Bootstrap Signal Processing: doing the impossible?

Carlos Alejandro López Molina

A Degree Thesis
Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya

In partial fulfillment
of the requirements for the degree

Engineering of Telecommunication Technologies and Services

Advisor
Prof. Jaume Riba Sagarra

Barcelona, June 2018

Acknowledgements

I would like to express my sincere gratitude to several people, without whom this project would not have been possible. The first one of them is Jaume Riba, my project supervisor, who always encouraged me to keep learning and keep pursuing bigger objectives while still being cautious, so this project could follow the right path. He made me enjoy and learn a lot while doing this project. I would like to make a special mention on Ferran de Cabrera, with whom we had interesting discussions in several meetings and also for the help he handed in performing some heavy simulations. Finally, I also appreciate the help of gAGE reseach group, who willingly answered me some of my questions about GPS PPP satellites. Of course, I am grateful for the support of my friends and family during this 4 years.

Abstract

Classical signal processing techniques are developed under prior statistical knowledge of the kind of data we are processing. Unfortunately, one does not know too much about reality in practice, therefore inferring information about the statistical behaviour of our data is needed in addition of processing. To properly infer information about the statistics, one may need more than one realization of the experiment, although this is not possible in some real-life environments, as one may have just a single realization of the random process or low amount of available samples. If one would like to tackle both issues at once using classical signal processing approaches, it may lead to an almost *impossible* problem. This is where Bootstrap statistical inference shines, which suits perfectly this kind of problems.

Moreover, we want to fuse all available data, which comes from different sources, to improve our knowledge of the working environment and eventual accuracy in further operations, such as estimating some parameter. However, there is a risk of fusing too much corrupted data without taking into account how contaminated a data set is, so the integrity of the final estimation gets compromised. Again, we will still consider small amount of data available to tackle this issue.

In reponse to the mentioned problems, the purpose of this project is to explore and analyze the potentials of the Bootstrap techniques. In particular, we will focus on the issues of data integrity and getting benefits from data redundancy in Precise Point Positioning receivers, whose context suits perfectly this kind of framework. Studying data integrity can be considered equivalent to detecting whether our data has been affected by the most harmful disturbances, such as scintillation or related effects, and profiting from data redundancy can be interpreted as the optimal estimation of a certain parameter that comes from the received signal, which comes from several satellites.

Nomenclature

GNSS	Global Navigation Satellite System
PPP	Precise Point Positioning
$\mathbf{x}$	Column vector
$\mathbf{A}$	Matrix
$()^T$	Transpose operator
$\mathbf{1}_N$	$N \times 1$ all ones vector
$\mathbf{0}_N$	$N \times 1$ all zeroes vector
H_0	Null Hypothesis
H_0	Positive Hypothesis
P_D	Detection probability
P_{FA}	False alarm probability
i.i.d	Independent and identically distributed
WSS	Wide Sense Stationary
MVUE	Minimum Variance Unbiased Estimator

List of Figures

Contents

1 Introduction

1.1 General context

GNSS Satellites

The inspiring context for this project is the one depicted by GNSS PPP satellites [1], where the determination of position and receiver's clock consists on carrier phase observations to be able to achieve milimetrical precision. One of the biggests issues in this kind of data acquisition, is the problem of dealing with cycle-slips due to scintillation, receiver clock jumps and satellite-receiver dynamics, among other disturbances. Among these disturbances, cycle slips are the kind of disturbances that motivates this project. Cycle slips are caused by scintillation which is due to the scattering and modifications of radio signals by reason of disturbances on concentrations of varying densities of free electrons that can be found throughout the ionosphere [12]. Therefore, in this project we will propose methods to tackle this kind of disturbances, but we will follow a general path, so the methods can be applied to other environments.

The first one is the one, explained in section 2, that is focused on detecting the *integrity* of the received data by previously estimating the frequency of the received signal which can also be used for further processing (i.e. doppler analysis), from where integrity, in this context, means the property of not having any kind of disturbance in the measured data. On the second algorithm, shown in section 3, we will have as an ultimate objective, cancelling cycle slips. Finally, the last algorithm in section 4, is the integration of the previous algorithms, so we will be able to avoid or cancel this kind of disturbances.

However, the third algorithm is also implementing an optimal Data Fusion framework which fits this GNSS satellites environment. The mentioned framework is suitable because generally in navigation, a minimum of four satellites on the line of sight are needed to deliver a positioning solution. Usually, these satellites are chosen from a set of visible satellites on the basis of the signal to noise ratio of the received waveforms. However, even if the selected satellites have sufficiently high signal to noise rations, it could possibly be affected by cycle slips. Then, if a satellite affected by cycle slips is erroneously selected as candidate from which deliver the navigation solution, this effect will be translate to a poor quality of position determination, and thus losing the milimetrical precision.

Data Fusion

This is the description of the framework we will be using during this project especially concerning the nomenclature, which is based on [2]. On several Signal Processing environments, one could have several sources of information that come from different antennas, sensors, experiments or other type of sources. In those kind of environments, it is reasonable to expect that a single source of information cannot contain the sufficient amount of information to fully depict the aforementioned environment. With this idea in mind, the concept of Data Fusion makes an appearance.

As defined in [2] and reformulated to fit a typical estimation scenario in Signal Processing, Data Fusion is the analysis of several data sets such that each data set contributes with an added value that cannot be obtained from combining the remaining data sets, so it provides a proper estimation of the desired parameter or even improve this estimation. In this reformulation, there are implicitly several key concepts that will be explored through this document, but first there is a need to introduce and relate these concepts.

One of the basic concepts is the *modality* which consists on each acquisition method that gives a data set as an output, so from our point of view, referencing a given modality is equivalent to reference its output data set. As we start to link several modalities, we consider that our framework has become *multimodal* and that is where Data Fusion plays an important role. A key feature of multimodality is the *complementarity* between every single modality, which is the added value that has been metioned, that will be related to statistical independence between different modalities in section 3. Finally, the key concepts in Data Fusion are the ones related to what has been mentioned as *proper estimation* or *improved estimation*, which are the so called *diversity* and *redundancy*.

In the case of diversity, it has been heuristically defined in [2] as the property of Data Fusion that enhances the estimation of the parameter, or *latent variables*, which cannot be achieved just using a single modality.

However, this is somehow obvious in a Data Fusion environment and needs to be reformulated so we can have a proper contrast between diversity in Signal Processing and in Communications. Last but not least, redundancy is something we will make profit in section 3, which consists on having some common information between modalities, yet maintaining a certain complementarity, so we can provide a better estimation.

Our main challenge to face in Data Fusion is to face *heterogeneous* noise on the multimodal data, which means having different noises with different powers for each single modality. Having heterogeneous noise is an issue when one is not implementing an optimal data fusion scheme, in a sense that if there is no mechanism to cope with these kind of noise,

Main scope

The main scope of this project is to analyze and explore Bootstrap Signal Processing in a way that it implements the Data Fusion framework in environments that are similar to the GNSS PPP context and being able to test whether our data has been affected by disturbances.

The most natural flow of tackling these kind of problems is, firstly, to estimate the quality of a given estimator, in order to be able to detect when a given outlier phenomena has been ocurred, which we can relate to the issues of providing integrity to a set of estimations and not having enough data to be able to perform this estimation, which one can found the main assumptions for these problems and the proposed solution in section 2. Secondly, we must be able to exploit the available redundancy in our multimodal environment, again, under the conditions of not having enough data to be able to properly estimate the needed parameters to perform this data fusion. The key that links these problems to an acceptable solution, which is *la raison d'être* of this project, are the so called Bootstrap Techniques. Bootstraping consists on a computer-intensive resampling technique, with the main aim to get a better approximation of the sampling distribution of a statistic, such as the variance, p-values, confidence intervals,... [11]

The approaches that have been presented in sections 2, 3 and 4 are original to this project, which are based on several reference papers such as [3] or [4]. In order to have a general idea of what will be implemented in each section and which are the relevant context, Figure 1 can be used as a useful guide.

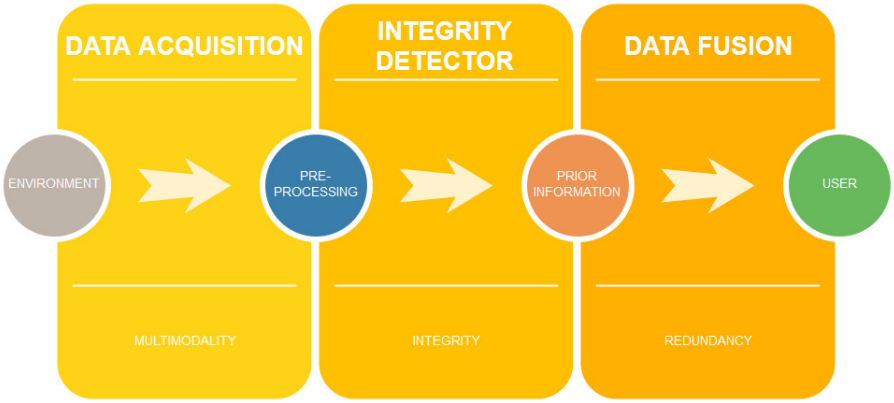


Figure 1: General Summary

1.2 Basic Tools

Throughout the documents sections where there is some algorithm explanation, we will be constantly using Bootstrap techniques to face the proposed challenges. In ordert to make it easier, we will start by explaining all the basic techniques we will be using, so there is a general reference for this document.

As a special remark to these basic tools, there is a Bootstrap Toolbox developed by the inspiring paper's author [3] of this project, which joins all these kind of techniques, and can be found in [7].

1.2.1 Bootstrap variance estimate

Let's start by presenting the basic idea for estimating variance using Bootstrap resampling, which is shown in a single *modality* environment (no redundancy). Consider as reference the problem of estimating the mean μ in a simple model, where the total number of available samples is N.

$$\mathbf{y} = \mu\mathbf{1} + \mathbf{w} \tag{1}$$

In this setting, we assume no knowledge of the noise statistics, with the exception of the samples being statistically independent between each other. Then, we choose the sample mean as a robust estimator:

$$\hat{\mu} = \frac{1}{N}\mathbf{y}^T\mathbf{1} \tag{2}$$

Let $\mathbf{y}_l$ be a resampling with replacement of $\mathbf{y}$ [3], with dimentions $L \times 1$. This resampling is obtained as follows: Let q be a uniform random variable from 1 to N. Let a resampling index vector $\mathbf{r}_l$ obtained from N independent realizations of the random variable q. Then, using Matlab notation, we have:

$$\mathbf{y}_l = \mathbf{y}[\mathbf{r}_l] \tag{3}$$

We do that L_σ times (l goes form 1 to L_σ) and obtain L_σ resampled data vectors. From them we obtain L_σ bootstrap sample mean estimates:

$$\hat{\mu}_l = \frac{1}{N}\mathbf{1}^T\mathbf{y}_l \tag{4}$$

and estimate the variance of sample mean from bootstrap estimates as:

$$\hat{\sigma}^2_{\hat{\mu}} = \frac{1}{L_\sigma - 1}\sum_{l=1}^{L_\sigma}(\hat{\mu} - \hat{\mu}_l)^2 \tag{5}$$

It was possible to do this resampling in a straight forward way because of the assumption of independency between noise sampling. In the case of correlated data, block resampling techniques are formally needed. However, it was not required, as it could be avoided where there is some *need* to apply it, which is section 2.1.

1.2.2 Bootstrap p-value estimation

Basic Bootstrap estimation implicitly estimates the CDF/PDF in order to compute all necessary statistical parameters, such as in [3] or [4]. Hovever, we need to highlight some features to be able to estimate the upper/lower tail p-value, which will be necessary especially in the Hypothesis testing section.

Now, taking advantage of the previous problem, let's assume that we would like to estimate the upper-tail and also the lower-tail p-value of the previous mean estimator. We can start from obtaining L_{CDF} bootstrap sample mean estimates as in (4), so we can store those sample mean estimates in a vector as:

$$\hat{\boldsymbol{\mu}} = [\hat{\mu}_1, ..., \hat{\mu}_L] \tag{6}$$

The next step, will be to order the vector $\hat{\boldsymbol{\mu}}$ such as its elements satisfy $\hat{\mu}_1 < \hat{\mu}_2 < ... < \hat{\mu}_L$, therefore the desired p-values, p_L for the lower-tail p-value and p_U for the upper-tail p-value, can be computed as simply as:

$$p_L = \hat{\boldsymbol{\mu}}[p_L L_{CDF}] \qquad p_U = \hat{\boldsymbol{\mu}}[(1 - p_U)L_{CDF}] \tag{7}$$

Note that after the sorting, $\hat{\boldsymbol{\mu}}$ is approximately equivalent to the inverse CDF and thus, this is a non-parametrical approach of estimating the CDF.

2 Bootstrap Hypotesis Testing

In this block, we will introduce our own proposed *hypotesis test* in the detection of disturbances when estimating the frequency based on Kay's frequency estimator [5] and Bootstrap resampling techniques. As it has been mentioned in the Introduction, this frequency estimation can be helpful in further processing in a GPS PPP environment, for doppler processing as an example, and it enables the user to be able to determine whether the effects of scintillation have been occurred or not due to some of its phase disturbances nature. Thus, the main scope of this hypothesis test is to determine the integrity of a certain data set, which can be used to apply a Data Fusion algorithm in a GPS PPP environment.

Motivation

From one hand, we have a set of data that carries information about frequency, from which we know limited information about its statistics, such as its distribution but limited in a sense that we do not know the parameters that determine such distribution. On the other hand, there is a *set of disturbances* that could potentially affect our data, which can have very different natures such as a strong bias or an unexpected increase of variance that can destroy the information we want to get from the estimation of the carrier frequency.

One of the biggest issues when facing real-life detection applications is facing a problem where one has little to no information about the statistics of the different hypotesis, where we note H_0 as the null hypotesis which is the one that it is usually easier to find its statistic model data and H_1 as the *positive* hypotesis, or also the detection itself, which usually its statistic model is unknown as contraposed to H_0. Thus, if we are considering to detect a variety of disturbances, the modelation of the resulting hypotesis may be complicated but feasible, but we may have troubles when deriving a threshold from the *Test Function*, and this is where Bootstrap Hypotesis Test as proposed in this section can be useful, as it inherently computes the necessary parameters or functions to be able to detect those aforementioned disturbances.

Basic Tools

- Kay's frequency estimator, and more specifically, Kay windows.
- Bootstrap techniques, which consists of Bootstrap resampling and p-values estimation

2.1 Initial Set-up. Kay's Frequency estimation

Firstly, we will consider an accurate estimator of the frequency that gets as close as possible to the CRLB. The ideal candidate that permits to apply any kind of Bootstrap technique in a frequency estimation, is the Kay's estimator of the frequency [5], and thus from now on we will consider this simplified model of the received signal:

$$s_{Rx}[n] = Ae^{j(2\pi \frac{F_0}{F_s} n + \phi)} + w[n] \tag{8}$$

this signal consists on N samples of a pure complex tone immersed in AWGN, being A the amplitude of the tone, F_0 the tone frequency, F_s the sampling frequency which satisfies the Nyquist criterion, ϕ the phase offset and $w[n] \sim CN(0, N_0)$ where N_0 is the noise power and all noise samples are i.i.d. Now, let's assume it has a sufficiently high SNR and following a similar procedure in order to simplify the phase expresion as in [5], we can get an approximated linear model to the received signal's phase:

$$phase(s_{Rx}[n]) = p[n] \approx 2\pi \frac{F_0}{F_s} n + \phi + \bar{w}[n] \triangleq \theta n + \phi + \bar{w}[n] \tag{9}$$

where now $\bar{w}[n] \sim N(0, \bar{N}_0)$, $\bar{N}_0$ relates to the previous parameters as $\bar{N}_0 = \frac{N_0}{2A^2}$ and its role is the *variance of the phase disturbance* due to the receiver's noise under the assumption of H_0, which will be further discussed in the integrity detection section. Finally, θ is the parameter that we want to estimate, which is related to F_0 as $\theta = 2\pi \frac{F_0}{F_s}$.

The next step still follows Kay's approach to estimate the received frequency. It consists on computing the discrete derivative with respect to n of (9), so the undesired terms, such as ϕ, are cancelled out and the whole estimation problem gets simplified as the ML estimation of a parameter under AWGN. Note that this step reduces the total number of samples from N to $N-1$ and correlates the noise between the samples $n-1$, n and $n+1$.

$$diff(p[n]) = d[n] = \theta + \omega[n] \tag{10}$$

this correlation implies that if one is interested in estimating some statistics from $d[n]$, such as the $N-1 \times N-1$ correlation/covariance matrix, using Bootstrap techniques, it is necessary to apply a block resampling instead of a independent data resampling in order to emulate this correlation [3]. However, if one is just interested in estimating the variance of $d[n]$, there is no need to use block resampling as it just brings unnecessary extra complexity (block length, more operations,...) in an estimation that, as a final objective, does not need information about the dependence structure. On the following paragraphs a method to avoid this need of using block resampling in the estimation of variance, in this context, will be shown.

Reaching up to this point, we need to find the MVUE of θ. There are several ways to find this estimator, such as finding the ML estimator, remember that if MVUE exists, ML and MVUE are equivalent, or using Fischer's matrix derivation. Our proposed approach to find this estimator that simplifies the estimation of the estimator's behaviour, which relates to the *integrity* of the estimation. The mentioned approach consists on an estimator of the form $\hat{\theta} = \mathbf{g}^T\mathbf{d}$, where $\mathbf{g}$ is such that this estimator is unbiased and minimizes its variance, and is $P \times 1$. It can be summed up as the following optimization problem. (The whole derivation is proved in the annex)

$$arg\,min_{\mathbf{g}}\ \mathbf{g}^T\mathbf{C}_\omega\mathbf{g} \quad subject\ to \quad \mathbf{g}^T\mathbf{1}_{N-1} = 1 \tag{11}$$

where $\mathbf{C}_\omega$ is the $P \times P$ covariance matrix of $\omega[n]$, which in this case, $P = N-1$, so we will have more freedom in further steps. The solution to this optimization problem is:

$$\mathbf{g}_{opt} = \frac{\mathbf{C}_\omega^{-1}\mathbf{1}_P}{\mathbf{1}_P^T\mathbf{C}_\omega^{-1}\mathbf{1}_P} \tag{12}$$

as it has been mentioned, this estimator also corresponds to the MVUE estimator of θ, but this approach shows some properties that are useful and make simpler to compute the estimation of the variance of the frequency estimator (or in this case, θ), denoted as $\hat{\sigma}_\theta^2$. Let's show how this can be useful to estimate the variance of the estimator. Here we got the expression of the analytic variance of the aforementioned estimator:

$$\sigma_\theta^2 = E\{(\hat{\theta} - \theta)^2\} = \mathbf{g}_{opt}^T\mathbf{C}_\omega\mathbf{g}_{opt} \overset{CRLB}{=} \frac{1}{\mathbf{1}_P^T\mathbf{C}_\omega^{-1}\mathbf{1}_P} \tag{13}$$

from this expression, we can obtain the estimated version just by introducing the estimation of $\mathbf{C}_\omega$ and this enhances to obtain an alternative expression that is simpler in terms of bootstrap resampling, as it does not need that the correlation structure is emulated with block resampling. Applying the simple bootstrap resampling to enhance the estimation by enlarging the number of residuals samples, we can obtain $\hat{\sigma}_F^2$ as:

$$\hat{\sigma}_\theta^2 = \mathbf{g}_{opt}^T\hat{\mathbf{C}}_\omega\mathbf{g}_{opt} = \mathbf{g}_{opt}^T\left(\frac{1}{L_\sigma(P-1)}\sum_{i=1}^{L_\sigma}\mathbf{r}_i\mathbf{r}_i^T\right)\mathbf{g}_{opt} = \frac{1}{L_\sigma(P-1)}\sum_{i=1}^{L_\sigma}|\mathbf{g}_{opt}^T\mathbf{r}_i|^2 \tag{14}$$

where $\hat{\mathbf{C}}_\omega$ is the estimation of $\mathbf{C}_\omega$, $\mathbf{r}_i$ are the i-th $P \times 1$ resampled residuals of $d[n]$ and L_σ are the total Bootstrap realizations. The mentioned residuals are just resamplings of the following vector

$$\mathbf{r} = \mathbf{d} - \hat{\theta}\mathbf{1}_P \tag{15}$$

where now $\mathbf{d}$ is the $P \times 1$ vector that contains P values of $d[n]$. A remarkable condition between these parameters is $B > P$ and it has 2 interpretations: It should be satisfied so the *virtual* Covariance matrix has full rank or also, is the minimum required value so we do not have any simulation errors in this case, although is recomended to take a higher value. Also, the assumption of the different resamplings being linearly independent,

The last expression simplifies the estimation of $\hat{\sigma}_\theta^2$ as it is not necessary to directly compute $\hat{\mathbf{C}}_\omega$ to obtain it, we just need the residuals and $\mathbf{g}_{opt}$. There's still an error associated to this estimation if we just apply independent Bootstrap resampling, considering the amount of correlation in this scenario, but we can neglect it, as it will be discussed in the next section. In this scenario, we can accept this amount of error, as we are interested in applying a hypotesis testing, not in estimating precisely the estimator's variance, σ_θ^2.

2.2 Hypotesis testing

Now that we are able to estimate θ, we would like to detect whether an extra undesired disturbance, other than the receiver's noise, affects our signal with the condition of being able to determine a threshold just using previously validated data. These disturbances can have of very different natures and can have many different behaviours in our model. To simplify this issue, we will model and process disturbances that are manifested as either an increment of variance or an added bias in the signal model in (9), therefore, the hypothesis that we are considering are the following:

$$H_0: \qquad p[n] = \theta n + \phi + \bar{w}[n]$$

$$H_1: \qquad p[n] = \theta n + \phi + \bar{w}[n] + c[n]$$

where the parameters that compose H_0 were explained in the previous section. The new parameter in H_1 is the sequence $c[n]$, which can be modeled in different ways, but we consider 2 independent components regarding an increment of variance or added bias, $c[n] = \tilde{w}[n] + z[n]$. The first of them, $\tilde{w}[n]$, is the term associated to an increment of variance which can be modeled as a random variable i.i.d, independent of $\bar{w}[n]$, with satistics $N(0, D)$ and the second one, $z[n]$, is the one related to added bias in different instants, which can be interpreted as our signal being affected to sudden *steps* or *jumps* that can be perfectly characterized with a *jump probability*, P_j, and the *jump's height*, h. These jumps results into impulsive noise when applying the discrete derivative as in (10).

The combination of these 2 components can be useful in different contexts, such as the already mentioned scintillation issues in GNSS receivers, where we can find a suddent increments in variance, or also a measurament error in some other contexts, and also the so called cycle slips, which are the ones that we model as *steps*.

Assumptions

- Having prior knowledge about the models under H_0 (includes statistic distributions).
- No prior knowledge about the values of the parameters.
- We have a *training* set of data, which is known to be under H_0, to initialize the algorithm.
- No prior knowledge about the statistics under H_1 (It will be explained).

In a hypothesis testing problem, one should start by defining a criterion of finding a suitable *Test function* $T(X)$ for the particular problem we are facing. Ideally, in a given context where these models make some sense, one of which could be the already mentioned GPS PPP, the optimum approach would be applying a *Reversed Neyman-Pearson Criterion* that consists on minimizing P_{FA}, given a certain P_D, in order to derive the mentioned Test Function. In such way, we can assure a certain degree of data integrity (P_D), which is necessary for further processing steps as detecting scintillation/cycle slips can be helpful for either cancellating this phenomena or *disconecting* the affected satellite.

Under these assumptions, which are reasonable in an actual environment, we do not have all the necessary information to apply a suitable classical approach to this problem, as we are assuming we do not have enough information of the statistics under H_1. Instead, we will make use of the information that can be extracted from statistical moments to extract both the Test Function and the *Test Threshold*, T_H. Taking into account

that we have set our data to estimate θ, directly related to F_0, and we know that under H_0 equation (13) tends to be equal to the CRLB, which in the case of frequency it is:

$$CRLB = \frac{1}{\mathbf{1}_P^T \mathbf{C}_\omega^{-1} \mathbf{1}_P} \propto \frac{1}{P^3 SNR} \tag{16}$$

where $\propto$ indicates the $CRLB$ tendency with respect to those parameters. Then, it is reasonable to set $T(X) = \hat{\sigma}_\theta$ as the considered set of disturbances cause an unexpected increment of variance, especially the *step* disturbance as a single step can break the model, so it also breaks the $\frac{1}{P^3}$ behaviour.

2.2.1 Threshold estimation

Now that we have estimated the frequency in an optimal way under H_0, we now want to test whether in a new block of N samples in the next time window $k+1$, denoted as $\mathbf{p}_{k+1}$ is affected by disturbances, which is the same as asking if the hypothesis H_0 holds in the given set of data. We will consider $T(X) = \hat{\sigma}_\theta$, which is the estimation of the variance of $\hat{\theta}$ under the assumption that H_0 holds. In the newest set of data we expect this variance to be much higher, if the data comes from an environment where the hypothesis H_1 holds than in a set of data that is known to be under H_0. It is assumed a Wide Sense Stationary process through all time windows that fit H_0. There are different ways of implementing a theshold estimation/hypothesis testing with using Bootstrap techniques, such as the ones described in [4] or [8], however the following hypothesis test will be based on the former, as it is the simplest, but effective, way to implement it.

First, we need to compute the upper tail empirical p-value, to determine the threshold T_H, in order to fix a P_{FA} to α which can also be called the level of significance of this hypothesis, but note that we cannot determine how P_D will behave as we are under a Blind data context. It is needed much more data to be able to fix P_D than to fix P_{FA}, which just needs to have a statistically rich set of data to be determined. To compute the desired p-value, we just need to estimate the Cumulative Density Function of $\hat{\sigma}_\theta^2$ as explained in section 1.2, using $\mathbf{p}[n]_{train}$ as explained in the previous subsection, using L_{CDF} Boostrap realizations of the mentioned estimator. Then, assuming $F[n]$ is the CDF of $\hat{\sigma}_\theta^2$, where n is the number of the n-th biggest realization, the threshold T_H is such as:

$$T_H = F^{-1}[ceil((1-\alpha)L_{CDF})] \tag{17}$$

where the ceil function rounds its argument to the closest integer smaller than it. Now, having computed this threshold, whenever a new set of N samples, $\mathbf{p}_{k+1}$, is available, we will just need to proceed as in the frequency estimation section, so then we can compute the residuals such as:

$$\mathbf{r} = \mathbf{d}_{k+1} - \hat{\theta}_{train} \mathbf{1}_P \tag{18}$$

Still, there are some issues that should be remarked from this algorithm. First, T_H is just an approximation to the desired p-value, so the actual P_{FA} will be as closer to α as L increases and also as N increases. The justification of this, will be explained deeply in section 3.4, but it can be advanced that L is related to a simulation error and N to a statistical error, related to the statistical richness of our data. The first one can be solved by forcing L to be a function of N, i.e. $L = 100N$, but the second one is limited by each environment. It could also be possible to increase this threshold by a certain amount, Δ_{th}, so it gives some margin to hypothesis H_1.

And finally, the other issue that has been mentioned is the fact that we are not taking care properly of P_D with this approach, so depending on the context, this approach could be futile if the statistics of H_1 are similar to the ones under H_0. An appropiate theoretical way to quantize this similarity would be the Kullback-Leibler divergence between the PDF of both hypothesis, if one could know the statistics under H_1 [9].

2.3 Simulation results

It is necessary to show the aforementioned algorithm for hypothesis testing, which in the end is testing the integrity of the data, under certain conditions to see in which environments does it suit better. In order

to better characterize the environment that we are facing in the mentioned hypothesis test, we will have to introduce a new paramenter that will determine the similarities between both hypothesis. This parameter will be called *Jump to Noise Ratio* and it will be defined as:

$$JNR \triangleq \frac{h}{\sqrt{\bar{N}_0}} \tag{19}$$

where both parameters involved were defined in previous paragraphs. For simplicity, we will get rid of the added variance in H_1 hypothesis, as it does not give any interesting insight in the solution of this problem,. Let's now recall both resulting hypothesis.

$$\begin{aligned} H_0: &\qquad p[n] = \theta n + \phi + \bar{w}[n] \\ H_1: &\qquad p[n] = \theta n + \phi + \bar{w}[n] + c[n] \end{aligned}$$

from where now, $c[n] = z[n]$, as we have already noted. The full list parameters that we will take into account in this study will be shown in the following table, with their respective default values:

Parameter	Value	Description
JNR	8	Jump to noise ratio. Critical in detection
P_j	0.01	Jump's probability for each sample
N_0	10	Normalized Noise Power
θ	10	Latent variable, related to Frequency
N	100	Number of samples in each time window
L_{CDF}	1000	Total number of values in the inverse CDF
L_σ	$10N$	Number of Bootstrap Samples
α	0.005	Level of Significance

There is something that should be noticed before getting into the simulations. For this kind of hypothesis testing a *Receiver Operation Characteristic* or related plots (P_{FA}-P_{MD} ROC, Area Under the Curve, 1-AUC,...) that are not suitable for showing the behaviour of this integrity detector. These plots are not suitable, in a sense that there are some regions in the domains of these plots that show some features that do not make sense for a usual detector, especially in the boundaries of these plots. For example, it is possible that when evaluating $P_{FA} = 1$, we do not get the expected value $P_D = 1$. This is due to the nature of the environment, remember that we consider a blind hypothesis test, where we do not know, in principle, any information about H_1 and even less about the statistics of $T(X)$ under H_1. What is more, we will not emphasize on L_σ, as this parameter is mostly related to the variance estimation, which can be found on the simulation results in section 3, as the latter is focused on Bootstrap estimation.

The performance study of this hypothesis testing algorithm will be separated into two blocks: The first block will be the analysis of the performance of P_D, where we will vary the parameters related to the definition of H_1, from which the JNR will be highlighted (remember, in the point of view of the algorithm these parameters are unknown), and more general parameters like N, so it can be quantized a certain degree of integrity of the studied set of data as a function of the mentioned parameters.

The remaining block is focused on P_{FA}. We will focus the study in α, as this is the most interesting feature of this algorithm. The main objective of this second block is to see how well does Bootstrap techniques can approximate the threshold in order to get closer to the desired value of P_{FA}.

2.3.1 Study of detection probability P_D

To study the behaviour of P_D, all the studied sets of data must be under H_1 hypothesis and we should determine first all the parameters that somehow have a potential impact in this performance figure. The first two candidates would be the ones directly related to H_1, which are P_j and JNR. The reason why these two parameters are of interest in detection probability is because essentially we are measuring a variance to

perform the hypothesis test and therefore the threshold will be proportional to $\bar{N}_0$. This fact implies that in order to study the *distance* between both hypothesis, which is essentialthe distance between $\hat{\sigma}_\theta^2$ under H_1 and $\hat{\sigma}_\theta^2$ under H_0, we will have to focus on the features that add variance in the estimation under H_1, which are the number of jumps, related to P_j , and the height of each jump with respect to the standard deviation of the noise, JNR.

As P_j is related to how many jumps are present in the testing set of data, but we are assuming that under H_1 the given set of data has at least one jump, we will ensure that there is at least one jump in the testing data because otherwise the given set of data would be under H_0.

Impact of JNR In this test, we have linearly varied JNR from 0.5 to 30 to see its impact in P_D, maintaining the other parameters in their respective default values. We have done 1000 Monte Carlo realizations in order to estimate P_D in the following way, taking into account that all the test data sets are under H_1:

$$P_D \approx \frac{1}{M_{onteCarlo}} \sum_{m=1}^{M_{onteCarlo}} I(\hat{\sigma}_m^2 > T_H) \tag{20}$$

where now $\hat{\sigma}_m^2$ is the measured estimator's variance in the m-th realization and $I(\cdot)$ denotes the indication function, which returns 1 if the argument is true, 0 otherwise. The resulting plot is the one shown in Figure 2.

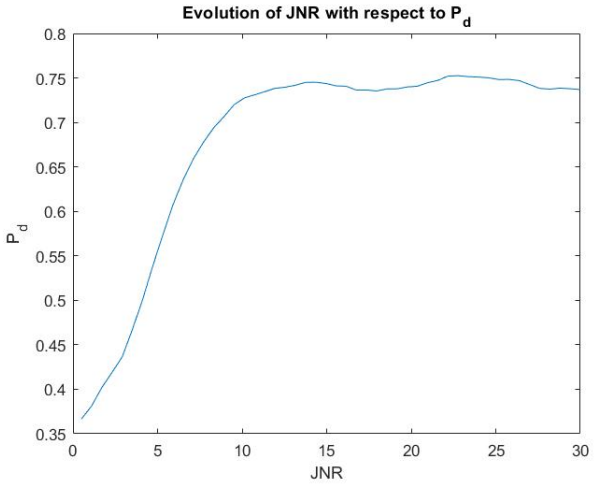


Figure 2: Impact of JNR in P_D

In this plot, it can be noted that up to a certain saturation value, JNR has a high impact in P_D, as one could have previously expect. Furthermore, this saturation value of JNR determines the operational jump's height, which can be considered to be the JNR that ensures $0.9P_{D_{MAX}}$, which in this case $JNR_{OP} = 7.5$ and $P_{D_{MAX}} = 0.745$. In conclusion, we cannot ensure a certain level of integrity for JNR below the saturation value JNR_{OP}. Note that the typical value ofJNR in sets of data affected by cycle slips is around 10.

Another feature that can be extracted from this plot is that there is a maximum P_D, for a given set of parameters excluding JNR, so if there is a need of a higher $P_{D_{MAX}}$, we shall modify those values to enhance the detection, as we will see in the following paragraphs.

Impact of P_j Now, it is time to see the number of jumps' impact in detection. To do so, we have done two simulations for two values of JNR, where we used $JNR = 10$ and $JNR = 5$, to show two different scenarios. Furthermore, we also performed a Monte Carlo simulation of 1000 realizations to estimate P_D as in (18) and we have swept P_j logarithmically between 0.001 and 0.1. The resulting plot can be found in Figure 3.

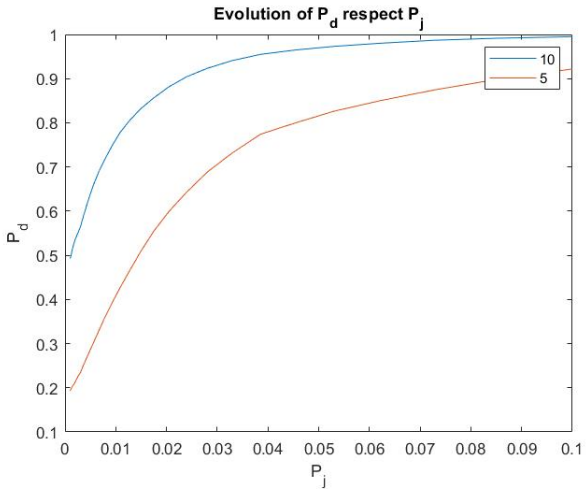


Figure 3: Impact of P_j in P_D

Unlike in JNR, the number of jumps does not sature the P_D, which means that the higher P_j, the closest to 1 gets P_D. It was expected since the larger the number of jumps, the more *impulses* will be found in the residuals, so it translates into more variance in the frequency estimation. As for the different values of JNR, we can see that they start from different lower or higher depending on its value. In addition, as $JNR = 10 > JNR_{OP}$ if we increase this value, we will not get better performance, so this should be taken into account to determine the level of integrity after this block, as this is tighly related to the specific environment.

Impact of N The last but not least important parameter to study in detection is N, which is also impactful due to being related to the statistical richness of our data set. This simulation consists on the logaritmic increment of N from 10 to 500, from where we estimate P_D as in (18), using 1000 realizations. Note that this simulation is computationally costly, so we could not extend it to larger values of N. The final result is shown in Figure 4.

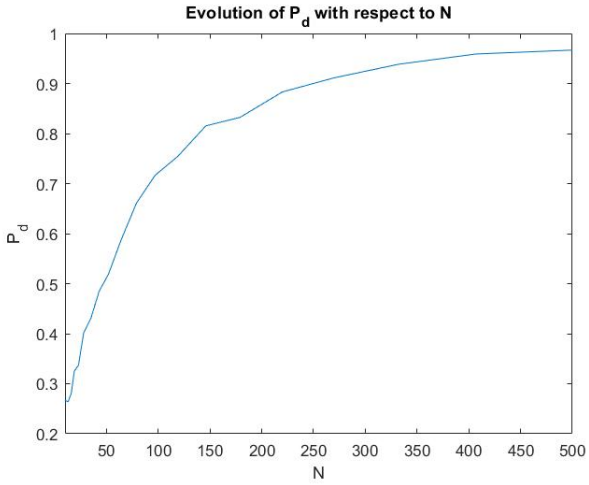


Figure 4: Impact of N in P_D

As it was expected, when N is being increased, P_D also increases, in a way it assymptotically approaches

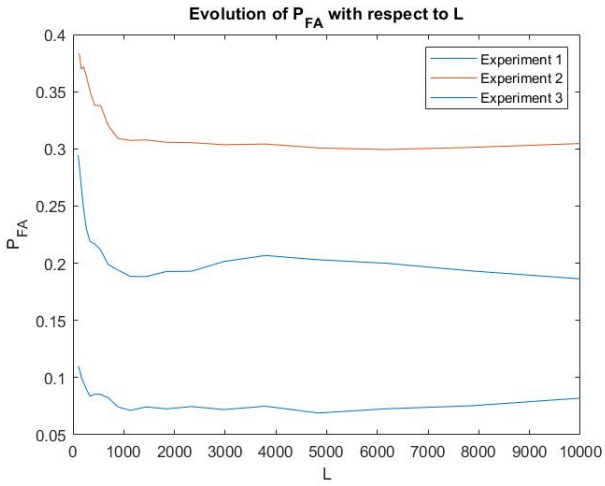


Figure 5: Impact of L in P_{FA}

1. This tendency is due, among other reasons such as the values of the remaining parameters, to the gain of statistical richness of our data, so the statistical error is inherently reduced.

2.3.2 Study of false alarm probability P_{FA}

Studying false alarm probability, in this kind of detector, is tightly related to the study of the estimation of the threshold, T_H, in a sense that it is closer to the actual p-value that we wanted to compute. The candidates that relates to this estimation are L and N, both related to the simulation error and statistical error, respectively. This step is really important, as the main objective of this detection algorithm was to fix P_{FA} for the integrity's detection for a given data set. For now on, all data sets are under H_0.

Impact of L In this simulation, we set the rest of parameters to their default values and we will just focus on L, so the experiment consists on 3 different realizations of a training data, so it is possible to see if this algorithm is more or less dependable of the training data. As it has been emphasized, and will revisit in section 3, L must be a function of N, and that is the reason why we have logarithmically swept this parameter between N and $100N$. In addition, we have estimated P_{FA} in a similar way than the P_D, by using Monte Carlo simulations as

$$P_{FA} \approx \frac{1}{M_{onteCarlo}} \sum_{m=1}^{M_{onteCarlo}} I(\hat{\sigma}_m^2 > T_H) \tag{21}$$

where now we are considering that all our data is under H_0. You can find this plot in Figure 5.

Even though it has been masked by applying a smooth function, in this plot there was more evidence of the randomness of this technique, which was somehow masked in previous figures. We can see that, for the given set of parameters, the estimation of P_{FA} is not accurate, yet we can remark that there is a tendency towards (decreasing) the desired level of significance as L increases, up to a certain point where the algorithm converged to a certain P'_{FA}.

As for this P'_{FA}, in none of these realizations have converged to the expected value of α, which is 0.005. This phenomena is tightly related to the amount of samples N, as when N increases, the statistical error is reduced, so we can estimate more accurately the estimator's variance CDF. Still, this shows that there is more room for improvement in this block.

Impact of N As in the previous simulation, we will just emphasize in N and set the rest of parameters to their default values, so we have increased logarithmically N between 10 and 500, as in the study of P_D. Again, we estimated P_{FA} as in (21), using 1000 Monte Carlo realizations. The final result is shown in Figure 6.

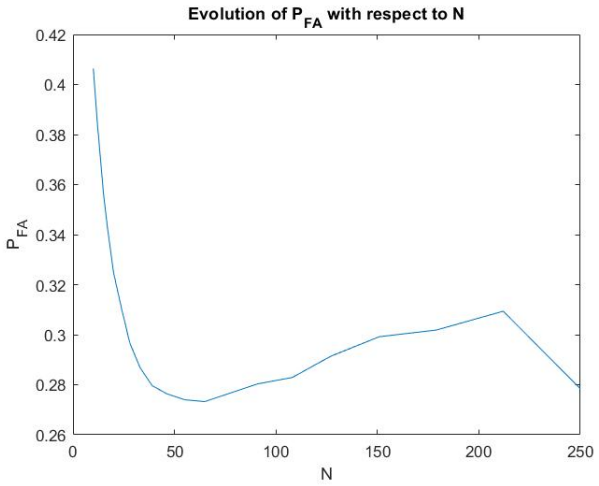


Figure 6: Impact of N in P_{FA}

This plot makes more evident the implicit randomness of this algorithm, which in this case a smoothing function was applied again to observe the general tendency. It again shows a high dependence of this performance figure on the training data set, as it could either increase a lot or being reduced a lot by just changing the training data set. Note that the updating of the training data set was necessary since we are increasing N. Nevertheless, it should be highlighted that there is a tendency towards reducing this P_{FA} as we increase N, for the same reasons that P_D tends to 1, due to the nature of this detector.

3 Bootstrap Data Fusion

Motivation

In this section we will explore the concept of redundancy in optimal data fusion when facing a multimodality environment [2] for estimating a certain parameter and from which we do not know the statistics of the available data. The main goal is to optimallly fuse the available data to yield an estimate as precise as possible by exploiting the available redundancy and diversity, so we can automatically discard or down-weight contaminated measurements. A desired feature that the system should implement is the capability of cancelling out added biases just using the available data and extra information that can be given from the detection of data integrity, in the previous section.

Being solved the issue of data integrity by the previous section 2 and that we have to be aware of added biases, another important issue in this block, will be facing with *heterogeneous noise* [2]. Heterogeneous noise has a relevance in a sense that, when fusing data without too much care, the whole latent variables' estimate is corrupted by those sources that are affected with high variances. What is more, disconecting those sources can also be harmful for the estimate due to the loss of necessary information, or also some redundancy, that can be produced by the lack of sources of data.

3.1 Initial set-up. Multimodal environment

Now, we will present an environment and an algorithm that can exploit multimodality and diversity to deliver an optimal data fusion, based on Bootstrap resamplings. We consider a case where we have redundancy in the available data with respect to the latent variable. The objective is to exploit the Bootstrap capacity of delivering integrity, so it improves the data fusion stage in forming the estimate. To be specific, we focus on a spatio-temporal model described as follows:

$$\mathbf{Y} = \mathbf{m}(\mathbf{x})\mathbf{1}_N^T + \mathbf{W} \tag{22}$$

note that time is in the horizontal domain and space is in the vertical domain, as in array processing. We assume that the columns of $\mathbf{W}$ are independent between them, as the spatial measurements come from different modalities, but not equally distributed. We also assume that the rows of $\mathbf{W}$ are i.i.d, as this could be interpreted as the noise that comes from a given sensor and thus, every independent row from this model is what we already pointed out as a *modality* [2]. As $\mathbf{W}$ is the noise term, it has mean $\mathbf{0}_N$, which is a full zeroes vector $N \times 1$.

In the model, $\mathbf{Y}$ and $\mathbf{W}$ are $M \times N$ (with $M < N$), $\mathbf{m}(\mathbf{x})$ is $M \times 1$ and $\mathbf{1}_N$ is $N \times 1$. The latent variables, denoted as $\mathbf{x}$, are $D \times 1$ with $D < M$, which is the redundancy condition. In the context of satellite navigation, we can make an interpretation of these parameters, where N is the number of observed samples (in time), M is the number of available satellites and the dimension of the latent variables, $D = 4$, which consists on the three positioning parameters: *longitude, latitude* and *altitude*; plus the temporal parameter: the received *satellite clock*. One of the key parameters of this model is the relationship between variances of different rows, which can be expressed as $\mathbf{C}_w$, the covariance matrix between different modalities, which ideally is a diagonal matrix $M \times M$, as it is in our initial assumption. It can be defined as:

$$\mathbf{C}_w = E\{\mathbf{W}\mathbf{W}^T\} \tag{23}$$

the vector $\mathbf{m}(\mathbf{x})$ is the generalized mean vector for a given $\mathbf{x}$. From now on, for simplicity, and as a first approximation to the problem, we consider a simplified linear model:

$$\mathbf{m}(\mathbf{x}) = \mathbf{A}\mathbf{x} \tag{24}$$

In positioning, this model has plenty of sense in tracking conditions, where $\mathbf{A}$ is the Hessian matrix, which is $M \times D$, or derivatives of the positioning observables with respect to the user position, which is valid up to 1 or 2 Kilometers, so we can assume a stationary process. This is similar to the approach followed by the

classical Extended Kalman Filter. For futher Bootstrap operations, the sample mean has to be estimated, so the M sample mean estimate can be expressed as:

$$\hat{\mathbf{m}} = \frac{1}{N}\mathbf{Y}\mathbf{1}_N \tag{25}$$

from the sample mean estimates, a linear reduced noisy model can be extracted and it will be useful to understand some feature that will florish in the Bootstrap estimation. This reduced model can be written as:

$$\hat{\mathbf{m}} = \mathbf{A}\mathbf{x} + \mathbf{u} \tag{26}$$

where $\mathbf{u}$ is the measurement noise associated to the sample mean estimators. The relationships between $\mathbf{u}$ and the rows of $\mathbf{W}$ can be easily be proven: The mean of the i-th component of $\mathbf{u}$ is exactly the mean of the i-th row of $\mathbf{W}$ and the $\mathbf{u}$ covariance matrix, denoted by $\mathbf{C}_u$ is:

$$\mathbf{C}_u = \frac{1}{N}\mathbf{C}_w \tag{27}$$

3.2 Classical approaches

In order to undersand why Bootstrap techniques will be applied in such way in the next section, we will start by explaining how this *Data Fusion* can be performed, having some extra knowledge of the statistics. Let's start by considering a known covariance matrix, $\mathbf{C}_u$ or $\mathbf{C}_w$, under the asymptotic approximations which is assuming that the distribution between rows of $\mathbf{W}$ or each realisation of different rows of $\hat{\mathbf{m}}$, is such as $\mathcal{N}(\mathbf{0}, \mathbf{C}_w)$ or $\mathcal{N}(\mathbf{0}, \mathbf{C}_u)$ respectively, remember that in the same row, it is assumed a Gaussian Wide Sense Stationary process. Therefore, in order to achieve the *closed form* MVUE we could just derive the Maximum Likelihood estimator:

$$arg\,max_{\mathbf{x}} \frac{1}{(2\pi)^{N/2} det(\mathbf{C}_u)^{1/2}} exp(-\frac{1}{2}(\hat{\mathbf{m}} - \mathbf{A}\mathbf{x})^T \mathbf{C}_u^{-1}(\hat{\mathbf{m}} - \mathbf{A}\mathbf{x})) \tag{28}$$

whose solution is already known and is known that the MVUE of this problem exists, which in this case is the same as a Weighted Least Squares approach, under complete knowledge of the statistics:

$$\hat{\mathbf{x}}_{CRLB} = \left(\mathbf{A}^T \mathbf{C}_u^{-1} \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{C}_u^{-1} \hat{\mathbf{m}} \tag{29}$$

as this estimator is MVUE, it achieves the Cramer-Rao Lower Bound (CRLB) for the variance, which is what it is called an efficient estimator and thus, it is the best estimation we can achieve under complete knowledge conditions, the *optimal estimation*, and the reason why we want to estimate $\mathbf{C}_u$, $\mathbf{C}_w$ or any covariance matrix related to the noise in our environment. In addition, this estimation can be seen as an oblique projection, but this idea will be further developed in a future paragraph.

Another approach, which in advance can be considered in data fusion as the *risky option*, under asymptotic approximations is assuming the previous distributions to be distributed as $\mathcal{N}(\mathbf{0}, \sigma_{\mathrm{w}}^2\mathbf{I})$ or $\mathcal{N}(\mathbf{0}, \sigma_{\mathrm{u}}^2\mathbf{I})$. Then, if we follow a similar path as in the previous assumption, which is achieving the MVUE under the previous asumption, we get as a solution the Least Squares approach:

$$\hat{\mathbf{x}}_{LS} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \hat{\mathbf{m}} \tag{30}$$

note that this case has been labeled as risky because the lack of need in estimating the variances, a single noisy source can destroy the whole estimation.

3.3 Bootstrap Techniques in Blind Data Fusion

Now that we have set the main issue in this data fusion environment, the next step should be extend the previous classical approaches to a *blind* context, in which we do not have any information about the statistics of the received signals, and enhancing the estimation of the critical parameters, which in this case are $\hat{\mathbf{C}}_u$ or

$\hat{\mathbf{C}}_w$, with Bootstrap Techniques. The fact that Bootstrap Techniques are proposed to face this problem, is that they are extremely useful in environments that there are not much available data, which means that N will be bounded.

There are several ways to resample the data, but all of them fullfil a key requirement: It should, somehow, emulate another realization of the *experiment* we are interested in, so there is enough data to estimate. On the following paragraphs, 2 variations will be shown, as we consider that they are the most suitable in this scenario.

3.3.1 Straight forward bootstrap resampling

This is the straight forward Bootstrap resampling option. It consists on the usual Bootstrap Resampling approach [3] which consists on calculating the residuals (depending on the problem it could be the *estimated* noise, modeling errors,...) using the mean estimation or other kind of estimations from the data. In this particular case we are facing, the residuals are *estimations* of noise. Using the resampling idea explained in the first block:

$$\hat{\mathbf{R}} = \mathbf{Y} - \hat{\mathbf{m}}\mathbf{1}_N^T \tag{31}$$

using the resampling with replacement idea already explained in Basic Tools (1.2), we obtain L estimates of the Residuals matrix, by resampling the received data columns, denoted as $\hat{\mathbf{R}}_L$ in the case of $\hat{\mathbf{R}}$ and $\mathbf{Y}_L$ in the case of $\mathbf{Y}$, from which we can estimate the diagonal covariance matrix, for the rows, as:

$$\hat{\mathbf{C}_w} = \frac{1}{(L-1)}\hat{\mathbf{R}}_L\hat{\mathbf{R}}_L^T = \frac{1}{(L-1)}(\mathbf{Y}_L - \hat{\mathbf{m}}\mathbf{1}_L^T)(\mathbf{Y}_L - \hat{\mathbf{m}}\mathbf{1}_L^T)^T \tag{32}$$

where $\hat{\mathbf{C}}_w$ is the bootstrap estimated noise covariance matrix $\mathbf{W}$. It could also be more accurate if it was done by resampling columns of each row independently, as this kind of resampling takes into account the denpencency between rows, but it could be potentially inefficient if the data is independent, which is the assumption that we have adopted between modalities and also an usual assumption for noise in RF receivers.

3.3.2 Reduced model bootstrap resampling

Taking advantage of the simplicity of the reduced model, the statistics of $\hat{\mathbf{m}}$ can be easily characterized, with the aid of Bootstrap Signal Processing. We propose the same resampling principle, as in the previous method, but estimating the covariance matrix $\hat{\mathbf{C}_u}$ in a simpler way, making use of a simpler matrix notation. First, we calculate a matrix containing L Bootstrap estimates of $\hat{\mathbf{m}}_l$ by columns, denoted as $\mathbf{M}_L$ that has $M \times L$ as dimensions. These *estimates* are estimates of multiple realizations of $\hat{\mathbf{m}}$

$$\hat{\mathbf{C}_u} = \frac{1}{(L-1)}(\mathbf{M}_L - \hat{\mathbf{m}}\mathbf{1}_L^T)(\mathbf{M}_L - \hat{\mathbf{m}}\mathbf{1}_L^T)^T \tag{33}$$

where all the parameters involved in (33), have already been mentioned except for the $\mathbf{1}_\mathrm{L}$ vector, which in this case has $L \times 1$ as dimensions and, as a remark. Note that we will be getting the same results as in the previous resampling due to the fact that the same computations are made in both methods.

There are other variants of bootstrap estimation of $\hat{\mathbf{C}}_w$, $\hat{\mathbf{C}}_u$ or even $\hat{\mathbf{C}}_x$, which is the covariance matrix from the latent variables estimation, but they offer similar results as the previosly mentioned resamplings. The main thing to remark, and possibly its biggest flaw, in this estimation algorithm is that it has to assume a WSS process during the time window we are analyzing.

3.3.3 Estimation of latent variables

Now that we have estimated $\hat{\mathbf{m}}$ and $\hat{\mathbf{C}}$, which can be either $\hat{\mathbf{C}}_u$ or $\hat{\mathbf{C}}_w$, we will follow a parametric way of data fusion that has already been advanced in 3.2, which is a Weighted Least Squares estimation. It is parametric in a sense that we just have to estimate 2 set of parameters to define these estimations. Consider a Weighted Least Squares estimate of $\mathbf{x}$, where $\hat{\mathbf{C}}$ is Bootstrap estimations of the respective Covariance matrixes. The

solution is equivalent for both matrixes due to the fact that the relationship between them is a constant that will cancel out:

$$Min_{\mathbf{x}} \left(\hat{\mathbf{m}} - \mathbf{A}\mathbf{x}\right)^T \hat{\mathbf{C}}^{-1} \left(\hat{\mathbf{m}} - \mathbf{A}\mathbf{x}\right) \tag{34}$$

From eq. (34), it can be deduced that what we are doing is minimizing the distance between $\hat{\mathbf{m}}$ and $\mathbf{A}\mathbf{x}$ weighted by $\hat{\mathbf{C}}^{-1}$, which means it is *down-weighting* those rows, or single modalities, that have high variances or are highly contaminated with noise. The solution to the optimization is:

$$\hat{\mathbf{x}} = \left(\mathbf{A}^T \hat{\mathbf{C}}^{-1} \mathbf{A}\right)^{-1} \mathbf{A}^T \hat{\mathbf{C}}^{-1} \hat{\mathbf{m}} \tag{35}$$

$$\hat{\mathbf{x}} = \frac{1}{N} \left(\mathbf{A}^T \hat{\mathbf{C}}^{-1} \mathbf{A}\right)^{-1} \mathbf{A}^T \hat{\mathbf{C}}^{-1} \mathbf{Y}\mathbf{1} \tag{36}$$

from the estimate of the free parameter $\mathbf{x}$ we can now obtain a refined estimate of the sample mean vector as:

$$\hat{\mathbf{m}}' = \mathbf{A}\hat{\mathbf{x}} = \frac{1}{N} \left[\mathbf{A} \left(\mathbf{A}^T \hat{\mathbf{C}}^{-1} \mathbf{A}\right)^{-1} \mathbf{A}^T \hat{\mathbf{C}}^{-1} \mathbf{Y}\right] \mathbf{1} \tag{37}$$

which can be seen as the sample mean of the data after applying a soft projector to the subspace created by $\mathbf{A}$ column vectors as in [6]. The soft projector takes benefit of the available redundancy of the data and from its bootstrap integrity estimate, in order to reduce the impact of the corrupted data in the detrending process.

Recapitulating to the GNSS PPP context, it is expected that those satellites affected by strong variances or strong biases, the latter being caused by accumulated phase cycles slips, be progressively downweighted in the estimation process. In the case of the added biases, its processing will be explored in the next subsection, but first, we have to advance that there is a need of prior knowledge about the integrity of the received data, to know whether or not is affected by cycle slips, which means that every row has to be processed in a *integrity detector* block, such as the explained in section 2.

3.3.4 Dealing with strong biases

Up to this point, we are able to estimate the covariance matrix $\hat{\mathbf{C}}$ to estimate the latent variables, $\mathbf{x}$, under nominal conditions, which means that we had the certainty that every single source of data is not affected by an unexpected bias. Now, we would like to provide some robustness to this estimation by handing some information about each row's integrity before processing this data the same way we have already been explained, for each time window, and using previously estimated data. By previously estimated data, we mean the latest block in which it has been ensured the integrity in every row, remember we have assumed a WSS over the whole data set. Let's assume now, that our linear model has been modified to:

$$\mathbf{Y}_k = \mathbf{A}\mathbf{x}\mathbf{1}_N^T + \mathbf{W} + \mathbf{b}_k \mathbf{1}_N^T \tag{38}$$

where now $\mathbf{b}_k$ is a vector full of zeroes, except for those rows that are known to have an added bias in the k-th time window. In this case, we can cancel out this bias by using the sample mean estimate from the last block that his integrity has been ensured, this mean will be noted as $\hat{\mathbf{m}}_i$. Our main objective is to turn this bias problem to a variance problem, so we will down-weight those rows where there is an added bias. Then, for both resampling methods in this note, we will compute the residuals as:

$$\hat{\mathbf{R}} = \mathbf{Y} - \hat{\mathbf{m}}_i \mathbf{1}_{\mathrm{N}}^T \rightarrow \hat{\mathbf{R}}_L \tag{39}$$

$$\hat{\mathbf{R}}_L = \mathbf{M}_L - \hat{\mathbf{m}}_i \mathbf{1}_{\mathrm{L}}^T \tag{40}$$

note that (39) corresponds to section 3.3.1 and (40) to section 3.3.2 and subindex L denote L Bootstrap resamples by columns. Both residuals matrixes have been expressed in a similar way and have the same dimentions $M \times L$, so there is no ambiguity in the following explanation.

Let's consider that the j-th component of $\mathbf{b}_k$ is different from 0. In this case, the j-th component of $\hat{\mathbf{R}}_L$ is also affected by this added mean that has not been eliminated. If we just detrend this row, we will not be fullfilling our objective, which is to down-weight this row by transforming this bias to variance. Therefore, to heuristically transform this undesired mean to variance the j-th row of $\hat{\mathbf{R}}_L$, denoted as $\mathbf{r}_j$ $(L \times 1)$, we have to apply an elementwise product to $\mathbf{r}_j$ by a vector containing -1 and 1. The mentioned vector has to have an equal number of -1's than the number of 1's, so the resulting mean is $\mathbf{0}_M$, so L has to be even. It has been proposed the following approach to face this issue:

$$\bar{\mathbf{r}}_j = \mathbf{r}_j \odot \cos(\pi \mathbf{n}) \tag{41}$$

where $\odot$ denotes the Hadamard product, or also the elementwise product between matrixes, $\bar{\mathbf{r}}_j$ is the transformed vector, which will substitute the j-th row of $\hat{\mathbf{R}}_L$ and $\mathbf{n}$ is such that it contains L consecutive integers, note that cosine function returns a vector of the same length of $\mathbf{n}$. In the case that the mean is still much different to $\mathbf{0}$, one could still detrend the resulting data.

Having performed these operations, we can just estimate the covariance matrixes the same way that has been explained and the principal objective in this subsection has been achieved, converting bias into variance with additional information of the integrity of each time window. As an additional remark, it will be equal to $\hat{\mathbf{C}}_u$ or $\hat{\mathbf{C}}_w$ depending of how the residuals have been computed:

$$\hat{\mathbf{C}} = \frac{1}{(L-1)} \hat{\mathbf{R}}_L \hat{\mathbf{R}}_L^T \tag{42}$$

3.4 Simulation results

We have already proposed algorithms to achieve near optimum estimates using just informtation we can get from our data. However, there is still the need to show the performance we can get from Bootstrap Estimation on several aspects. As we have already advanced in previous sections, the key estimation in this context is the Covariance matrix estimation, so we will make more emphasis in this step of the estimation.

In order to have proper simulations, a general environment will be set for the simulations. First, let's remember the assumed model in (38), which is the most general scenario.

$$\mathbf{Y}_k = \mathbf{A}\mathbf{x}\mathbf{1}_N^T + \mathbf{W} + \mathbf{b}_k \mathbf{1}_N^T$$

A remark that has to be done in advance, is that we have generated matrix $\mathbf{A}$ with random Gaussian values using a known seed so the resulting matrix can be reproducible and known. Whose parameters have the following values:

Parameter	Value	Description
$\mathbf{A}$	Random	Linear transformation to the Latent variables
$\mathbf{b}_k$	$\mathbf{0}_M$ (default)	Bias vector
$\mathbf{x}$	$\mathbf{1}_D$	Latent variables
N	35 (default)	Number of samples in each time window
M	7 (default)	Number of independent modalities
D	4	Latent variables' dimensions
$\mathbf{C}_w$	$diag(100, 50, 10, 100, 30, 100, 0.1)$	Intermodal covariance matrix
L	$40N$ (default)	Number of Bootstrap Samples

3.4.1 Bootstrap Covariance matrix estimation

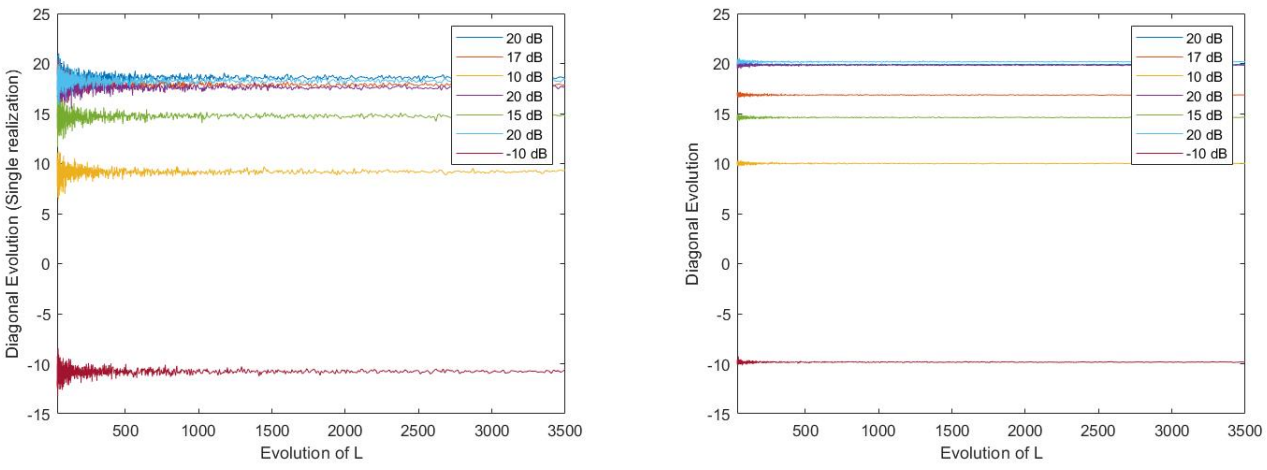
Having set and defined the environment where the simulations will be made, there is a need to identify the parameters that could affect somehow the Covariance matrix estimation. Most of the parameters have already been mentioned in the previous reminder, but there are some that need some special care. Let's start by studying the Bootstrap related parameters, which are: L and N.

As it is said in the literature [3] using Bootstrap techniques, there comes 2 types of errors: The *statistical error*, which depends on the actual sample size N, that consists on the lack of statistical richness of the sample data. We cannot do much about it, as it is dependent of the data we have been received. The other kind of error, is what it is called the *simulation error*, similar to Monte Carlo's simulation errors, which is minimized by increasing the number of Bootstrap samples L. The objective is to reduce the latter up to being smaller than the statistical error. Note that, in order to achieve the mentioned objective, L has to be a function of N. In some literature [10], it is proposed heuristically to choose L to be in between 10 and 100 times N. This is appropriate in many applications, but the optimal value is still dependant of each application, this is why a proper study has to be done.

In all Bootstrap simulations, the mean Bootstrap samples estimations will be prefered, since both resampling methods that have been previously explained are equivalent, in a sense that both methods can estimate either $\mathbf{C}_u$ or $\mathbf{C}_w$ through a linear transformation with an equivalent performance

Impact of L For the study of this parameter, we will show 2 plots related to the estimation of $\mathbf{C}_w$. During the experiment, for one of the plots we have plotted the evolution of the main diagonal component of $\hat{\mathbf{C}}_w$ as a function of L, in dBs for readibility reasons, then, the expected values of the diagonal values in dB are $diag(20, 17, 10, 20, 15, 20, -10)$. The aim of this plot is to show the overall bias, or whether it is unbiased, of the Bootstrap estimation, so it consists on the average of 50 Monte Carlo realizations of the estimation of the diagonal components of $\hat{\mathbf{C}}_w$, remember that we have assumed that this matrix is diagonal, so the estimation assumes that it is diagonal.

Note that this average also reduces the variability of the estimation, which also contains information about the overall performance. For this reason, there is also a second plot, which consists on just the evolution of L for a single realization, so it is possible to appreciate the variability of the estimation. We have swept L logaritmically between N and $100N$. Note that it makes no sense starting from a value less than N, as it is just worsening the estimation. The resultant plots are seen in Figure 7.



(a) Single realization

(b) Multiple averages

Figure 7: Main diag evolution as a function of L

As it can be seen from the plot that displays just a single realization, the variance of each diagonal component's is proportional to the actual estimated value. This dependency between the parameter and its actual value is expected, as it is known that the variability of the variance estimation is directly proportional to the square of the estimated parameter. However, as L increases, it can be observed that the variability is being reduced due to the reduction of the simulation error.

Another remarkable feature is the little bias can be appretiated in the figure, which is also proportional to the value it is estimating. This little bias is related to the statistical error that we have already mentioned. This is the reason why in the plot that displays the average, we can see that this bias tends to be cancelled, because we are artificially providing *richness* by adding more data into the estimation, so the statistical error is reduced. There is still a little bias, but as we increase the number of Monte Carlo averages, this bias is reduced.

Impact of N In this simulation, we are testing how the data richness affects the performance of a Bootstrap estimation. One of the key features of Bootstrap Techniques [3] is the capability of delivering an acceptable performance having a small amount of available data.

We have fixed the value $L = 40N$ for this simulation, since it is a number of Bootstrap samples high enough so the simulation error in the Bootstrap estimation is low enough that we can consider it to be lower than the statistic error in a given realization. As in the previous simulation, the evolution is presented in dB, but it will not use Monte Carlo averages. The evolution of the estimation as a funcion of N can be observed in Figure 8.

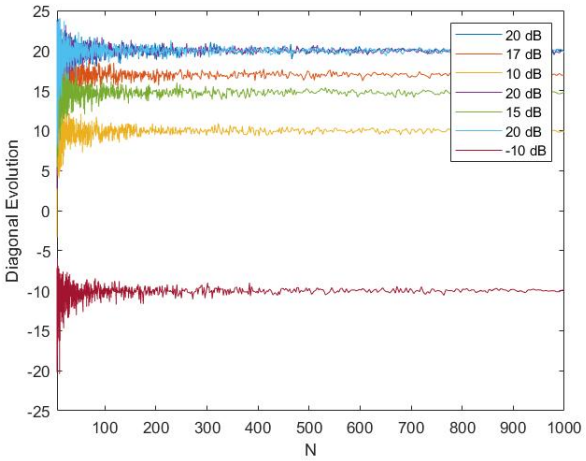


Figure 8: Main diag. evolution as a function of N

This figure shows, again, that this estimation is asymptotically unbiased, and also that the bias which is obtained due to thestatistical error, for a certain N, depends on the realization. As it happened when we varied L, the variance of the covariance matrix diagonal components, with respect to N, also depends on the estimated parameter, as we would expect in an estimation of the variance.

It can be concluded from these simulations that Bootstrap estimation of covariance matrixes is a sort of *booster* to classical approaches of variance estimation up to a certain limit due to lack of statistical richness in the data, but as we give it more data, the estimation improves, as it was already expected.

3.4.2 Latent variables' estimation

Let's now take a look on the true objective in this estimation problem, which is the estimation of the latent variables, $\mathbf{x}$. One of the key issues (estimation of $\mathbf{C}_w$) has already been solved, but there are still 2 important issues that have been developed through the whole note that should be taken into account: The diversity gain as we increase the redundancy, which is connected to the impact of having a larger M, and the issue of having a $\mathbf{b}_k$ different from $\mathbf{0}_M$, which can be related to the cycle slips cancellation problem. Having studied the Boostrap statistical related parameters, let's now focus on the previously mentioned parameters. For the next simulations, the parameters that are not mentioned have their default value.

Impact of M The total number of modalities, M, plays positive role in the estimation of $\mathbf{x}$, which means the larger the value, the better the estimation. However, we have already advanced that if a given modality is contaminated by noise with a high variance, this single source can corrupt the estimation. In order to illustrate this phenomenon, we will compare the estimation of 3 estimators that have been explained through the section 3: The first one will be the Least Squares approach, which needs no special information about the data, the second one is the Weighted Least Squares with knowledge of $\mathbf{C}_w$ and the third one will be the Bootstrap Approach. Also, we will modify the values of $\mathbf{C}_w$ to be initially $diag(500, 50, 10, 100, 30, 100, 10)$, so this phenomenon is more obvious. For all these approaches, we will compute the Mean Square Error by 1000 averages of Monte Carlo simulation. The MSE definition that will be used is (where $\hat{\mathbf{x}}_m$ is the m-th Monte Carlo realization of $\mathbf{x}$):

$$MSE = E\{(\hat{\mathbf{x}} - \mathbf{x})^T(\hat{\mathbf{x}} - \mathbf{x})\} \approx \frac{1}{M_{onteCarlo}} \sum_{m=1}^{M_{onteCarlo}} (\hat{\mathbf{x}}_m - \mathbf{x})^T(\hat{\mathbf{x}}_m - \mathbf{x}) \tag{43}$$

Increasing the total number of modalities, M, means that we have to modify $\mathbf{C}_w$ and $\mathbf{A}$ in each iteration so the model is still consistent. For the matrix $\mathbf{C}_w$, we will just add more diagonal components in an ascending way, so we stay in an heterogeneous noise context. This means the increased $\mathbf{C}_w$ would look like $diag(500, 50, 10, 100, 30, 100, 10, 15, ...40)$, for example. As for matrix $\mathbf{A}$, we will just generate a random matrix with the same random seed for each value of M, as there is no other way to maintain the same matrix for each realization, for obvious reasons. The swept of M are consecutive values between 7 and 20. Note that the minimum value is higher than the latent variables' dimension D, as this is necessary because as a rule of thumb, to have an acceptable estimation, $M - \Delta$ has to be higher than D, where Δ is the number of contaminated channels. The resulting plot, which has been slightly smoothed, can be seen in Figure 9.

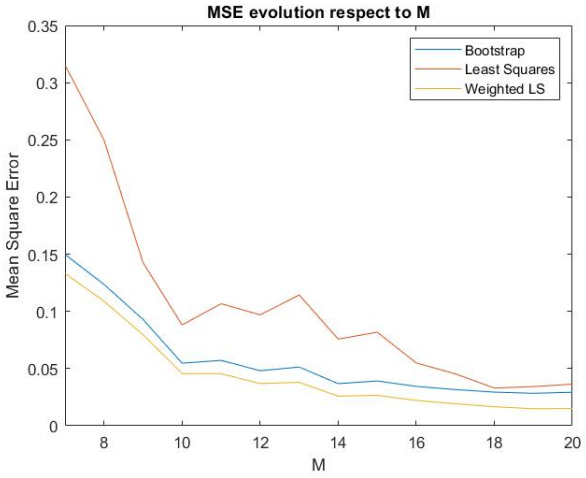
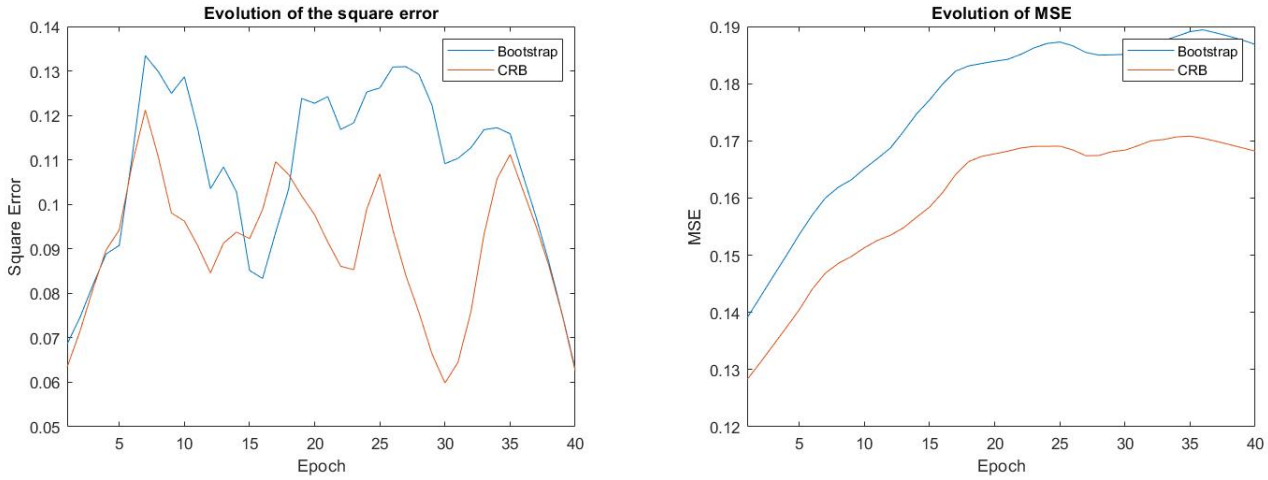


Figure 9: Mean Square Error evolution with respect to M

There are some features to remark from this plot. The first one of them is the obvious outperformance of Weighted Least Squares and Bootstrap approaches over the Least Squares approach, especially when M is low. This is the evidence of data fusion without taking care of contaminated data, where the estimation corrupted by a single source. Then, we can observe that the Bootstrap approach performs slightly worse than the Weighted Least Squares with full knowledge of $\mathbf{C}_w$, this is mainly because the WLS with full knowledge corresponds to the CRLB to this problem, so it is the optimum way of estimating the latent variables. Still, Bootstrap approach is a great approximation of the CRLB, considering the fact that little knowledge about the model and small amount of data was needed.

Time varying variance Now, we would like to observe how well the system adapts to a gradually increasing variance in a specific modality, and compare this evolution to the CRLB estimator (known variances). To perform this test we had to modify $\mathbf{C}_w$, so it could evolve progresively to study the adaptability of the algorithm. The evolution of $\mathbf{C}_w$ is the gradually increasing of a single component, from epoch 1 to 20, so we could express it as $\mathbf{C}_w = diag(500, 50, 10, 100, 30, 10, 0.1 + 5n)$, for $n = 1...20$, so then we maintain this new covariance matrix for an additional 20 epochs, $\mathbf{C}_w = diag(100, 50, 10, 100, 30, 10, 100.1)$. In Figure 10, we will show two plots, one that containt the evolution of a single realization and the other that contains the Mean Square Error as computed in (43), using 1500 Monte Carlo realizations.



(a) Single realization

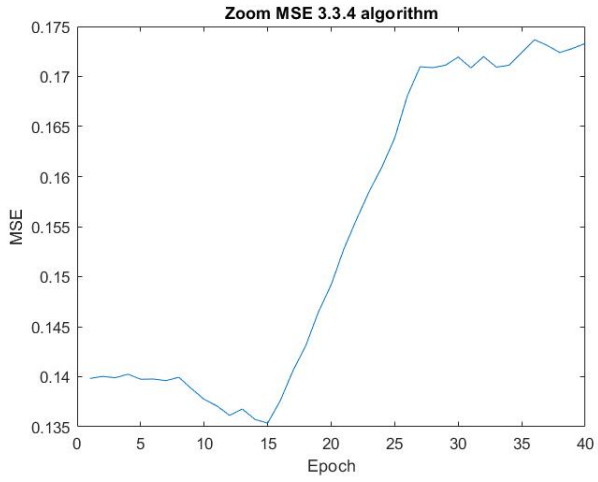
(b) Multiple averages

Figure 10: Evolution of the square error, in a varying context

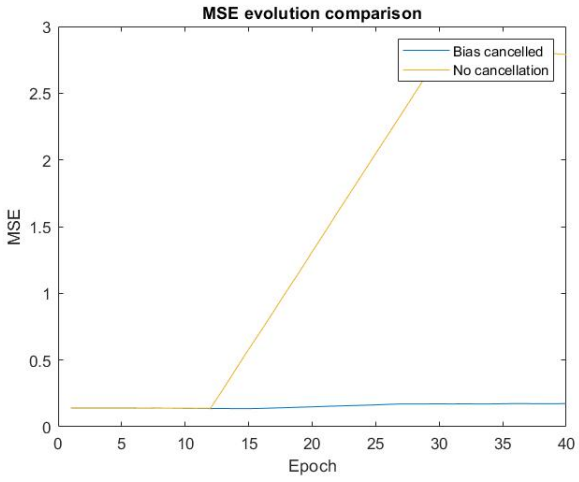
From the multiple averages figure, we can see an obvious loss of quality in the general estimator, due to the loss of redundancy, which is caused by the suddent increment of the noise power of a given modality. This was expected to this experiment, as it ends with just having 4 lightly contaminated sources and 3 highly contaminated sources, but still, these estimators adapt well to this kind of evolutions as there are no clear outliers in this evolution. On the other side, in the single realization plot, we can observe the whole evolution. However, we can just note that it is just a random evolution with no clear tendency at all.

Bias cancellation The main objective of this simulation is to test the proposed algorithm in subsection 3.3.4, therefore, we will compare between the implementation of the proposed algorithm and the implementation with straight Bootstrap Data Fusion, without considering this bias. The simulation will start with 20 Epochs of nominal data fusion, which means $\mathbf{b}_k = \mathbf{0}_M$, so then it continues with 20 more epochs with $\mathbf{b}_k = [0, 0, 0, 0, 0, 0, 10]^T$, so it is affecting the most *uncontaminated* data. It also implements 500 averages of Monte Carlo realizations as in (43). The results can be seen in Figure 11.

We can observe from this figure that in both cases there is an increment of the MSE when in presence of an added bias. However, in the case of no cancellation, an obvious destruction of the whole estimation can be noted, which was expected since biases are the biggest threats in data fusion. On the contrary, when applying the algorithm in 3.3.4, we can see that, althought there is a little increment of the MSE, this algorithm could still be able to mitigate the contamination of the added bias.



(a) Cancelled bias evolution zoomed



(b) General comparison

Figure 11: Evolution of the square error, in presence of an added bias

4 Integration of Hypothesis Testing and Data Fusion

Even thought it was not in the scope of this project, this project reached up to a point where some sort of integration of the previously explained algorithms was needed. The main objective in long term was to provide an added integrity block to an optimal Data Fusion framework, making use of Bootstrap Techniques and approaches that have been explained and explored during this project. Despite the main Signal Processing techiques were focused to be Bootstrap based in this project, as this last section targets a higher level functionality than the previously mentioned algorithms, this last section pretends to consider general methods for Data Fusion and for data integrity detection, and thus this framework can be useful for Signal Processing contexts that are similar to the GNSS PPP satellites.

We have advanced some features of this integration through this document, specially in section 3, where we needed some prior information to cancel out added biases, but now it is time to generalize it. Let's assume now that we have K time windows, which can be overlapped or not, for N samples of multimodal data, which consists on M independent single modalities of N samples. Then, the proposed integrity Data Fusion framework can be seen in Figure 12.

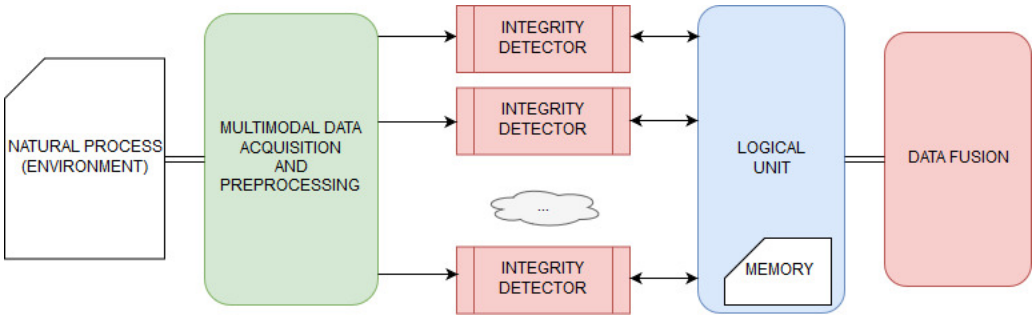


Figure 12: Integrity Data Fusion scheme

Note that all Integrity Detector blocks and the Generalized Data Fusion block can be the ones that we have explained in sections 2 and 3, which is the main idea in this project. In this scheme, the proposed algorithm would implement the Integrity Detector blocks, the Logical unit block and the Data Fusion block. As we have already explained the Data Fusion and Detector blocks, we will focus in this section in the Logical Unit, but first, let's recall some of the key feature that each block needed from this Logical Unit.

In each Integrity Detection block, there is a need of adaptive updating of the training data set if one is interested on monitoring a certain type of disturbances such as the so mentioned, cycle slips. To be able to adaptively perform this *tracing,* this logical unit must store every rejection of H_1 and store the data related to this rejection and this modality as the new training data set for the related modality. In such way, we can permit some adaptability of our algorithm to slight changes of the environment statistics, although some regularization must be implemented in those detectors. By regularization, we mean some *preference* towards reducing P_{FA} or increasing P_D, and for example, in the detector proposed in section 2, we can regularize it by adding some regularization value to the threshold T_H, so the P_{FA} could be effectively reduced.

On the other hand, the Logic Unit could also modify the behaviour of the Data Fusion block in a way that we can perform a *hard* integrity Data Fusion or a *soft* Integrity Data Fusion. In a hard integrity Data Fusion, we would like to priorize the integrity of the global estimation over the added information that we can get from having more redundancy, so in order to achieve this kind of preference, the algorithm could just *disconnect* all modalities that did not reject H_1 and perform the Data Fusion as in section 3, excluding subsection 3.4. However, in such way we may lose too much redundancy and thus, provide a poor quality estimation. To avoid this issue we could implement a soft integrity Data Fusion, where approaches as in subsection 3.4 are highly welcomed. In addition, a mixture of both aforementioned methods of integrity Data Fusion can be implemented in order to have a compromise between integrity and added redundancy.

5 Project plan

With the aid of the following Gantt diagram, in Figures 13 and 14, one could understand the path that this project has followed.

WP1	18/01/18	14/02/18
Bootstrap	18/01/18	1/02/18
Basic Principle	2/02/18	7/02/18
Bootstrap Testing	8/02/18	10/02/18
Bootstrap Techniques Documentation	11/02/18	14/02/18
WP2	15/02/18	26/03/18
Data Fusion Alg. Design	15/02/18	28/02/18
Data Fusion Alg. Testing	1/03/18	7/03/18
Scintillation issues	8/03/18	12/03/18
Data Fusion Performance st...	13/03/18	20/03/18
Data Fusion Docu...	21/03/18	26/03/18
WP3	27/03/18	7/05/18
Bootstrap Hypotesis Testing Research	27/03/18	7/04/18
Detection Algorithm Design	8/04/18	20/04/18
Testing the Algorithm	21/04/18	30/04/18
Detection final doc	1/05/18	3/05/18
Critical Review	1/05/18	7/05/18
WP4	8/05/18	21/06/18
Integration and Final Analysis	8/05/18	22/05/18
Final Report	23/05/18	21/06/18

Figure 13: Gantt Description

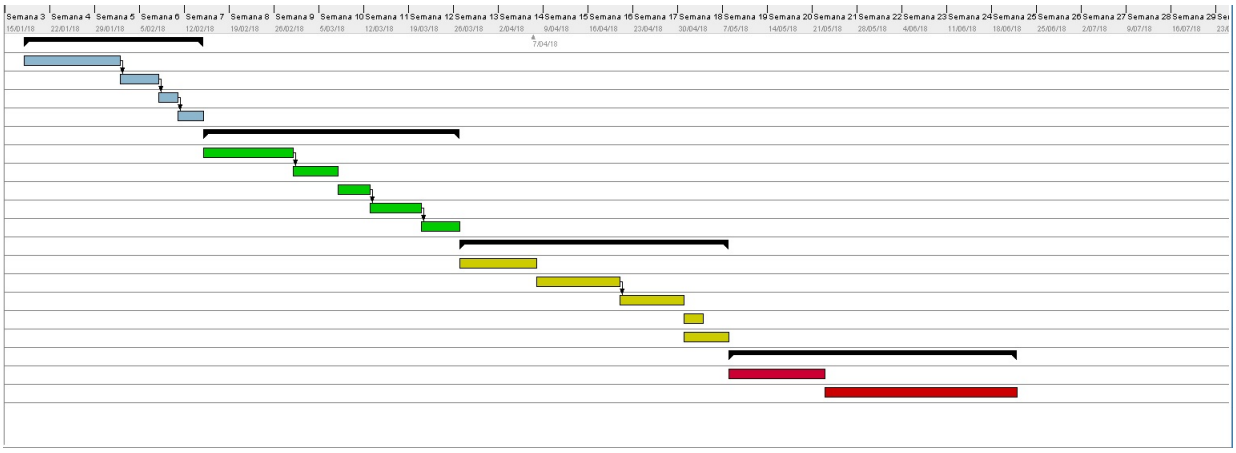


Figure 14: Gantt Diagram

6 Conclusions and future development

The main goal of this project was to study and explore a Bootstrap Signal Processing approach to an issue of delivering integrity to a framework of Data Fusion, inspired by GNSS PPP issue of cycle slips. In particular, we explored how well Bootstrap techniques could enable us to determine whether a received signal was being affected by some kind of disturbance and the generalization of Bootstrap variance estimation into a Covariance matrix estimation, although the mentioned matrix had some particularities. It is extremely useful in a way that it can also be used in Machine Learning classification problems, when facing Data Bases which work better with linear discriminant classifier rather than other Machine Learning approaches, as it has been shown in the annex document that shows some Bootstrap performance in a Pattern Classification competition. After exploring both algorithms, we have proposed a framework were both algorithms could deliver a certain level of integrity to an estimation of a given parameter.

In section 2 we have shown an approach to detect a certain set of disturbances, which can be englobed as the ones that increase the Frequency estimator's variance. The main conclusion for this analysis is that we found out that the estimated threshold may be too much data dependent, but it was a risk that we have assumed when we tried some non-parametric estimation of the mentioned threshold, hence there is still room for more improvements in this kind of approach. Another idea to highlight that appeared when this algorithm was being developed, is the idea of implementing a *Reversed Neyman-Pearson's Detector*, which came out when consulting gAGE research group. They commented that it was prefered to fix a high detection probability, P_D, and minimizing the false alarm probability, P_{FA}, rather than the original version of this Detection criteria. Nevertheless, this was too ambitious for a Bootstrap approach like the one presented in section 2, so this may be a topic for future research.

In section 3 we explored Bootstrap techniques to the estimation of a Covariance matrix, but also explored an approach to deal with added biases using prior information about these biases. It appeared that Bootstrap approach can be used as an acceptable *booster* to traditional variance, or Covariance matrixes, estimation problems, with the great advantage of being able to achieve an acceptable estimation with relatively low number of samples.

To conclude, we achieved the initial objective of *doing the impossible*, as we were capable of delivering information inferring solutions with the possibility of being operational with a small amount of available data.

Impact of this project

This project has been impactful in several aspects, which can be englobed as educational, research and *social media*. The first one of them, the eduactional aspect, this project has been used as an inspiration for a Signal Processing test this term. It showed the concepts of Data Fusion and Integrity, which are two of the main pillars of this project, by considering the estimation of a parameter that is present in two independent modalities, jointly with the basic concepts of estimation's theory such as ML estimators, MVUE, etc... This test can be found in the Annex.

The next aspect is related to research, as this project could potentially have some contributions in a multidisciplinar team with gAGE and some Signal Processing researchers as a result of the project carried out by ESA called SCIONAV, especially contributing with the estimation of the integrity in cycle slips detection. Finally, in the *social media* aspect, it has been observed that Bootstrap techniques are extremely useful in a way that it can also be used in Machine Learning classification problems, when facing Data Bases which work better with linear discriminant classifier rather than other Machine Learning approaches, as it has been shown in the annex document that shows some Bootstrap performance in a Pattern Classification competition.

7 Annex

The annex of this project consists on the following documents:

Proofs_varMVUE Contains a proof that did not fit in this document

G117_LopezMolinaCA_Prac7 Contains the impact of Bootstrap Techniques in a Pattern Classification competition

psavc2018riba Contains the Signal Processing test that showed some of the main concepts of this project

References

[1] J. Sanz, J. M. Juan, and M. Hernández, *GNSS Data Processing.* ESA Communications, (ESA TM-23/1, May 2013) 2013, vol. 1: Fundamentals and Algorithms.

[2] Dana Lahat, Tülay Adali, and Christian Jutten, *Multimodal Data Fusion: An Overview of Methods, Challenges, and Prospects.* Vol. 103, No. 9, September 2015

[3] Zoubir, Abdelhak M. and Iskander, D. Robert (2007) *Bootstrap Methods and Applications : A Tutorial for the Signal Processing Practitioner.* IEEE - Signal Processing Magazine 24(4), pp. 10-19.

[4] Jürgen Franke and Siana Halim, *Wild Bootstrap Tests: Regression models for comparing signals and images.* IEEE - Signal Processing Magazine, July 2007, pp. 31-37

[5] Stefen Kay, *A Fast and Accurate Single Frequency Estimator.* IEEE Transactions on Acoustic, Speech and Signal Processing, Vol 37, No 12. December 1989

[6] Richard T. Behrens and Louis L. Scharf, *Signal Processing Applications of Oblique Projection Operators.* IEEE Transactions on Signal Processing, Vol. 42, No 6, June 1994

[7] Zoubir, Abdelhak M., *Bootstrap Toolbox for Matlab.* URL http://www.spg.tu-darmstadt.de/res/dl/bootstraptoolboxformat

[8] James MacKinnon, *Bootstrap Hypothesis Testing.* Queen's Economics Department Working Paper No. 1127, June 2009

[9] Sergios Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective.* 19 May 2015

[10] A.C. Davison and D.V. Hinkley, *Bootstrap Methods and their Applicat*ions . Cambridge, U.K.: Cambridge Univ. Press, , 1997.

[11] David M. Erceg-Hurn and Vikki M. Mirosevich, *Modern Robust Statistical Methods: An Easy Way to Maximize the Accuracy and Power of Your Research.* October 2008, American Psychologist.

[12] *Effect of Ionospheric Scintillations on GNSS—A White Paper*, SBAS Ionospheric Working Group, November 2010