFT)*Lfft);
NblocksFFT  = fix((N-(Lfft-shiftSignal))/shiftSignal);
allFFTsRR   = zeros(Lfft,NblocksFFT);
allFFTsUU   = zeros(Lfft,NblocksFFT);
if exist('smoothwindow','var')
    switch smoothwindow
        case 'hann'
            Hwin = hann(Lfft,'periodic');
        case 'bartlett'
            Hwin = bartlett(Lfft);
        case 'hamming'
            Hwin = hamming(Lfft);
        case 'rectwin'
            Hwin = rectwin(Lfft);
        case 'blackman'
            Hwin = blackman(Lfft);
    end
else
    Hwin     = hamming(Lfft);
end
%=========================================================================
% the normalisation below is
% not useful if we only consider PSD ratios
%=========================================================================
%   <-------- NaverageFFTs = 5 ------->
%  |******|******|******|******|******|
%     |******|******|******|******|
% Given NaverageFFTs and the duration to performing
% the spectral components, we derive the length
% of the FFT. Then we derive the total nb of FFT blocks
% in the full file, taking into account the overlap.
% NblocsFFT = Ttotal/shiftFFT
%       typically shiftFFT = 1/2
% We compute all FFTs before to averaging.
% In another langage it is not necessary
% to proceed in such a way.
%
%=========================================================================
Hwin = Hwin *sqrt(Lfft/(Hwin'*Hwin));
for ibF  = 1:NblocksFFT
    ibT  = (ibF-1)*shiftSignal+(1:Lfft);
    xU_i = xU(ibT) .* Hwin;
    xU_i = xU_i-mean(xU_i);
    xR_i = xR(ibT) .* Hwin;
    xR_i = xR_i-mean(xR_i);
    allFFTsUU(:,ibF) = fft(xU_i,Lfft)/sqrtLfft;
    allFFTsRR(:,ibF) = fft(xR_i,Lfft)/sqrtLfft;
end
NaverageFFT = fix(NaverageFFTs/(1-overlapFFT))-1;
shiftFFTs = fix((1-overlapSD)*NaverageFFTs);
allSDs    = struct;
time_sec  = struct;
NshiftFFTs_with_overlap = max([2*shiftFFTs-1,1]);
```

```
NSD       = fix(NblocksFFT/NshiftFFTs_with_overlap);
for ibB=1:NSD,
    indB1 = (ibB-1)*NshiftFFTs_with_overlap+1;
    indB2 = indB1+NaverageFFT-1;
    indB  = fix(indB1):fix(indB2);
    indB  = indB(indB<= NblocksFFT);
    allSDs(ibB).RR  = mean(abs(allFFTsRR(:,indB)) .^ 2,2);
    allSDs(ibB).UU  = mean(abs(allFFTsUU(:,indB)) .^ 2,2);
    allSDs(ibB).UR  = mean(allFFTsRR(:,indB) .* ...
        conj(allFFTsUU(:,indB)),2);
    allSDs(ibB).MSC = (abs(allSDs(ibB).UR) .^2) ./...
        (allSDs(ibB).RR .* allSDs(ibB).UU);
    allSDs(ibB).Rsup  = allSDs(ibB).UU ./ allSDs(ibB).UR;
    allSDs(ibB).Rinf  = allSDs(ibB).UR ./ allSDs(ibB).RR;
    allSDs(ibB).det   = (abs(allSDs(ibB).RR) .* abs(allSDs(ibB).UU)) ...
        -(abs(allSDs(ibB).UR) .^2);
end
frqsFFT_Hz = (0:Lfft-1)*Fs_Hz/Lfft;
time_sec.FFT = ((0:NblocksFFT-1)+1/2)*shiftSignal/Fs_Hz;
time_sec.SD =  ((0:NSD-1)+1/2)*(shiftSignal/Fs_Hz)* ...
    NshiftFFTs_with_overlap;
time_sec.signals =  ((0:N-1)+1/2)/Fs_Hz;
%=========================================================================
function [Rsup, Rinf, SCPsupeta, MSC, Nsupthreshold] ...
    = estimSUT(allSDs, MSCThreshold)
%=========================================================================
% synopsis:
% estimate the stastitics of interest from the spectral
%     components. Then extract the values over the
%     threshold.
%
% [Rsup, Rinf, SpMatrix, MSC, Nsupthreshold] ...
%    = estimSUT(allSDs, MSCThreshold)
%
%====
% Inputs:
%        allSDs:
%            all spectrals components (provided by the function
%                estimSD.
%        MSCThreshold: MSC threshold, typicall 0.99
%
%====
% Outputs:
%       Rsup: ratio SUU on SUR (see document)
%         structure
%            Rsup.tabmod: table of the modulus of Rsup
%            Rsup.tabmodcst: table of the modulus of Rsup with MSC
%                   above the threshold
%
%            Rsup.tabphase: table of the phase of Rsup
%            Rsup.tabphasecst: table of the phaseof Rsup with MSC
%                   above the threshold
%
%            Rsup.mod: modulus of Rsup estimated on Rsup.tabmod
%            Rsup.modcst: modulus of Rsup estimated on Rsup.tabmod
%            Rsup.stdmodcst: STD estimated on Rsup.tabmod
%                   with MSC above the threshold
%            Rsup.phase: table of the phases estimated on Rsup.tabphase
%                   with MSC above the threshold
%            Rsup.phasecst: phases of Rsup estimated on Rsup.tabphase
%                   with MSC above the threshold;
%            Rsup.stdphasecst: STD estimated on Rsup.tabphase
%                   with MSC above the threshold
%
%       Rinf: ---- idem Rsup
%
%       SCPsupeta: spectral components 3 x Lfft
%           for cells with MSC is over the threshold.
%           For each frequency the spectral components is performed
%           as the mean for all cells over the threshold.
%
%       MSC: structure
%           MSC.tab: table of the MSC
%           MSC.tabcst: table of the MSC with values above the threshold
%           MSC.indexcst: index of the
%               table of the MSC with values above the threshold
%
%       Nsupthreshold: number of values above the threshold
%
%
%=========================================================================
%=========================================================================

%=========================================================================
%================ if weightingflag = 1
% a ponderation is applied using the estimated variance
% of the MSC estimate
%=========================================================================
weightingflag = 1;

nbblocksAVE   = length(allSDs);
Lfft          = length(allSDs(1).UU);
indextabMSCsupthreshold = false(Lfft,nbblocksAVE);
indextabMSCsupthreshold([allSDs.MSC]>MSCThreshold) = true;
Nsupthreshold = sum(sum(indextabMSCsupthreshold));

tabUU    = [allSDs.UU];
tabRR    = [allSDs.RR];
tabUR    = [allSDs.UR];
tabMSC   = [allSDs.MSC];

tabMSCwithMSCsupeta = nan(Lfft,nbblocksAVE);
tabMSCwithMSCsupeta(indextabMSCsupthreshold) = ...
    (tabMSC(indextabMSCsupthreshold));

tabUUwithMSCsupeta = nan(Lfft,nbblocksAVE);
tabUUwithMSCsupeta(indextabMSCsupthreshold) = ...
    (tabUU(indextabMSCsupthreshold));

tabRRwithMSCsupeta = nan(Lfft,nbblocksAVE);
tabRRwithMSCsupeta(indextabMSCsupthreshold) = ...
    (tabRR(indextabMSCsupthreshold));
tabURwithMSCsupeta = nan(Lfft,nbblocksAVE);
tabURwithMSCsupeta(indextabMSCsupthreshold) = ...
    (tabUR(indextabMSCsupthreshold));
```

```
SCPsupeta      = zeros(3,Lfft);
SCPsupeta(1,:) = nanmean(tabRRwithMSCsupeta,2);
SCPsupeta(2,:) = nanmean(tabUUwithMSCsupeta,2);
SCPsupeta(3,:) = nanmean(tabURwithMSCsupeta,2);

%=========================================================================
%================ on Rsup, Rinf
%=========================================================================
tabHUUUR    = [allSDs.Rsup];
tabHURRR    = [allSDs.Rinf];
%=========================================================================
%================ on Rsup
%===== real/imaginary part without constraint on MSC
%=========================================================================
tabrealHUUUR = real(tabHUUUR);
tabimagHUUUR = imag(tabHUUUR);
tabmodHUUUR  = sqrt(tabrealHUUUR .^2 + tabimagHUUUR .^2);
tabphaseHUUUR = atan2(tabimagHUUUR,tabrealHUUUR);

%=========================================================================
%================ on Rinf
%===== real/imaginary part without constraint on MSC
%=========================================================================
tabrealHURRR = real(tabHURRR);
tabimagHURRR = imag(tabHURRR);
tabmodHURRR  = sqrt(tabrealHURRR .^2 + tabimagHURRR .^2);
tabphaseHURRR = atan2(tabimagHURRR, tabrealHURRR);

%=========================================================================
%================ Rsup HAT by averaging
%=========================================================================
hatrealHUUUR = nanmean(tabrealHUUUR,2);
hatimagHUUUR = nanmean(tabimagHUUUR,2);
%===== mod/phase part without constraint
hatmodHUUUR   = sqrt(hatrealHUUUR .^2 + hatimagHUUUR .^2);
hatphaseHUUUR = atan2(hatimagHUUUR, hatrealHUUUR);

%=========================================================================
%================ Rinf HAT by averaging
%=========================================================================
hatrealHURRR = nanmean(tabrealHURRR,2);
hatimagHURRR = nanmean(tabimagHURRR,2);
%===== mod/phase part without constraint
hatmodHURRR   = sqrt(hatrealHURRR .^2 + hatimagHURRR .^2);
hatphaseHURRR = atan2(hatimagHURRR, hatrealHURRR);

%=========================================================================
%================ Rsup with constraint on the MSC
%=========================================================================
%===== real part with constraint
tabrealHUUURwithMSCsupeta = nan(Lfft,nbblocksAVE);
tabrealHUUURwithMSCsupeta(indextabMSCsupthreshold) = ...
    (real(tabHUUUR(indextabMSCsupthreshold)));
%===== imaginary part with constraint
tabimagHUUURwithMSCsupeta = nan(Lfft,nbblocksAVE);
tabimagHUUURwithMSCsupeta(indextabMSCsupthreshold) = ...
    (imag(tabHUUUR(indextabMSCsupthreshold)));
%===== module with constraint
tabmodHUUURwithMSCsupeta = sqrt(...
    tabrealHUUURwithMSCsupeta .^2+...
    tabimagHUUURwithMSCsupeta .^2);
%===== phase with constraint
tabphaseHUUURwithMSCsupeta = atan2(tabimagHUUURwithMSCsupeta,...
    tabrealHUUURwithMSCsupeta);

%=========================================================================
%================ Rinf with constraint on the MSC
%=========================================================================
%===== real part with constraint
tabrealHURRRwithMSCsupeta = nan(Lfft,nbblocksAVE);
tabrealHURRRwithMSCsupeta(indextabMSCsupthreshold) = ...
    (real(tabHURRR(indextabMSCsupthreshold)));
%===== imaginary part with constraint
tabimagHURRRwithMSCsupeta = nan(Lfft,nbblocksAVE);
tabimagHURRRwithMSCsupeta(indextabMSCsupthreshold) = ...
    (imag(tabHURRR(indextabMSCsupthreshold)));
%===== module with constraint
tabmodHURRRwithMSCsupeta = sqrt(...
    tabrealHURRRwithMSCsupeta .^2+...
    tabimagHURRRwithMSCsupeta .^2);
%===== phase with constraint
tabphaseHURRRwithMSCsupeta = atan2(tabimagHURRRwithMSCsupeta,...
    tabrealHURRRwithMSCsupeta);


%=========================================================================
%======= useful to perform the weights
RsuptheowithMSCsupeta = tabmodHUUURwithMSCsupeta .* ...
    tabmodHURRRwithMSCsupeta;

%=========================================================================
%================ if weightingflag = 1
% a ponderation is applied using the estimated variance
% of the MSC estimate
%=========================================================================
weightMSCsupeta = (tabMSCwithMSCsupeta .^2) ./  ...
    (1-tabMSCwithMSCsupeta) .* RsuptheowithMSCsupeta;
weightMSCinfeta = 1 ./  ...
    (1-tabMSCwithMSCsupeta) .* RsuptheowithMSCsupeta;
%=========================================================================
%========== Rsup HAT with constraint and weighting coeffs
%=========================================================================
if weightingflag
    hatrealHUUURwithMSCsupeta = ...
        nansum(tabrealHUUURwithMSCsupeta .* weightMSCsupeta,2) ...
        ./ nansum((weightMSCsupeta),2);

    hatimagHUUURwithMSCsupeta = ...
        nansum(tabimagHUUURwithMSCsupeta .* weightMSCsupeta,2) ...
        ./ nansum((weightMSCsupeta),2);
else
    hatrealHUUURwithMSCsupeta = ...
        nanmean(tabrealHUUURwithMSCsupeta,2);
```

```
    hatimagHUUURwithMSCsupeta = ...
        nanmean(tabimagHUUURwithMSCsupeta,2);
end

%===== perform module and phase part with constraint
hatmodHUUURwithMSCsupeta = sqrt(...
    hatrealHUUURwithMSCsupeta .^2+...
    hatimagHUUURwithMSCsupeta .^2);

hatphaseHUUURwithMSCsupeta = atan2(...
    hatimagHUUURwithMSCsupeta,...
    hatrealHUUURwithMSCsupeta);

%===== perform STD on module and phase
stdmodHUUURwithMSCsupeta   = nanstd(tabmodHUUURwithMSCsupeta,[],2);
stdphaseHUUURwithMSCsupeta = nanstd(tabphaseHUUURwithMSCsupeta,[],2);

%=========================================================================
%========== Rinf HAT with constraint and weighting coeffs
%=========================================================================
if weightingflag
    hatrealHURRRwithMSCsupeta = ...
        nansum(tabrealHURRRwithMSCsupeta .* weightMSCinfeta,2) ...
        ./ nansum((weightMSCinfeta),2);

    hatimagHURRRwithMSCsupeta = ...
        nansum(tabimagHURRRwithMSCsupeta .* weightMSCinfeta,2) ...
        ./ nansum((weightMSCinfeta),2);
else
    hatrealHURRRwithMSCsupeta = ...
        nanmean(tabrealHURRRwithMSCsupeta,2);
    hatimagHURRRwithMSCsupeta = ...
        nanmean(tabimagHURRRwithMSCsupeta,2);
end

%===== perform module and phase part with constraint
hatmodHURRRwithMSCsupeta = sqrt(...
    hatrealHURRRwithMSCsupeta .^2+...
    hatimagHURRRwithMSCsupeta .^2);

hatphaseHURRRwithMSCsupeta = atan2(...
    hatimagHURRRwithMSCsupeta,...
    hatrealHURRRwithMSCsupeta);

%===== perform STD on module and phase
stdmodHURRRwithMSCsupeta   = nanstd(tabmodHURRRwithMSCsupeta,[],2);
stdphaseHURRRwithMSCsupeta = nanstd(tabphaseHURRRwithMSCsupeta,[],2);

%=========================================================================
%============================ SAVE ========================================
Rsup.tabmod      = tabmodHUUUR;
Rsup.tabmodcst   = tabmodHUUURwithMSCsupeta;
Rsup.tabphase    = tabphaseHUUUR;
Rsup.tabphasecst = tabphaseHUUURwithMSCsupeta;

Rsup.mod         = hatmodHUUUR;
Rsup.phase       = hatphaseHUUUR;
Rsup.modcst      = hatmodHUUURwithMSCsupeta;
Rsup.stdmodcst   = stdmodHUUURwithMSCsupeta;
Rsup.phasecst    = hatphaseHUUURwithMSCsupeta;
Rsup.stdphasecst = stdphaseHUUURwithMSCsupeta;

Rinf.tabmod      = tabmodHURRR;
Rinf.tabmodcst   = tabmodHURRRwithMSCsupeta;
Rinf.tabphase    = tabphaseHURRR;
Rinf.tabphasecst = tabphaseHURRRwithMSCsupeta;

Rinf.mod         = hatmodHURRR;
Rinf.phase       = hatphaseHURRR;
Rinf.modcst      = hatmodHURRRwithMSCsupeta;
Rinf.stdmodcst   = stdmodHURRRwithMSCsupeta;
Rinf.phasecst    = hatphaseHURRRwithMSCsupeta;
Rinf.stdphasecst = stdphaseHURRRwithMSCsupeta;

MSC.tab          = tabMSC;
MSC.tabcst       = tabMSCwithMSCsupeta;
MSC.indexcst     = indextabMSCsupthreshold;
MSC.weightMSC    = weightMSCsupeta;
%========================== END ========================================
```

2. Cumulative function of $\widehat{\mathrm{MSC}}$

```
function P = cumulFunctionMSC(E,MSC,N)
%=============================================================
% Cumulative function of the MSC estimator
% derived from the spectral estimation
%                |S_xy|^2
%     MSC = ------------
%           S_xx S_yy
%=============================================================
% Synopsis
%          P = cumulFunctionMSC(MSC,N)
% where
%   MSC : true coherence value in (0,1)
%   N : size of the smoothing window
%
%=============================================================
% outputs
%   P = Prob(\hat MSC =< E; N, MSC)
%=============================================================
LE = length(E);
z = E*MSC;
sum_F = 1;
coef_ell=1;
R             = ((1-MSC) ./ (1-z)) .^ N;
for ell=1:N-2
    Tk        = ones(LE,1);
    sum_ell   = 1;
    for k=1:ell
```

```
        Tk      = (k-1-ell)*(k-N)* (Tk .*z) /k/k;
        sum_ell = sum_ell + Tk;
    end
    coef_ell    = coef_ell .* ((1-E) ./ (1-z));
    sum_F       = sum_F+coef_ell .* sum_ell;
end
P               = (R .* sum_F) .* E;
%=============================================================
```

3. Inverse cumulative function of $\widehat{\mathrm{MSC}}$

```
function c = invcumulFunctionMSC(p,C0,Nd,precision)
%=============================================================
% Inverse cumulative function of the MSC estimates
% derived from the spectral estimation
%             |S_xy|^2
%    MSC = ------------
%           S_xx S_yy
%=============================================================
% Synopsis
%         c = invcumulFunctionMSC(p,C0,Nd,precision)
% where
%   p
%   C0 : true coherence value in (0,1)
%   Nd : size of the smoothing window or averaging window
%   precision: default value 1e-15
%=============================================================
% outputs
%   P = Prob(\hat MSC =< E; N, MSC)
%   E
%   CI : confidence interval at 100a%
%
% Rk: the resolution is obtained by dichotomy, whiic is easy
% because the cumulative function is by def monotonic.
%=============================================================
if nargin < 4
    precision = 1e-15;
end
lastpos = 1;
lastneg = 0;
c       = 1;
while abs(lastpos-lastneg)>precision
    dy = cumulFunctionMSC(c,C0,Nd,p);
    if dy < 0
        lastneg = c;
        c = (lastneg+lastpos)/2;
    else
        lastpos = c;
        c = (lastneg+lastpos)/2;
    end
end
%=============================================================
function dP = cumulFunctionMSC(E,C0,Nd,p)
z = E*C0;
sum_F = 1;
coef_ell=1;
R = ((1-C0) ./ (1-z)) .^ Nd;
for ell=1:Nd-2
    Tk=1;
    sum_ell = 1;
    for k=1:ell
        Tk = (k-1-ell)*(k-Nd)* (Tk .*z) /k/k;
        sum_ell = sum_ell + Tk;
    end
    coef_ell = coef_ell .* ((1-E) ./ (1-z));
    sum_F = sum_F+coef_ell .* sum_ell;
end
dP = (R .* sum_F) .* E - p;
%=============================================================
```

4. Probability density function of $\widehat{\mathrm{MSC}}$

```
function pdf = pdfMSC(E,MSC,N)
%=============================================================
% probability density of the MSC estimates
% derived from the spectral estimation
%             |S_xy|^2
%    MSC = ------------
%           S_xx S_yy
%=============================================================
% Synopsis
%         pdf = pdfMSC(E,MSC,N)
% where
%   E : range in (0,1)
%   MSC : true coherence value in (0,1)
%   N : size of the smoothing window
%
%=============================================================
% outputs:
%   pdf: probability density function of MSC
%=============================================================
LE      = length(E);
Emu     = E*MSC;
R1      = (((1-MSC) .*(1-E)) ./ ((1-Emu) .^2)) .^ N;
R2      = (1-Emu) ./ ((1-E) .^2);
Tk      = ones(LE,1);
sumT    = 1;
for k=1:N-1
    Tk   = (k-N)*(k-N)* (Tk .*Emu) /k/k;
    sumT = sumT + Tk;
end
pdf     = (N-1) * (R1 .* R2) .* sumT;
%=============================================================
```

5. Statistics of the spectral ratios

```
function [stat11on21, stat12on22, statMSC, statPhase] = ...
    statsRatiosHbis(allT, G, N, a)

% stat11on21, stat12on22 or statUUonUR, statURonRR
%================================================
% Compute the probabiity density function of
% the ratios:
%
%       |G(1,2)|             G(1,1)
%      ----------   and    ----------
%        G(2,2)             |G(2,1)|
% where G is a positive matrix.
%
% Derive the confidence interval at 100alpha%
% To  integrate use:
%    INTEGRAL on R2013
%    QUADGK   on R2010
%
% Inputs :
%       - allT:
%             allT.T11on21: list of values of S11/abs(S21)
%             allT.T12on22: list of values of S12/abs(S22)
%             allT.phase: list of values of arg(S22)
%             allT.MSC: list of values of MSC
%       - G: 2 x 2 spectral matrix at one frequency bin
%       - N: window length for averaging FFT frames
%       - a: level of confidence, typically 0.05%
%=========================================================================
% old notations
% 1=Reference, 2=Undertest
allT.T11on21 = allT.TUUonUR;
allT.T12on22 = allT.TURonRR;
% we need to re-arrange G
G = [G(2,2) G(2,1);G(1,2) G(1,1)];
% values of MSC
valMSC = allT.MSC;

%=======================================
rho = abs(G(1,2)/sqrt(G(1,1)*G(2,2)));
rho2 = rho ^2;
lambda = sqrt(G(2,2)/G(1,1));
% (1*2*...*(N-1))^(1/N)
gammaNm1P = exp(sum(log(1:N-1))/N);
MSC_theo = abs(G(2,1)) ^2/G(1,1)/G(2,2);
phase_HUminusHR_degree = -atan2(imag(G(2,1)),real(G(2,1)))*180/pi;
%=========================================================================
%====================== phase ============================================
%=========================================================================

stdphibydeltamethod_theo  = ...
    180*asin(sqrt((1-MSC_theo)/N/MSC_theo/2))/pi;
statPhase = ...
    (1/sqrt(2*pi)/stdphibydeltamethod_theo)* ...
    exp(-(allT.phase-phase_HUminusHR_degree) .^2/...
    (2 * stdphibydeltamethod_theo^2));

%=========================================================================
%====================== first ratio ======================================
%=========================================================================
T = allT.T12on22;
T = T(:);

zeta = (1-rho2) ./ (2*rho*lambda * T)/gammaNm1P;
cst  = lambda/rho;

xi = ((1+lambda*lambda*(T .* T))) ./(2*rho*lambda*T);
xim1 = xi-1;

LT = length(T);
p12on22 = zeros(LT,1);

for it=1:LT
    if exist('integral','file')
        p12on22(it)   = cst * integral(@(x) myf(x,xim1(it), ...
            zeta(it),N),0,inf);
    else
        p12on22(it)   = cst * quadgk(@(x) myf(x,xim1(it), ...
            zeta(it),N),0,inf);
    end
end
stat12on22.pdf = p12on22;

valMSC = valMSC(:);
statMSC.pdf   = pdfMSC(valMSC,rho2,N);
statMSC.CI(1) = invcumulFunctionMSC(a/2,rho2,N);
statMSC.CI(2) = invcumulFunctionMSC(1-a/2,rho2,N);

%=========================================================================
%====================== second ratio =====================================
%=========================================================================
Tper = 1 ./ allT.T11on21;
Tper = Tper(:);
Gper = [G(2,2) G(2,1);G(1,2) G(1,1)];
rho = abs(Gper(1,2)/sqrt(Gper(1,1)*Gper(2,2)));
rho2 = rho ^2;
lambda = sqrt(Gper(2,2)/Gper(1,1));


zeta = (1-rho2) ./ (2*rho*lambda * Tper)/gammaNm1P;
cst  = lambda/rho;

xi = ((1+lambda*lambda*(Tper .* Tper))) ./(2*rho*lambda*Tper);
xim1 = xi-1;

LTper = length(Tper);
p11on21 = zeros(LTper,1);

for it=1:LTper
    if exist('integral','file')
        p11on21(it)   = cst * integral(@(x) myf(x,xim1(it), ...
            zeta(it),N),0,inf);
    else
        p11on21(it)   = cst * quadgk(@(x) myf(x,xim1(it), ...
```

```
            zeta(it),N),0,inf);
    end
end
stat11on21.pdf = p11on21 .* (Tper .* Tper) ;
cumul12on22 = cumsum(stat12on22.pdf)*...
    (allT.T12on22(2)-allT.T12on22(1));
stat12on22.mean = ...
    sum(stat12on22.pdf .* allT.T12on22')*...
    (allT.T12on22(2)-allT.T12on22(1));
stat12on22.median = allT.T12on22(find(cumul12on22>0.5,1,'first'));
stat12on22.cumul = (cumul12on22);

cumul11on21 = cumsum(stat11on21.pdf)*...
    (allT.T11on21(2)-allT.T11on21(1));
stat11on21.median = allT.T11on21(find(cumul11on21>0.5,1,'first'));
stat11on21.mean = ...
    sum(stat11on21.pdf .* allT.T11on21')*...
    (allT.T11on21(2)-allT.T11on21(1));
stat11on21.cumul = (cumul11on21);

if nargin == 4

    id1         = find(cumul12on22<a/2,1,'last');
    if isempty(id1)
        stat12on22.CI(1) = NaN;
    else
        stat12on22.CI(1) = allT.T12on22(id1);
    end
    id2         = find(cumul12on22>1-a/2,1,'first');
    if isempty(id2)
        stat12on22.CI(2) = NaN;
    else
        stat12on22.CI(2) = allT.T12on22(id2);
    end

    id1         = find(cumul11on21<a/2,1,'last');
    if isempty(id1)
        stat11on21.CI(1) = NaN;
    else
        stat11on21.CI(1) = allT.T11on21(id1);
    end
    id2         = find(cumul11on21>1-a/2,1,'first');
    if isempty(id2)
        stat11on21.CI(2) = NaN;
    else
        stat11on21.CI(2) = allT.T11on21(id2);
    end
end
%================================================
function f = myf(x,xim1,zeta,N)
aux_var = N*log(zeta .* x) -xim1 .* x;
f       = besseli(0,x,1) .* exp(aux_var) ;
%================================================
```

6. Polynomial fitter

```
function [xo,yo] = ...
    smoothpolyLL(xi,yi,P,degs,logtrain,logfit)
%============================================================
% Smoothing with polynomials:
% Learning could be done on a part of the input values
%     Nan are removed
% Smoothing fit could be done on a given grid
%     of values.
% Synopsis
%    [xo,yo] = smoothpolyLL(xi,yi,P,degs,logtrain,logfit)
% Rk: we assume that frequency range is included in
%   (0,Fs/2). In many spectral calculations the values
%   close to Fs/2 are very inaccurate.
%============================================================
% Inputs:
%     (xi,yi): two arrays 1D
%     P : number of intervals with different fitting
%         each segment has the same number of values
%     degs : sequence of powers, ex: degs = (0:1.5:7)
%            NON INTEGERS CAN BE USED
%     logtrain: structure with
%               .flag, 0 for linear frequency scaling
%                      1 for log frequency scaling
%               .N number of input values for
%                training
%     logfit: structure with
%               .flag, 0 for linear frequency scaling
%                      1 for log frequency scaling
%               .N number of output values
% Ex:
%  [xo,yo] = smoothpolyLL(xi,yi,2,(0:1:6),logtrain,logfit)
%  with logtrain.flag = 0, logtrain.N = 300,
%       logfit.flag = 1, logfit.N = 100
%  300 values linearly distruted on xi are divided
%       in 2 segments for training
%  Polynomial order sequence are 0,1, ...,6
%  100 values log distruted for xo are used for
%       performing yo
%============================================================

% Possible improvement: adjust derivative at the junctions

% regularization factor
lambda  = 0.000001;
%======
xi = xi(:);
yi = yi(:);
yinan = yi(not(isnan(yi)));
xinan = xi(not(isnan(yi)));

if logtrain.N>length(xinan)
    error('********* not enough data on input values ********')
end
```

```
if logtrain.flag
    [xt,yt] = logextract(xinan,yinan,logtrain.N);
else
    xt = xinan(1:logtrain.N);yt=yinan(1:logtrain.N);
end
xt=xt(:);
yt=yt(:);
xp=xt(not(isnan(yt)));
yp=yt(not(isnan(yt)));
yp0=yp(not(xp==0));
xp0=xp(not(xp==0));

N = logfit.N;
xo = zeros(N,1);
yo = zeros(N,1);
if logfit.flag
    space_xe = logspace(log10(xi(2)),...
        log10(xp0(end)),N)';
else
    space_xe = linspace(xi(2),xp0(end),N)';
end
NonP = round(N/P);
for ip=0:P-1
    ide1=ip*NonP+1;
    ide2=min([ide1+NonP,N]);
    xo(ide1:ide2) = space_xe(ide1:ide2);
    ind1_ip = find(xp0>xo(ide1),1,'first');
    ind2_ip = find(xp0<xo(ide2),1,'last');
    xp_ip = xp0(ind1_ip:ind2_ip);
    yp_ip = yp0(ind1_ip:ind2_ip);
    Hp_ip = exp(log(xp_ip)*degs)+...
        lambda*eye(length(xp_ip),length(degs));
    alpha_ip=Hp_ip\yp_ip;

    He_ip =  exp(log(xo(ide1:ide2))*degs);
    yo(ide1:ide2)=He_ip*alpha_ip;
end
% yo = filtfilt(ones(3,1)/3,1,yo);
%=======================================================
function [Fo,Ho] = logextract(Fi,Hi, Nop)
% extract log scaling values wrt Fi

N=length(Hi);
Flog = logspace(log10(Fi(2)),log10(Fi(N)),Nop)';
Fo = zeros(Nop,1);
Ho = zeros(Nop,1);
Fo(1)=Fi(1);
Ho(1)=Hi(1);
ic1=2;
for in=2:Nop
    ic = find(Fi(ic1:N)>Flog(in),1,'first');
    if not(isempty(Fi(ic1+ic-1)))
        Fo(in)=Fi(ic1+ic-1);
        Ho(in)=Hi(ic1+ic-1);
        ic1=ic1+ic;
        inlast=in;
    end
end
Fo=Fo(1:inlast);
Ho=Ho(1:inlast);
[Fo, indFo] = sort(Fo);
Ho = Ho(indFo);
%=======================================================
```

Chapter 12

Miscellaneous utilitaries

1. GeneFB

```
%============= geneFB.m ==========================
% this program built the file .mat which consists
% of thecharacteristic of the time/frequency
% pavement.
% the array of the following structures is saved
% in a file .mat
%=================================================
%     SCPperiod_sec
%     overlapDFT: typically 0.5
%     overlapSCP: typically 0
%     ratioDFT2SCP:
%     Norder
%     designname
%     windowshape
%     Wlow_Hz
%     Whigh_Hz
%=================================================
% the length in second of the FFT is :
%          SCPperiod_sec/ratioDFT2SCP
% If overlapDFT=0.5, and if ratioDFT2SCP=5,
% the number of FFTs to perform the SCP is 9
% If overlapSCP=0, the next SCP is performed by shifting
% the FFT block with 9
%
%=================================================

clear

commandsave = 'save filtercharacteristics/filtercharacteristics filtercharact';

allcolors = ['g.';'y.';'m.';'r.';'k.';'b.'; ...
    'rx';'yx';'mx';'rx';'kx';'c.';'k.';'r.';'c.';'m.';'g.';...
    'b.';'k.';'r.';'c.';'m.';'g.';'k.'];

M                               = 5;
Pfilter                         = 6;
listofperiods                   = [1000,800,350,150,80,30];
listOrder                       = [2 2 2 2 3 4];
filtercharact = struct;

%==== Remark ====================================
% due to some constraints in programming
% the value filtercharact(iP).overlapSCP must be
% 0, 1/M, 2/M, 3/M, ..., M-1/M which corresponds to
% the frontier of the FFT block.
% In pratice we advise to take 0

for iP = 1:Pfilter
    filtercharact(iP).SCPperiod_sec  = listofperiods(iP) ;
    filtercharact(iP).overlapDFT = 0.5;
    filtercharact(iP).overlapSCP = 0;
    filtercharact(iP).ratioDFT2SCP = M;
    filtercharact(iP).Norder  = listOrder(iP) ;
    filtercharact(iP).designname     = 'butter';
    filtercharact(iP).windowshape    = 'hann';
    filtercharact(iP).overlapDFT     = 0.5;
end
filtercharact(1).Wlow_Hz        = 0.005;
filtercharact(1).overlapDFT     = 0.5;
TFFT_Pfilter                    = filtercharact(1).SCPperiod_sec/M;
for iP=2:Pfilter
    %     filtercharact(Pfilter).SCPperiod_sec  = filtercharact(Pfilter-1).SCPperiod_sec*0.5;
    TFFT_Pfilter                    = filtercharact(iP).SCPperiod_sec/M;
    filtercharact(iP).Wlow_Hz       = (1/TFFT_Pfilter)/0.2;
    filtercharact(iP-1).Whigh_Hz    = 1.1*filtercharact(iP).Wlow_Hz;
    filtercharact(iP).Wlow_Hz       = (1/TFFT_Pfilter)/0.08;
    filtercharact(iP-1).Whigh_Hz    = filtercharact(iP).Wlow_Hz;
    filtercharact(iP).Wlow_Hz       = (1/TFFT_Pfilter)/0.2;
    filtercharact(iP-1).Whigh_Hz    = 1.1*filtercharact(iP).Wlow_Hz;
end
filtercharact(iP).Whigh_Hz          = 8;

scal1 = 1 ./ ([filtercharact(:).SCPperiod_sec] .* ...
    ([filtercharact(:).Whigh_Hz]-[filtercharact(:).Wlow_Hz]));

scal2 = 2 ./ ([filtercharact(:).SCPperiod_sec] .* ...
    ([filtercharact(:).Whigh_Hz]+[filtercharact(:).Wlow_Hz]));

Fs_Hz                = 20;
Lfft                 = 16*2048;
frqs_Hz              = (0:Lfft-1)*Fs_Hz/Lfft;
%========== plots of the response
```

```
figure(1)
clf
%========= end

for ip=1:Pfilter
    subplot(121)
    [filnum,filden] = butter(filtercharact(ip).Norder,...
        2*[filtercharact(ip).Wlow_Hz ...
        filtercharact(ip).Whigh_Hz]/Fs_Hz);
    Hf = abs(fft(filnum,Lfft) ./ fft(filden,Lfft));
    semilogx(frqs_Hz, 10*log10(Hf),allcolors(ip),'linew',2)
    hold on
    set(gca,'xlim',[1e-3 10])
    set(gca,'ylim',[-20 2])
    set(gca,'color',[1 1 1]*0.9)
    grid on
end
hold off
set(gca,'fontname','times','fontsize',12);
xlabel('frequency - Hz')
ylabel('gain - dB')
loc = get(gca,'position');
% bar graphics, length proportional to the duration
loc(1)  = loc(1)+0.4;
loc(2)  = 0.92;
highbar = 0.12;
for ip=1:Pfilter
    switch ip
        case 1
            lenbar  = (filtercharact(ip).SCPperiod_sec)/2200;
            locx = loc(1)+loc(3)*0.12-0.14;
        case 2
            lenbar  = (filtercharact(ip).SCPperiod_sec)/2200;
            locx = loc(1)+loc(3)*0.12;
        case 3
            lenbar  = (filtercharact(ip).SCPperiod_sec)/1800;
            locx = loc(1)+loc(3)*0.12+0.6;
        case 4
            lenbar  = (filtercharact(ip).SCPperiod_sec)/1700;
            locx = loc(1)+loc(3)*0.12+0.7;
        case 5
            lenbar  = (filtercharact(ip).SCPperiod_sec)/1700;
            locx = loc(1)+loc(3)*0.12+1;
        case 6
            lenbar  = (filtercharact(ip).SCPperiod_sec)/1700;
            locx = loc(1)+loc(3)*0.12+1;
    end
    midy    = (filtercharact(ip).Whigh_Hz+filtercharact(ip).Wlow_Hz)/12;
    loc(2)  = loc(2)-highbar;
    subplot('position',[loc(1) loc(2) lenbar highbar])
    set(gca,'xtick',[],'ytick',[],'box','on')
    tt=sprintf('%i seconds - [%4.2f %4.2f] Hz',filtercharact(ip).SCPperiod_sec,...
        filtercharact(ip).Wlow_Hz,...
        filtercharact(ip).Whigh_Hz);
    text(locx,midy*0.03+0.355,tt,'fontname','times','fontsize',14)
    set(gca,'color',allcolors(ip))
end
set(gca,'fontname','times','fontsize',12);
% xlabel('time of stationarity - frequency bandwidth')
HorizontalSize = 26;
VerticalSize   = 8;
set(gcf,'units','centimeters');
set(gcf,'paperunits','centimeters');
set(gcf,'PaperType','a3');
% set(gcf,'position',[0 5 HorizontalSize VerticalSize]);
set(gcf,'paperposition',[0 0 HorizontalSize VerticalSize]);

set(gcf,'color', [1,1,0.92]);
set(gcf, 'InvertHardCopy', 'off');

%
%         print -depsc -loose ../../textes/6distConjointHMSC/filterbank.eps
%     !epstopdf ../../textes/6distConjointHMSC/filterbank.eps
%     !rm ../../textes/6distConjointHMSC/filterbank.eps
%      print -depsc -loose ../../slides/7sessionmeeting/filterbank.eps
%         print -depsc -loose ../../slides/6slidessummary01072915/.eps
%     !epstopdf ../../slides/6slidessummary01072915/filterbank.eps
%     !rm ../../slides/6slidessummary01072915/filterbank.eps
%
%     printdirectory  = ' ../slidesITW2015/';
%     fileprintepscmd = sprintf('print -depsc -loose %sfilterbank.eps',printdirectory);
%     fileeps2pdfcmd  = sprintf('!epstopdf %sfilterbank.eps',printdirectory);
%     filermcmd       = sprintf('!rm %sfilterbank.eps',printdirectory);
%     %
%       eval(fileprintepscmd)
%         eval(fileeps2pdfcmd)
%         eval(filermcmd)


%==== parameters for TeX
tabFB = [];
for ip = 1:Pfilter
    tabFB = [tabFB ...
        sprintf('$[%4.3f-%4.2f]$&$%i$\n%s %s',...
        filtercharact(ip).Wlow_Hz, ...
        filtercharact(ip).Whigh_Hz,filtercharact(ip).SCPperiod_sec ,'\\ \hline')];
end
tabFB
%====
eval(commandsave)

filtercharact
```

2. execGPARSE

3. convertCSS2matlab