Data Assimilation under Affine Projection Algorithm

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

Ramon Roca Oliver

In partial fulfillment of the requirements for the degree in
Grau en Enginyeria de Tecnologies i Serveis de Telecomunicació **ENGINEERING**

Advisors: Jaume Riba Sagarra and Carlos Alejandro Lopez Molina
Barcelona, Spring 2022

Abstract

Data assimilation consists of combining multiple sources of information in order to estimate the true state of a physical system as it evolves over time. As this field has multiple applications in today's technology such as automate driving, a lot of research has been done by Bayesian, Shannon's entropy and Deep Learning approaches. This thesis discusses how a classical communications system (GSLC) performs at learning the optimal way to fuse data from a set of sensors using adaptive algorithms (LMS, R-APA) and projective techniques. This study provides a solution and simulation results for two main issues: Selecting the sensors with less output uncertainty and excluding the sensors that attenuate the input signal. To end with, this work also contains its management, a brief description of the economic and environmental impact and the used MatLab code.

Resumen

La asimilación de datos consiste en combinar múltiples fuentes de información para estimar el verdadero estado de un sistema físico a medida que evoluciona en el tiempo. A consecuencia de que este campo tiene múltiples aplicaciones en la tecnología actual, como en la conducción autónoma, se ha investigado mucho y con diferentes enfoques: Bayesianos, utilizando la entropía de Shannon y con modelos de Deep Learning. En esta tesis se discute cómo un sistema de comunicaciones clásico (GSLC) se desempeña al aprender cuál es la forma óptima de fusionar los datos recibidos por un conjunto de sensores empleando algoritmos adaptativos (LMS, R-APA) y técnicas proyectivas. Este estudio proporciona una solución y resultados en forma de simulaciones para dos escenarios: Como seleccionar los sensores que manifiestan una mayor precisión al captar información y como identificar y descartar los sensores que atenúan la señal de entrada. Para finalizar, este trabajo también contiene su plan de gestión, una breve descripción del impacto económico y medioambiental y el código MatLab usado para realizar las simulaciones.

Revision history and approval record

Revision	Date	Purpose
0	04/05/2022	Document creation
1	12/06/2022	Document revision
2	20/06/2022	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Ramon Roca Oliver	ramon.roca@estudiantat.upc.edu
Jaume Riba Sagarra	jaume.riba@upc.edu
Carlos Alejandro Lopez Molina	carlos.alejandro.lopez@upc.edu

Written by:		Reviewed and approved by:	
Date	12/06/2022	Date	16/06/2022
Name	Ramon Roca Oliver	Name	Carlos Alejandro Lopez Molina
Position	Project Author	Position	Project Supervisor

Contents

Abbreviations

- MSE (ε): Mean Squared Error.
- MMSE (ξ) Minimum Mean Squared Error.
- SVD: Singular Value Decomposition.
- FIR: Finite impulse response filter.
- IIR: Infinite impulse response filter.
- LMS: Least Mean Square.
- APA: Affine Projection Algorithm.
- RLS: Recursive Least Squares.
- KF: Kalman Filter.
- GSLC: Generalized SideLobe Canceler.
- BLUE: Best Linear Unbiased Estimator.
- MRC: Maximal Ratio Combiner.
- PCA: Principal Component Analysis.
- SIMO: Single Input Multiple Output.
- CSIR: Receiver Channel State Information.
- FIFO: First In First Out.
- DOA: Direction of Arrival.
- SNR: Signal Noise Ratio.
- AWGN: Additive White Gaussian Noise.
- μ: Step Size.
- $\mathbf{R}$: Covariance Matrix.
- ^{H},^{T}: Hermitian or Conjugate Transpose.
- $E[\ ]$: Average operator.
- $\mathcal{P}$: Projection operator.
- Π: Affine Subspace.
- $||\ \ ||$: Euclidean distance.
- KKT: Karush–Kuhn–Tucker conditions.
- LiDAR: Light Detection and Ranging.
- IoT: Internet of Things.

List of Figures

List of Algorithms

List of Tables

1 Introduction

1.1 Topic

With the imminent arrival of Industry 4.0 and the quest for automation of everyday actions, the need to perceive information from the environment has become an indispensable task. Not only this, but also the estimation of this information must be as accurate as possible in order for the technology to be reliable. One of the many ways to ensure that we are receiving a signal correctly is to compare it with an array of multiple receivers or sensors.

The objective of this thesis is to develop and test different algorithms that find the optimal way to fuse data while assuming that we are receiving a desired signal simultaneously through a wide set of sensors and from a single source.

More specifically, this document emphasizes in how the Affine Projection Algorithm, also known as *APA*, performs at minimizing the least square error inside a interference canceler with spatial reference model known as *GSLC*. Summarizing the contents in this document, following in this section there is a detailed description of the project's management. In the next chapter, an overview of the current paths of investigation among the main thesis topics such as adaptive algorithms and data fusion will be provided. Moving into my contribution, the succeeding sections from 3 to 5 are going to embrace the theory of the investigated models, systems and algorithms along with the fundamental concepts to understand all derivations. A high detail result comparison and its extracted conclusions are going to complete that chapter. To end with, an overall thesis conclusion will close this investigation project.

1.2 Milestones and their Specifications/Requirements

In order to resolve this generic topic, it's mandatory to understand the Stochastic Processes statistics and Signal Filtering theory among complex scenarios. That's why we set the following objectives with their requirements to consider them solved:

- Under perfect circumstances: To understand why and how *GSLC* estimates properly the way to fuse data.
- When *calibration* is assumed: Test how *LMS* performs in *GSLC* and obtain simulation results.
- Still in the *calibration* assumption: Improve our model by using *APA* instead of *LMS* and get exhaustive conclusions from the results and simulations.
- Extend the sensor fusion case to the *miscalibration* scenario and also extract relevant comparisons between the past and new environment.

1.3 Tools and Methodology

To ensure result's accuracy and comprehension, all the methods used have been object to study and investigation. With a solid math background and using nowadays software, the way that this thesis has been carried out begin with the study from the books mentioned in section (6) or the academic papers from: Mendeley's repository, IEEE and ResearchGate. Secondly, as soon as knowing the manner to resolve the aimed objective, set a MatLab experiment that confirms that theory was correctly understood. Thirdly, although some of the illustrations used in this document are from external sources, the most of them will be drawn using Geogebra and Illustrator so they will help to visualize concepts and solutions. Finally, this document has been written in Overleaf LaTeX web editor.

1.4 Work Plan

The main idea of the Work Plan is to understand the steps necessary to overcome the aimed objectives, address the problem, obtain results and move on to more complex scenarios. The thesis is divided into 4 blocks: The first one ensures that we understand the goal and also makes us to prepare the simulation environment.

<table>
<tr><td colspan="2">Project: Data Assimilation under APA</td><td colspan="2">WP ref: (WP1)</td></tr>
<tr><td colspan="2">Major constituent: Define the environment</td><td colspan="2">Sheet 1 of 4</td></tr>
<tr><td colspan="2" rowspan="2">Description:

This initial block of endeavors are not planned to solve any of the milestones but for defining the challenge that we are facing and thinking logical ways to approach it.</td><td colspan="2">Planned start date: 01/02/2022

Planned end date: 04/03/2022</td></tr>
<tr><td colspan="2">Start event: Assignation of the project.
End event: Ending the internal tasks.</td></tr>
<tr><td>Internal task T1: Define Sensor Fusion field, its applications and its similarities with Digital Communication processing.
Internal task T2: Code a Matlab simulation that represents the acquisition of a signal from a bunch of sensors</td><td>Deliverables:

Results PDF</td><td>Dates:

04/03/2022</td></tr>
</table>

Once finished we will have a solid base of the maths involved and also a Matlab environment design. Moving towards, the next step is trying to solve the case where we know a signal signature to be estimated by using simple adaptive methods.

<table>
<tr><td>Project: Data Assimilation under APA</td><td colspan="2">WP ref: (WP2)</td></tr>
<tr><td>Major constituent: Sensor Calibration Resolution</td><td colspan="2">Sheet 2 of 4</td></tr>
<tr><td rowspan="2">Description:

Now we’ll face the sensor fusion ideal case where all the sensors perform as good as possible and receive the same signal with AWGN.</td><td colspan="2">Planned start date: 20/02/2022

Planned end date: 04/03/2022</td></tr>
<tr><td colspan="2">Start event: End of WP1
End event: Ending the WP tasks.</td></tr>
<tr><td>Internal task T1: Code and resolve the case where we have 10 sensors, i.i.d noise with GSLC.
Internal task T2:Try the same model but changing the variance array from the 10 sensors. (Now some of the sensors will add more noise than others, but they will still output the signal equally).
Internal task T3: See how our model fails when miscalibration.</td><td>Deliverables:

Results PDF</td><td>Dates:

04/03/2022</td></tr>
</table>

Hence, after verifying that GSLC converges to the optimal data merger, at WP3 we are going to exchange LMS for APA and verify that we reach the same solution with an improved convergence rate and jitter.

<table>
<tr><td>Project: Data Assimilation under APA</td><td colspan="2">WP ref: (WP3)</td></tr>
<tr><td>Major constituent: APA implementation</td><td colspan="2">Sheet 3 of 4</td></tr>
<tr><td rowspan="2">Description:

Study, comprehend and implement the Affine Projection algorithm to our GSLC model in exchange of the LMS.</td><td colspan="2">Planned start date: 04/03/2022

Planned end date: 15/05/2022</td></tr>
<tr><td colspan="2">Start event: End of WP2
End event: Simulation results.</td></tr>
<tr><td>Internal task T1: Implement APA.
Internal task T2:Simplex scenario modifications.</td><td>Deliverables:
Results PDF</td><td>Dates:
15/05/2022</td></tr>
</table>

Until now we have assumed the knowledge of the desired signal signature. Translated to Communications, this means that the receiver knows signal’s angle of arrival or the channel state information. In the final work package, as long as the supposition is sometimes quite unrealistic, we are going to resolve the paradigm where we don’t have any prior knowledge from the signal that we are receiving.

<table>
<tr><td>Project: Data Assimilation under APA</td><td colspan="2">WP ref: (WP4)</td></tr>
<tr><td>Major constituent: Miscalibration case resolution</td><td colspan="2">Sheet 4 of 4</td></tr>
<tr><td rowspan="2">Description:
The principle of GSLC is to use a matrix to force the input signal signature to be a 0 array. If we are under miscalibration, this matrix cannot be defined with non adaptive values. In this WP, we'll be entering for the first time into research in order to find possible solutions to miscalibration.</td><td colspan="2">Planned start date: 04/03/2022

Planned end date: 15/05/2022</td></tr>
<tr><td colspan="2">Start event: End of WP3
End event: Simulation results.</td></tr>
<tr><td>Internal task T1: Study the traditional solution to GSLC signal signature unpointment.
Internal task T2:Adapt our model in order to solve the miscalibration scenario.
Internal task T3: Put together the results of our first 3 WP.
Internal task T4: (Optional) Compare APA with KF models.</td><td>Deliverables:

Results PDF</td><td>Dates:

14/06/2022</td></tr>
</table>

1.5 Gantt Diagram

The following figure shows the diagram that has been followed throughout these 4 months of work. The first two columns give the task description and the work package to which it belongs. Third one provides the initial prediction of the endeavour duration while the fourth stands for the real time spent. The last column is task's percentage of completion.

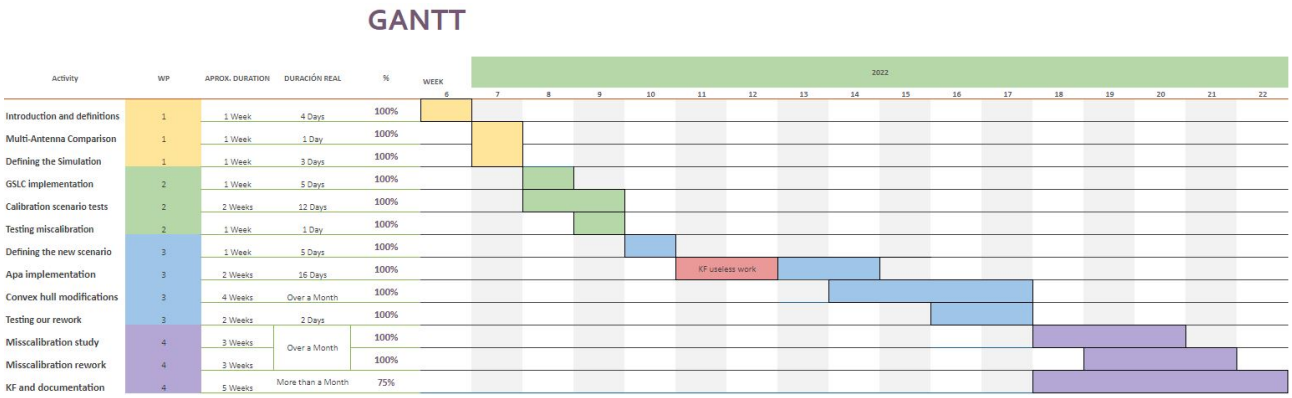

Figure 1: GANTT Diagram.

2 Economic and Environmental Impact

2.1 Budget

Assume that a tech corporation needs to estimate data precisely and decides to invest in sensor fusion investigation. The costs of conducting a research project similar to this thesis would be:

Category	Topic	Description	Anual Inversion
Tools	Hardware	This contains: 1) High-end processor computer: 849€ 2) Peripherals: -Monitor: 119€ -Keyboard and Mouse 18€ -Headphones: 32,90€	1018,9 €
	Internet	1 GB Internet over fiber: 30.95€ / month	371,4 €
	Liscenses	Microsoft 365 Personal: 7€ / month Matlab - 1 year of updates: 150€	234 €
Employee	Salary	36100 € / year gross salary.	46930 €
	Extras	75 € / month	900 €
		Total:	49454,3 €

The outflow will vary slightly from the annual inversion if we apply full costing method and take into account that the project lasts 4 months:

- With the hardware we only have to calculate the depreciation. Warranties least close to a year, so we are going to take a depreciation of the 20 percent: $\frac{(1018,9 \cdot 0.2) \cdot 4}{12\ months} = 67,92$ €
- 1 GB for each month can be shared with at least 3 users. Then, 4 months of this subscription will cost: $\frac{4\ months}{3\ users} \cdot 30,95 = 41,26$ €
- Even though we should pay the entire annual Matlab plan, the cost for 4 months of licenses is: $7 \cdot 4 + \frac{4}{12} \cdot 150 = 78$ €
- The average annual Teleco salary is 36100€, adding the social insurance taxes (30 percent) it turns into 46930€. Hence, the cost for paying an engineer is: $\frac{4}{14} \cdot 46930 = 15643$ €. Finally, as the hours spent are not 8h/day it can be divided by 2 so: $\frac{5 \cdot 15643}{8} = 7821,5$: € working 4h/day.
- The extras stand for worker needing like access to books, academic papers and other needs. $75 \cdot 4 = 300$ €

Hence, deploying this thesis as an investigation would approximately cost:

$$67,92 + 41,26 + 78 + 7821,5 + 300 = 8308,68\ €$$

2.2 Environmental Impact

As we are not manufacturing any products, the only environmental impact inflicted by this thesis is the one produced by the use of computers and Internet.

Taking as reference that the consumption of a Laptop is 0,88 kWh and router's one is 0.05 kWh, the consumption per hour results as: 0.93 kWh. From the budget section, we concluded that the employee should work 4h/day so in a day we spend 0.93 kWh·4 h $= 3.72$ kWh/day and 5 days $\cdot 3.72$ kWh/day $= 18.6$ kWh every week. From the beginning of March to the end of June there are 104 working days, excluding national holidays. From all that, the approximate energy used is: 104 days $\cdot$ 3.72 kWh/day $= 386.88$ kWh.

Using the EPA's eGRID emission factors based on 2020 data published in 2022, we emit 0.371 Kg of CO_2 for every kWh. To end with, printing this thesis will also add residual increases of CO_2e.

Hence, the total footprint is: 386.88 kWh $\cdot$ 0.371 Kg CO_2e/kWh $= 143.53$ Kg CO_2e.

3 The State of the Art of Data Assimilation

As it can be foreseen, this thesis covers different disciplines that have been continuously investigated in the communications paradigm and tries to combine them for solving a data science issue. This section should help to understand the different methods that can be used to advance towards the Data Assimilation casuistic.

Data Assimilation is a technique in which numerical data estimations and observations are combined in order to obtain an analysis that best represents the state of any phenomena of interest. Narrowing this discipline, is it proven in [10] that combining data from a wide set of sensors turns out in obtaining more precise information than the one that the best performing sensor in the set could get. What first drew the attention of the researchers was how the Kalman Filter and its variants behaved at finding the best way to combine data from multiple sources. Although some studies date from 2010 such as [14] which contains models like the Ensemble Kalman Filter, these topics and algorithms are still under investigation. An interactive graph with a set of related studies can be found in [1]. All along the last two decades, I have taken as a reference a study that contains different ways on how to approach Data Assimilation with particle filtering, EnKF, and Markov chain Monte Carlo sampling [17]. At 2008 a study of square root filters performing specifically in sensor fusion was made [11]. In the following years, not only theoretical studies were published but also the Data Assimilation implications in physical models were studied. Some interesting examples are: State estimation for a mobile robot with sensor fusion [6], multi-sensor fusion in automated driving [16], object detection via radar and LiDAR [19], data assimilation in geoscience [5] and IoT using cellphone sensors to interpret human activity recognition [3].

It should also be taken into account that when I particularize to sensor fusion, I assume that adaptive algorithms must be online so our system process the data in real time. If we don't take this assumption there can also be used batch processing methods. In order to visualize the relationship that this thesis has with communications, in [12] it is analyzed how multi-path diversity can improve at the estimation of the received signal parameters.

Focusing on the APA algorithm as it's a fundamental topic in this thesis, a wide study of his derivation and its multiple versions is covered in the book [8]. There are not direct studies on how it performs in the Data Assimilation paradigm. The reason why this study would be interesting is because APA convergence rate is faster than traditional adaptive algorithms and it is easier to implement than the KF family filters.

Beside this Bayesian approach where the main objective is to minimize the output variance, this challenge can also be dealt by trying to maximize the joint information from the inputs or minimize the Shannon entropy [4]. Finally, as it has happened with almost all fields of science and technology, Deep Learning can be applied in this scenario, covering a large part of the studies currently being conducted [7][18]. However, those papers are only implementations for solving specific issues and not theoretical studies on the benefits of using multi-layered neural networks over traditional methods for Data Assimilation.

4 Preliminary Concepts

The purpose of this section is ensuring that the thesis is self contained by offering a brief overview of the signal processing topics tightly related to this investigation, making useful but not necessary appending out to external references.

4.1 Mean Square Error

Mean Squared Error filter theory reefers to a wide bunch of methods that, depending on model's objective, learn how to accomplish that purpose from the statistical data parameters or the data itself. Some applications are: Linear prediction, noise or interference cancellation and system identification.

Being $d[n]$ a reference digital signal that holds a value at each sample, $\mathbf{h}$ a filter of N weights and $\mathbf{x}[n]$ a Snapshot N-array:

$$\mathbf{x}[n] = \begin{bmatrix} x_1[n] \\ x_2[n] \\ \vdots \\ x_N[n] \end{bmatrix}, \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{bmatrix}.$$

The error between the reference signal and the filter output $y[n] = \mathbf{h}^H\mathbf{x}[n]$ where ^{H} refers to hermitian conjugation, is quantified as:

$$\varepsilon[n] = d[n] - \mathbf{h}^H\mathbf{x}[n]. \tag{1}$$

Considering $\mathbf{x}[n]$ is a zero mean stochastic process, the MSE is defined with the following equation:

$$\xi = \mathbf{E}[|\varepsilon[n]|^2] = \mathbf{E}[(d[n] - \mathbf{h}^H\mathbf{x}[n])(d[n] - \mathbf{h}^H\mathbf{x}[n])^*], \tag{2}$$

$$\xi = \mathbf{E}[d[n]d[n]^* - d[n]\mathbf{x}[n]^H\mathbf{h} - \mathbf{h}^H\mathbf{x}[n]d[n]^* + \mathbf{h}^H\mathbf{x}[n]\mathbf{x}[n]^H\mathbf{h}], \tag{3}$$

$$\xi = Pd + \mathbf{h}^H\mathbf{R}_x\mathbf{h} - \mathbf{E}[d[n]\mathbf{x}[n]^H\mathbf{h}] - \mathbf{E}[\mathbf{h}^H\mathbf{x}[n]d[n]^*]. \tag{4}$$

Where Pd is the reference's signal power, $\mathbf{R}_x$ is the input data correlation and the last two terms reefer to the cross correlation arrays between input data values with the reference array and vice versa.

The MMSE criterion in filters is the process of finding the $\mathbf{h}$ values that minimize ξ.

4.2 Wiener Filter

Wiener is a linear filter known for being the optimal solution to MMSE when assuming stationarity. It can be derived from the frequency or the time domain and it can also be formed as a FIR or IIR filter. This thesis only concerns about FIR filters in the time domain. Thus, Wiener filter is going to be defined as the **h** values that minimize (4).

$$\arg\min_{\mathbf{h}}(\xi) \triangleq \nabla_{\mathbf{h}}^{H}(Pd + \mathbf{h}^{H}\mathbf{R}_x\mathbf{h} - \mathbf{E}[d[n]\mathbf{x}[n]^{H}\mathbf{h}] - \mathbf{E}[\mathbf{h}^{H}\mathbf{x}[n]d[n]^{*}]) = 0, \tag{5}$$

$$\mathbf{R}_x\mathbf{h} - \mathbf{E}[\mathbf{x}[n]d[n]^{*}] = 0, \tag{6}$$

$$\mathbf{h}_{opt} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}. \tag{7}$$

$$\mathbf{r}_{dx}[n] = \frac{1}{N}\sum_{k=0}^{N-1} d[n]^{*}x_k[n]. \tag{8}$$

As seen in equation (7), Wiener filter solution requires the knowledge of the data self-correlation and the cross-correlation between the data and the reference signal at every sample n, this is often unrealistic and requires a lot of computation.

4.3 Quick introduction to Sensor Fusion

Filtering theory embrace multiple methods for processing data displayed in different setups. Particularly, Wiener's solution not only demands $\mathbf{R}_x$ and $\mathbf{r}_{dx}$ but also having the entire set or a bunch of blocks of data. When acquiring data via multiple sensors, our system receives a Snapshot vector every *Ts*. If it is assumed that the delays between transmission and reception are not perceptible (*Example: Sensing temperature with near-by sensors.*), then the sensors are receiving at each n data that has been distorted by every sensor's own *Additive White Gaussian Noise*.

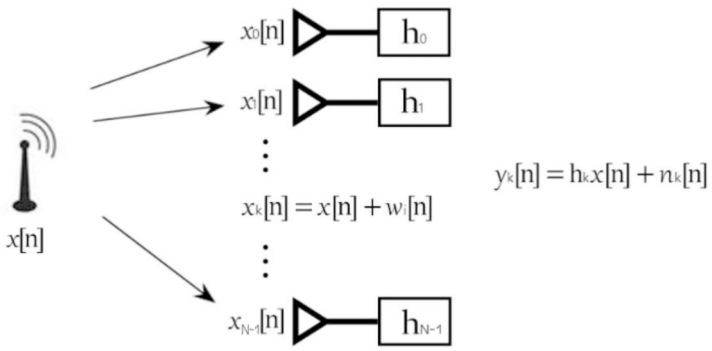

Figure 2: SIMO and Sensor Fusion comparison.

This scenario has a particular similitude with *SIMO* in the *Wireless Communications* field, where $\mathbf{h}$ could mean path attenuation when knowing *CSIR*. In vector notation, the system is receiving after *Ts* or at every n: $\mathbf{y} = \mathbf{h}x[n] + \mathbf{n}[n]$. Moving now to sensor fusion notation and redefining $\mathbf{h} \rightarrow \mathbf{s}$ as the values that hold each sensor's calibration. The received signal is reformulated as $\mathbf{y} = \mathbf{s}x[n] + \mathbf{n}[n]$. The elements of $\mathbf{s}$ span between one and zero, being 1 working perfectly and 0 malfunctioning. Finally, $\mathbf{n}$ describes sensor's noise.

Thus, it is assumed that the sensors are working under processes with real time incoming data, being the objective of the filter to find the weights $\mathbf{h}$ that output the most precise:

$$\hat{x} = \frac{1}{N}\sum_{k=0}^{N-1} h_k y_k \equiv \mathrm{E}[\mathbf{h}^T(\mathbf{s}x[n] + \mathbf{n}[n])]. \tag{9}$$

Notice that data self-correlation: $\mathbf{R}_y = \mathbf{E}[\mathbf{y}[n]\mathbf{y}[n]^H]$ will now depend on sensor's uncertainty or noise covariance matrix.

To sum up, sensor fusion casuistic requires to perform a non identical but similar optimization problem of $\mathbf{h}$ while receiving data continuously and without the prior knowledge of $\mathbf{R}_y$. Understanding that the optimal solution is the Wiener Filter but walking towards a feasible solution, *Least Mean Squared* filters are known as adaptive methods that converge into Wiener's solution by operating Stochastic Gradient Descent without needing to know the data self-correlation.

4.4 Adaptive filters:

As seen in equation (5), the goal is to find the $\mathbf{h}$ that minimize ξ. If the model is sensing real time data, it still aims at reaching this minimum but employing an adaptive method. Most of the solutions, including *LMS*, perform it applying different estimations at the Gradient Descent algorithm.

Extensive theory of Gradient Descent and LMS can be found in the chapter 7.7 of [13].

4.4.1 Gradient Descent

From now on, adaptive filter weights will be defined as $\mathbf{g}$, leaving $\mathbf{h}$ referring to the Wiener Filter. With that in mind, in order to solve the problem the algorithm is expressed as:

$$\mathbf{g}[n+1] = \mathbf{g}[n] + \mu\left[-\frac{\partial\xi}{\partial\mathbf{g}[n]}\right], \tag{10}$$

$$\mathbf{g}[n+1] = \mathbf{g}[n] + \mu[\mathbf{r}_{dx} - \mathbf{R}_x\mathbf{g}[n]] \text{ where } \mu \in \mathbb{R} > 0. \tag{11}$$

The interpretation of equation (11) is: Instead of resolving ξ 's gradient equal to 0, setting iteratively $\mathbf{g}$ so at the next iteration [n+1]: $(\mathbf{r}_{dx} - \mathbf{R}_x\mathbf{g}[n])\Big|_{g[n]}$ nears ξ closer to 0.

This evades calculating $\mathbf{R}_x^{-1}$ while taking profit from the continuous entering flux of data. Contrastingly, it does not elude the need of having to know $\mathbf{R}_x$ and $\mathbf{r}_{dx}$. Self-correlation matrix eigenvalues could let us calculate the optimal step-size μ (*In an opposite way, a non logic value of μ could cause the algorithm to diverge*).

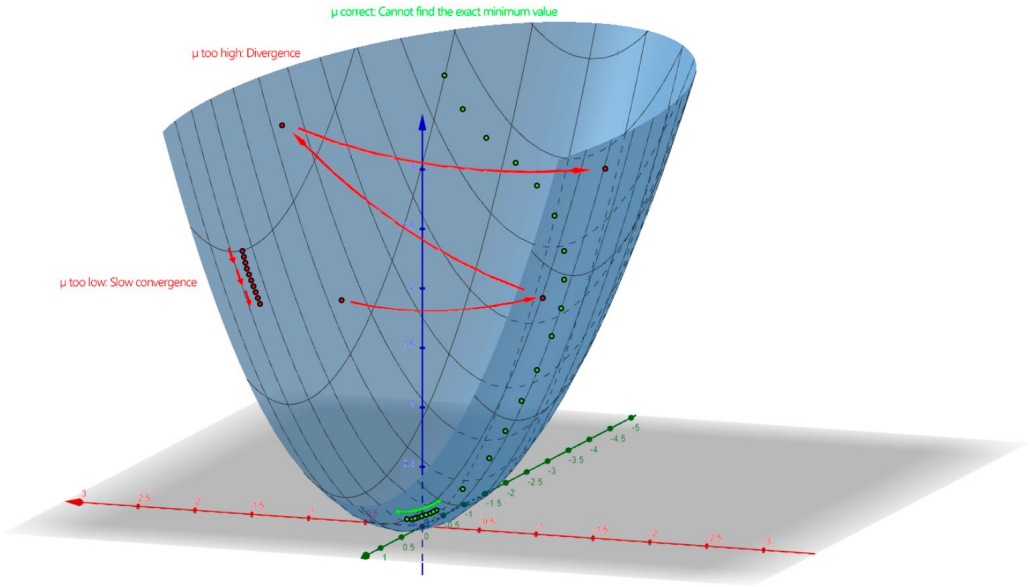

Figure 3: Step size impact in convergence.

4.4.2 Least Mean Squares: LMS

The simplest solution at pursuing the optimal **h** without correlation knowledge and also being capable of adapting to statistic alterations, is by calculating adaptively $\hat{\mathbf{r}}_{dx}$ and $\hat{\mathbf{R}}_x$. Thus, LMS takes as estimators the instantaneous samples and substitute them for the gradient error.

$$\hat{\mathbf{r}}_{dx} = d[n]^*\mathbf{x}[n], \qquad \hat{\mathbf{R}}_x = \mathbf{x}[n]\mathbf{x}[n]^H, \tag{12}$$

$$\mathbf{g}[n+1] = \mathbf{g}[n] + \mu[d[n]^*\mathbf{x}[n] - \mathbf{x}[n]\mathbf{x}[n]^H\mathbf{h}[n]]. \tag{13}$$

The step-size μ is now an small user parameter value. In the stationary case, when μ is too small, LMS converges so slowly into function's minimum or it directly doesn't in non-stationarity. In opposition, higher μ values mislead the algorithm to diverge onto huge weight values by crossing the minimum and going back up through the error function. Comparing the steps from the Gradient Descent and LMS, when μ is accurate they both end up reaching the MMSE, but LMS does not always choose correctly the path with the steepest slope.

4.4.3 Affine Projection Algorithm: APA

Even though LMS with a well chosen step-size would converge into the solution, the convergence rate can be improved by finding an adaptive way to adjust the step size so the algorithm descends faster in the first steps, and it gets more precise with smaller μ values when it nears to the minimum. Although Normalized LMS, which scales μ depending on data and error volume is a valid solution to this issue, in this section is going to be explained its generalization, the Affine Projection Algorithm.

Affine Projection algebraic derivation:

1- Defining $\mathbf{x}$ and $\mathbf{y} \in \mathbb{R}^n$, and being $\mathbb{V}$ a linear subspace of $\mathbb{R}^n$, the orthogonal projection of $\mathbf{x}$ onto $\mathbb{V}$ is $\mathcal{P}_{\mathbb{V}}\mathbf{x} = \mathbf{x}_0 \iff (\mathbf{x} - \mathbf{x}_0) \perp \mathbb{V}$. Thus, if $\mathbf{y} \in \mathbb{V}$, $||\mathcal{P}_{\mathbb{V}}\mathbf{x} - \mathbf{x}|| \leq ||\mathbf{x} - \mathbf{y}||$, defining that the shortest path from a point to a subset is its orthogonal projection.

2- Being $\mathbf{A} \in \mathbb{R}^{m \times n}$, its pseudoinverse matrix is calculated as $\mathbf{A}^+ \in \mathbb{R}^{n \times m} = \mathbf{A}^T(\mathbf{A}^T\mathbf{A})^{-1}$. It verifies $\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$ and $\mathbf{A}^+\mathbf{A}\mathbf{A}^+ = \mathbf{A}^+$. This is also a partial solution to a minimization problem: Any matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ can be considered as a mapping from $\mathbb{R}^n \to \mathbb{R}^m$. Then, if $\mathbf{z} \in \mathbb{R}^m$, the vector $\mathbf{v} \in \mathbb{R}^n$ that solve $\mathbf{X}\mathbf{v} = \mathbf{z}$ so minimizes $||\mathbf{X}\mathbf{v} - \mathbf{z}||^2$ is: $\mathbf{v} = \mathbf{X}^+\mathbf{z}$.

After this brief Algebraic content and before explaining the Affine Projections, it is denominated Affine Subspace Π the subspace where $\forall\, \mathbf{a} \in \Pi$, $\exists\ \Pi - \mathbf{a}$ which is a subspace of $\mathbb{R}^n$ modeled as $\mathbb{V} \triangleq \{\mathbf{x} - \mathbf{a}/x \in \Pi\}$. Finally, for being $\mathbb{V}$ considered as a subspace it has to contain $\mathbf{0}$ and for $\mathbf{u}, \mathbf{v} \in \mathbb{V}$ and any $\lambda \in \mathbb{R}$, $\lambda\mathbf{u} + (1 - \lambda)\mathbf{v} \in \mathbb{V}$.

Putting together this past introductory paragraphs, it could be setted as an environment:

$$\mathbf{x}_0 = \mathcal{P}_{\Pi}\mathbf{x} \iff (\mathbf{x}_0 - \mathbf{x}) \perp \Pi. \tag{14}$$

Where $\mathbf{x}_0 \in \Pi$ is the orthogonal projection of $\mathbf{x} \in \mathbb{R}^n$ into Π. The affine subspace can be modeled as:

$$\Pi \triangleq \mathbf{X}^+\mathbf{z} + \mathcal{N}(\mathbf{X}). \tag{15}$$

Here, $\mathbf{X}^+\mathbf{z}$ is the affine's subset origin denomined '**a**', and $\mathcal{N}(\mathbf{X})$ is the subset Nullspace associated, defined as $\mathcal{N}(\mathbf{X}) \triangleq \{\mathbf{v} \in \mathbb{R}^n / \mathbf{X}\mathbf{v} = 0\}$. Hence, Π is the subspace that contains all the $\mathbf{v}$ vectors that solve $\mathbf{X}\mathbf{v} = \mathbf{z}$. Moreover, the orthogonal projection into this subspace from $\forall\, \mathbf{x} \in \mathbb{R}^n$, is the Euclidean minimum distance from $\mathbf{x}$ to Π.

Note that, if we have Π_1 and Π_2 associated to $\mathbb{V}_1$ and $\mathbb{V}_2$, when $\Pi_1 \cap \Pi_2 \neq \emptyset$, then every value in $\Pi_1 \cap \Pi_2$ has its Euclidean closest distance for any $\mathbf{x} \in \mathbb{V}_1$ or $\mathbb{V}_2$ in $\mathbb{V}_1 \cap \mathbb{V}_2$.

Remembering the formula (13), adaptive algorithm's weight update equation can be simplified as $\mathbf{g}[n+1] = \mathbf{g}[n] + \mu\Delta\mathbf{g}[n]$. In LMS, $\Delta\mathbf{g}[n] = [d[n]^*\mathbf{x}[n] - \mathbf{x}[n]\mathbf{x}[n]^H\mathbf{g}[n]]$. In APA, being P an user parameter denoted as algorithms memory, the update equation is modeled by:

$$\mathbf{g}[n+1] = \mathbf{g}[n] + \mu[\mathcal{P}_{\Pi_n \cap \Pi_{n-1} \cap \ldots \cap \Pi_{n-(P-1)}}\mathbf{g}[n] - \mathbf{g}[n]]. \tag{16}$$

Here Π_n is the affine subspace that contains the solutions of $\mathbf{g}[n]$ that make $\mathbf{g}^T\mathbf{x}[n] = d[n]$, defined by $\Pi_n \triangleq \{\mathbf{g}/d[n] - \mathbf{g}^T\mathbf{x}[n] = 0\}$.

Defining the matrix inside the sensor fusion paradigm described in figure (2): $\mathbf{X}$ is the matrix which its columns are the last P Snapshot arrays of data received by the N sensors.

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ x_k[n] & x_k[n-1] & \ldots & x_k[n-(P-1)] \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{P \times N} \ , \ P < N.$$

Thus, what equation (16) means is that the future step will be modeled as the difference between the actual weights and its projection into the intersection of all the past P Π_p hyperplanes that contained the solutions of $\mathbf{g}^T[n]\mathbf{x}_p[n] = d_p[n]$.

The error can redefined by modifying the equation (1) as $\mathbf{e}[n] = \mathbf{d}[n] - (\mathbf{X}^T\mathbf{g})[n]$. These parameters can be seen as *FIFO* queues that contain P values when applied to $\mathbf{e}[n]$ or $\mathbf{d}[n]$ and P arrays of dimension N when referring to $\mathbf{X}$.
APA calculates at every sample n the already studied minimization problem $\mathbf{X}^T\mathbf{g} = \mathbf{d}$. As it has been seen before, the result of this minimization requires the calculation of the pseudoinverse $\mathbf{g} = (\mathbf{X}^T)^+\mathbf{d}$, and as long as it's contained in the intersection of P Affine Subspaces, the $\mathbf{g}$ projection into this solution is the best estimation as it is the orthogonal projection that minimizes Euclidean distance. The already defined APA's $\Delta\mathbf{g}[n]$ of equation (16) can be rewritten as:

$$\begin{aligned} \mathbf{g}[n+1] &= \mathbf{g}[n] + \mu\mathbf{X}(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{e}, \\ \mathbf{g}[n+1] &= \mathbf{g}[n] + \mu[(\mathbf{X}^T)^+(\mathbf{d}[n] - \mathbf{X}^T\mathbf{g})], \\ \mathbf{g}[n+1] &= \mathbf{g}[n] + \mu[(\mathbf{X}^T)^+\mathbf{d}[n] - \mathbf{g}]. \end{aligned} \tag{17}$$

Note that there is an static step-size user parameter μ which can be used in order to mitigate the noise impact on corrupting the hyperplane positions in space. This noise can make not necessarily optimal to fix the weights update to the hyperplanes intersection. However, the algorithm step is also modulated by the memory matrix $\mathbf{X}$. Also remember that $\mathbf{X}$ depends on n and so it does $(\mathbf{X}^T)^+$.

Affine Projection algorithm can be derived as well with the following minimization problem resolved by Lagrangian Multipliers:

$$\arg\min_{\mathbf{g}_o} ||\mathbf{g}_o - \mathbf{g}||^2 \quad \text{s.t } \forall p \;\; d_p[n] - \mathbf{g}_o^T[n]\mathbf{x}_p[n] = 0. \tag{18}$$

If you are interested in extensive APA theory, I hardly recommend to take a look at the chapter 3.4 of [8]. For a briefer lookout of this last derivation, the thesis from Marc Vilà Insa that you can find in UPC repository is very complete and visual.

The version of APA used in this thesis is the Robust APA. In order to make sure that the Pseudoinverse can be calculated, we add a small delta value:

$$\mathbf{g}[n+1] = \mathbf{g}[n] + \mu\mathbf{X}(\mathbf{X}\mathbf{X}^T + \delta\mathbf{I})^{-1}\mathbf{e}. \tag{19}$$

Finally, we can define NLMS from APA as the case where P = 1 and $0 < \mu < 2$.

4.5 Signal Restoration

Leaving aside algorithms and returning to systems, the received data can be distorted by noise and interferences. Could sensor fusion take profit from the diversity of receiving the same information through different sensors? If it can be solved with a Wiener Filter, it is already proven that adaptive filters can converge to its optimal solution. But what exactly is a reference value d[n] and how can be obtained?

4.5.1 Wiener as a noise canceler

Wiener Filter outputs the data that is most correlated to the reference. This section is going to contain the response of the Wiener Filter when it receives the signal model that has been seen in the section (4.3): $\mathbf{y}[n] = \mathbf{s}x[n] + \mathbf{n}[n]$ being in this case $\mathbf{s} = [1, 1, ..., 1]^T$. Transposed to a real life example, x[n] could be the singer's voice that goes through N sensors in perfect calibration **s** but distorted by the crowd's noise **n**[n]. If a second microphone is placed near to the crowd and only receives a similar **n**[n] shaped noise (highly correlated): $\mathbf{n}_c[n]$, then the Wiener's noise canceler block diagram is:

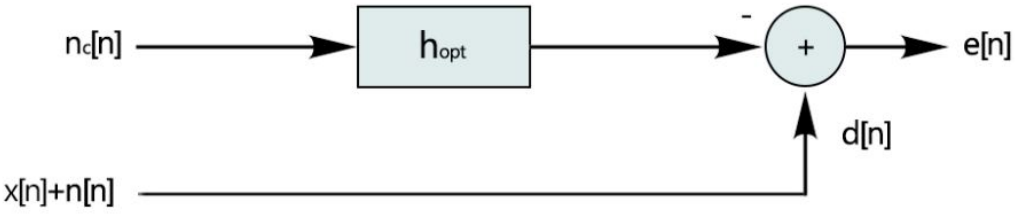

Figure 4: Wiener as a Noise Canceler.

From section (4.2) $\mathbf{h}_{opt} = \mathbf{R}_{n_c}^{-1}\mathbf{r}_{dn_c}$. Being $e[n] = d[n] - \mathbf{h}_{opt}^T\mathbf{n}_c[n]$ and after expanding the error: $e[n] = x[n] + n[n] - \mathbf{h}_{opt}^T\mathbf{n}_c[n] = x[n] + n[n] - \hat{n}[n] = \hat{x}[n] \approx x[n]$. Notice that the model is receiving Snapshot vectors but d[n] must be a scalar and $\mathbf{n}_c[n]$ a vector.

4.5.2 Spatial Reference Interference Canceler: GSLC

Making a hindsight, Wiener filter can clean desired signals if it has as an input a signal that is highly correlated with the noise while the desired signal that we are receiving is setted as the reference. It has been understood that Wiener's solution is the filter weights that minimize the MSE ξ which depend on the input self-correlation and the cross-correlation between the data and the reference. Finally, it has also been shown how to replace the Wiener Filter with adaptive algorithms.

Thus, GSLC stands for Generalized SideLobe Canceler and it is modeled by a system similar to the Wiener noise canceler.

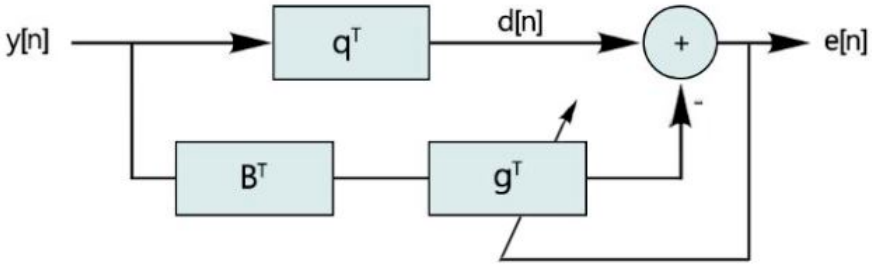

Figure 5: Block diagram of GSLC.

Assuming that all signals are zero mean processes and independent with each other:

- $\mathbf{y}[n]$ is the Snapshot vector that arrives as the n-input of the system. It has the following structure $\mathbf{y}[n] = \mathbf{s}_d x[n] + \mathbf{s}_{i_1} i_1[n] + ... + \mathbf{s}_{i_{N-1}} i_{N-1}[n] + \mathbf{n}[n]$. The non zero vectors $\mathbf{s}_k$ are the data or interference's signature. In communications, they could be the values of incident direction of arrival into an antenna array. It must be fulfilled that $\lambda \mathbf{s}_i + \mu \mathbf{s}_j \neq 0 \ \forall \lambda, \mu \in \mathbb{R}, \quad \forall i \neq j$. Finally, $x[n]$ is the desired signal, $i_k[n]$ are non desired signals and $\mathbf{n}[n]$ is the white Gaussian noise additive to each sensor by the electronics temperature dissipation.
- $\mathbf{q}^T$ is the MRC filter. If we know the desired signal's signature $\mathbf{s}_d$ then $\mathbf{q}^T = \frac{\mathbf{s}_d^T}{||\mathbf{s}_d||^2}$.
- $\mathbf{B}^T$ is a matrix $\in \mathbb{R}^{m \times n}$ that contains $\mathbf{s}_d$ in his nullspace: $\mathbf{s}_d \in \mathcal{N}(\mathbf{B}^T)$ so $\mathbf{B}^T \mathbf{s}_d = 0$.
- $\mathbf{g}^T$ is an adaptive filter that will learn to minimize the MSE of $\mathbf{e}[n]$.

Taking a look inside every block's output, $\mathbf{q}^T \mathbf{y}[n] = \frac{\mathbf{s}_d^T}{||\mathbf{s}_d||^2} \ (\mathbf{s}_d x[n] + \mathbf{s}_{i_1} i_1[n] + ... + \mathbf{s}_{i_{N-1}} i_{N-1}[n] + \mathbf{n}[n])$. This result, as long as it contains the best SNR for x[n], is going to be the reference d[n].

$$d[n] = \mathbf{q}^T \mathbf{y}[n] = x[n] + \frac{\mathbf{s}_d^T}{||\mathbf{s}_d||^2} (\mathbf{s}_{i_1} i_1[n] + ... + \mathbf{s}_{i_{N-1}} i_{N-1}[n] + \mathbf{n}[n]). \tag{20}$$

$\mathbf{B}^T$, is the digital way to separate the signal from the noise: This will create the simile of Wiener filter data input $\mathbf{y}'[n] = \mathbf{B}^T \mathbf{y}[n] = \mathbf{B}^T (\mathbf{s}_{i_1} i_1[n] + ... + \mathbf{s}_{i_{N-1}} i_{N-1}[n] + \mathbf{n}[n])$.

To end with, the error is calculated as seen in the past section: $e[n] = d[n] - \mathbf{g}^T \mathbf{y}'[n] = \hat{x}$.

The equations can be expanded.

$$\begin{aligned} e[n] &= \mathbf{q}^T \mathbf{y} - \mathbf{g}^T \mathbf{B}^T \mathbf{y}[n], \\ e[n] &= x[n] + \frac{\mathbf{s}_d^T}{||\mathbf{s}_d||^2} (\mathbf{s}_{i_1} i_1[n] + ... + \mathbf{s}_{i_{N-1}} i_{N-1}[n] + \mathbf{n}[n]) \\ &\quad - \mathbf{g}^T \mathbf{B}^T (\mathbf{s}_{i_1} i_1[n] + ... + \mathbf{s}_{i_{N-1}} i_{N-1}[n] + \mathbf{n}[n]), \\ e[n] &= x[n] + (\frac{\mathbf{s}_d^T}{||\mathbf{s}_d||^2} - \mathbf{g}^T \mathbf{B}^T)(\mathbf{n}[n] + \sum_{k=1}^{N-1} \mathbf{s}_{i_k} i_k[n]). \end{aligned} \tag{21}$$

Seeing clearly that $\mathbf{g}^T$ weights can adapt itself in order to minimize the interference and the error. Before moving into an introductory example, note that $\dim(\mathbf{y}) = N$ equals the number of sensors. Remembering the definition of signal signatures, it can only handle N-1 interference's and the desired signal so the arrays can be lineally independent among them. $\mathbf{B}^T$ has N columns and N-1 rows. Hence, $\mathbf{g}^T$ contains as weights as the number of $\mathbf{B}^T$ rows. It can be remarked that $\mathbf{g}^T$ weights can be updated with LMS, NLMS, APA, RLS, Kalman Filters as state identifiers... As a final appointment, see that the entire system can be modeled as:

$$\begin{aligned} \mathbf{h}^T &= \frac{\mathbf{s}_d^T}{||\mathbf{s}_d||^2} - \mathbf{g}^T \mathbf{B}^T. \\ \hat{x}[n] &= \mathbf{h}^T \mathbf{y}. \end{aligned} \tag{22}$$

In sensor fusion, it is assumed that there are no interference signals impacting but the variance of every sensor's noise has not any relation with others.

Example:

In order to put to practice all that has been seen, the results of a simplified case of GSLC are going to be exposed:

Assuming the knowledge of $\mathbf{s}_d = \mathbf{1}$ with dimension $N = 3$, $\mathbf{B}^T$ could be:

$$\mathbf{B}^T = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}.$$

$x[n]$ is a zero mean process that we do not know, no interference signals are inciding so it has the following structure $\mathbf{y}[n] = \mathbf{s}_d x[n] + \mathbf{n}[n]$. To end with, $\mathbf{g}^T$ adapts with LMS.

Although this is the best scenario, it can be an illustrative implementation as the solution can be easily contrasted: Assuming that every sensor receives the same $x[n]$ and none of the sensors attenuate the signal, the solution is $\mathbf{g}^T = 0$. Hence, $\mathbf{h}^T = \mathbf{q}^T$ is the MRC filter $\frac{\mathbf{1}}{N}$.

Two simulations were taken: In the left one all the sensors were under the same noise variance (SNR $\approx$ 10.00 dB). Contrastingly, the other one was done by setting the third sensor noise at a higher value (SNR_{s_3} = 1.23 dB). This will be widely studied in the next section, but as a conclusion for the example, it can be observed that when a sensor malfunctions, GSLC discards it as fast as possible in order to learn from the other sensor output and end up converging into the solution. Also, remark that when a sensor performs well, the $\mathbf{g}^T$ weights associated tend to 0 but if not, the adaptive filter learns to cancel this malfunctioning sensor.

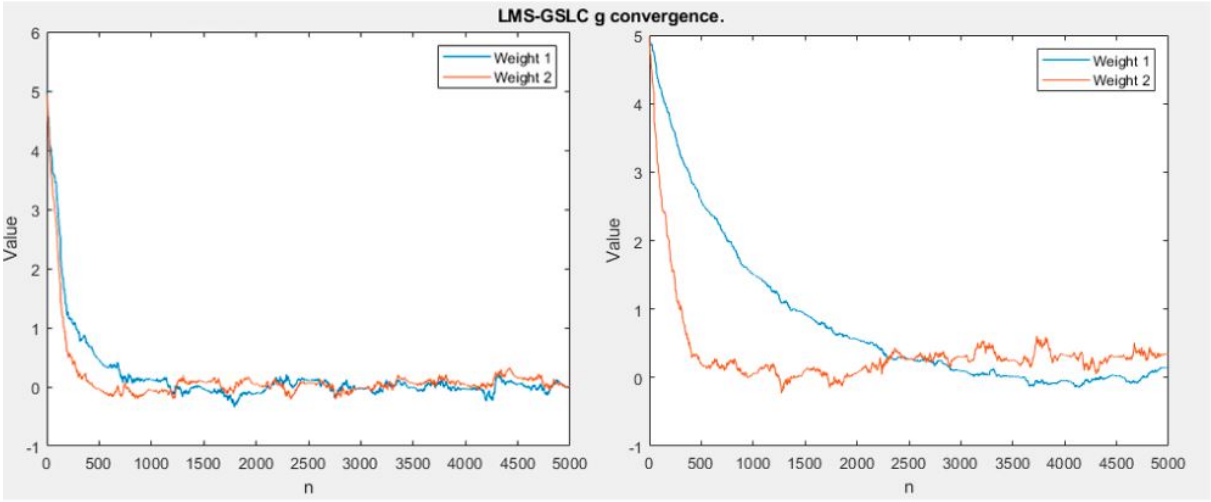

Figure 6: Adaptive filter convergence rate depending on sensors SNR in LMS-GSLC with $\mu = 0.001$.

4.5.3 GSLC into BLUE.

Keeping the model where an N dimensional array is formed by: $\mathbf{y}[n] = \mathbf{s}_d x[n] + \mathbf{n}[n]$, BLUE is defined as the best linear unbiased estimator $\mathbf{h}^T$ so $\mathbf{E}[\mathbf{h}^T\mathbf{y}[n]] = x[n]$. By expanding this equation we can obtain the following constraint:

$$\mathbf{E}[\mathbf{h}^T(\mathbf{s}_d x[n] + \mathbf{n}[n])] = x[n]. \tag{23}$$

And, as long as noise has 0 mean $\mathbf{E}[\mathbf{h}^T\mathbf{s}_d x[n]] = x[n]$ and finally: $\mathbf{h}^T\mathbf{s}_d = 1$. Now calculating the system's output variance σ^2 with the constraint $\mathbf{h}^T\mathbf{s}_d = 1$:

$$\begin{aligned}\sigma^2 = \mathbf{E}[(\mathbf{h}^T(\mathbf{s}_d x[n] + \mathbf{n}[n]))(\mathbf{h}^T(\mathbf{s}_d x[n] + \mathbf{n}[n]))^T] - \mathbf{E}[\mathbf{h}^T(\mathbf{s}_d x[n] + \mathbf{n}[n])]^2,\\ = \mathbf{E}[x[n]x[n] + \mathbf{n}^T[n]\mathbf{h}x[n] + \mathbf{h}^T\mathbf{n}[n] + \mathbf{h}^T\mathbf{n}[n]\mathbf{n}^T[n]\mathbf{h}] - x[n]^2,\\ = x[n]^2 + \mathbf{h}^T\mathbf{C}_n\mathbf{h} - x[n]^2,\\ = \mathbf{h}^T\mathbf{C}_n\mathbf{h}.\end{aligned} \tag{24}$$

Thus, BLUE stands for:

$$\begin{aligned}\arg\min_{\mathbf{h}}\ \mathbf{h}^T\mathbf{C}_n\mathbf{h}\ \ s.t\ \ \mathbf{h}^T\mathbf{s}_d = 1,\\ \nabla_{\mathbf{h}^T}\ \mathbf{h}^T\mathbf{C}_n\mathbf{h} - \gamma(\mathbf{h}^T\mathbf{s}_d - 1) = 0,\\ \mathbf{C}_n\mathbf{h} = \gamma\mathbf{s}_d,\\ \mathbf{h}_{blue} = \mathbf{C}_n^{-1}\gamma\mathbf{s}_d.\end{aligned} \tag{25}$$

The Lagrange coefficient can be easily found by:

$$\begin{aligned}\mathbf{C}_n^{-1}\gamma\mathbf{s}_d^T\mathbf{s}_d = 1,\\ \mathbf{s}_d^T\mathbf{C}_n^{-1}\mathbf{s}_d\gamma = 1,\\ \gamma = \frac{1}{\mathbf{s}_d^T\mathbf{C}_n^{-1}\mathbf{s}_d}.\end{aligned} \tag{26}$$

And substituting the gamma from (26) into the **h** from equation (25), we find that the filter that estimates x[n] with less uncertainty is:

$$\mathbf{h}_{blue} = \frac{\mathbf{C}_n^{-1}\mathbf{s}_d}{\mathbf{s}_d^T\mathbf{C}_n^{-1}\mathbf{s}_d}. \tag{27}$$

Notice that $\mathbf{C}_n$ is the noise correlation matrix that can be modeled as $\mathbf{C}_n = \sigma_n^2\mathbf{I} = diag(\sigma_n^2)$ where σ_n^2 is the array that defines the variance of the noise that every single sensor hold.

Now focusing in the $\mathbf{h}$ found in equation (22) from GSLC, do this solution converge to $\mathbf{h}_{blue}$? It has been seen that the adaptive algorithms converge to the Wiener solution taking steps from partial data and aiming at the Stochastic Gradient Descent direction. Thus, GSLC adaptive filter $\mathbf{g}^T$, when it minimizes MSE it's almost $\mathbf{g}_{opt} = \mathbf{R}_{\mathbf{y}'}^{-1}\mathbf{r}_{\mathbf{y}'\mathbf{d}}$.

If these correlation parameters are expanded,

$$\begin{aligned} \mathbf{R}_{\mathbf{y}'} &= \mathbf{E}[(\mathbf{B}^T\mathbf{y})(\mathbf{y}^T\mathbf{B})] = \mathbf{B}^T\mathbf{C}_n\mathbf{B}, \\ \mathbf{r}_{\mathbf{y}'\mathbf{d}} = \mathbf{E}[(\mathbf{B}^T\mathbf{y})(\mathbf{q}^T\mathbf{y})^*] &= \mathbf{E}[(\mathbf{B}^T\mathbf{n})(x(n) + \frac{\mathbf{s}_d^T}{||\mathbf{s}_d||^2}\mathbf{n})] = \frac{\mathbf{B}^T\mathbf{C}_n\mathbf{s}_d}{||\mathbf{s}_d||^2}. \end{aligned} \tag{28}$$

Replacing them at $\mathbf{g}_{opt}$:

$$\mathbf{g}_{opt} = (\mathbf{B}^T\mathbf{C}_n\mathbf{B})^{-1}\frac{\mathbf{B}^T\mathbf{C}_n\mathbf{s}_d}{||\mathbf{s}_d||^2}. \tag{29}$$

And, from equation (24):

$$\begin{aligned} \mathbf{h} &= \frac{\mathbf{s}_d}{||\mathbf{s}_d||^2} - \mathbf{B}\mathbf{g}, \\ \mathbf{h}_{opt} &= \frac{\mathbf{s}_d}{||\mathbf{s}_d||^2} - \mathbf{B}(\mathbf{B}^T\mathbf{C}_n\mathbf{B})^{-1}\frac{\mathbf{B}^T\mathbf{C}_n\mathbf{s}_d}{||\mathbf{s}_d||^2}, \\ &= (\mathbf{I} - \mathbf{B}(\mathbf{B}^T\mathbf{C}_n\mathbf{B})^{-1}\mathbf{B}^T\mathbf{C}_n)\frac{\mathbf{s}_d}{||\mathbf{s}_d||^2}. \end{aligned} \tag{30}$$

As long as $\mathbf{C}_n^{-1}$ is diagonal, it can easily be proved that: $\mathbf{h}_{opt} = \mathbf{h}_{blue}$ by

$$\mathbf{h}_{opt} = (\mathbf{I} - \mathbf{B}(\mathbf{B}^T\mathbf{C}_n\mathbf{B})^{-1}\mathbf{B}^T\mathbf{C}_n)\frac{\mathbf{C}_n^{-1}\mathbf{s}_d}{\mathbf{s}_d^T\mathbf{C}_n^{-1}\mathbf{s}_d}. \tag{31}$$

So when the numerator is multiplied with the final term in the bracket $\mathbf{B}^T\mathbf{C}_n\mathbf{C}_n^{-1}\mathbf{s}_d = \mathbf{B}^T\mathbf{s}_d$ that contains the constraint $\mathbf{B}^T\mathbf{s}_d = 0$, only the identity affects at the result which equals equation (27).

5 APA-GSLC for Sensor Fusion.

Piercing finally at the research's principal object of study, the main goal is to test a modified version of the affine projection algorithm in the GSLC scheme. This improves the convergence rate into the BLUE combiner. Later on, another modification will be implemented so the algorithm not only converges but also performs without knowing $\mathbf{s}_d$.

5.1 Environment

As seen before, sensor fusion focused in data science does not contemplate interference signals, so the input in our system still maintains the form of $\mathbf{y}[n] = \mathbf{s}_d x[n] + \mathbf{n}[n]$. The signature now refers to the sensor attenuation, being these values spanned from zero to one. Calibration is defined as the case where we know $\mathbf{s}_d$, so $\mathbf{B}^T$ and $\mathbf{q}^T$ can be formulated from it. $x[n]$ is an unknown zero mean process and $\mathbf{n}[n]$ is AWGN statistically independent between sensors which $\mathbf{C}_n = \sigma_n^2\mathbf{I}$ where σ_n^2 is a N dimensional array that contains the variance of the noise added at every single sensor: $\sigma_n^2 = [\sigma_{n_1}^2, \sigma_{n_2}^2, ..., \sigma_{n_N}^2]^T$. GSLC system remains as the one seen in the figure (4).

5.1.1 Convex Set: Simplex

The first modification that will help our system to converge faster is by exploiting that as long as $\mathbf{y}[n]$ covariance is a diagonal matrix, system's $\mathbf{g}^T$ is contained inside a delimited Convex Set.

Proof:

$$\begin{aligned} \mathbf{h} &= \mathbf{q} - \mathbf{B}\mathbf{g}, \\ \mathbf{B}^T\mathbf{B}\mathbf{g} &= \mathbf{B}^T(\mathbf{q} - \mathbf{h}), \\ \mathbf{g} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T(\mathbf{q} - \mathbf{h}) \quad &\text{where} \quad \mathbf{B}^T\mathbf{q} = \mathbf{0}, \\ &= -(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{h}. \end{aligned} \tag{32}$$

So, defining $(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T = \mathbf{B}' \in \mathbb{R}^{m\times n}$ under the data fusion constraints of BLUE $\mathbf{h}^T\mathbf{s}_d = 1$, $h_i \geq 0 \ \forall \ i$, the filter $\mathbf{h^T}$ will pertain into a Convex Set. Hence, $\mathbf{g}^T$ will also be contained at another Convex Set as $\mathbf{B}'$ can be understood as a mapping from $\mathbb{R}^n \rightarrow \mathbb{R}^m$.

Although the weight's projection onto the set will be explained in the next section, a description of the probability simplex and its possible linear transformations will be provided.

Probability Simplex is a set defined as $\{\mathbf{x} \in \mathbf{R}^n / \mathbf{1}^T\mathbf{x} = 1; x_i \geq 0 \ \forall \ i\}$. This equals the BLUE constraint seen at the equation (23), by modifying $\mathbf{s}_d = [0.5, 1, 1]^T$ $\mathbf{h}$ will no longer be inside the probability simplex but it will keep containing its possible values inside a similar $N-1$ dimension Convex Set.

The following figure illustrates the transformation of the Probability Simplex after changing $\mathbf{s}_d$.

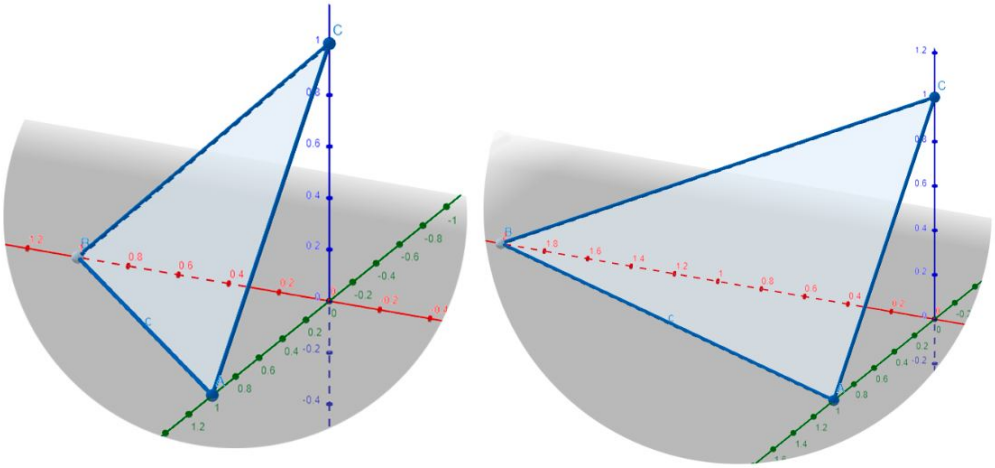

Figure 7: Probability Simplex and Convex Set of possible $\mathbf{h}$ results depending on $\mathbf{s}_d$.

Following with the past example, the result of the mapping $\mathbf{h}$ with $-\mathbf{B}'$ is:

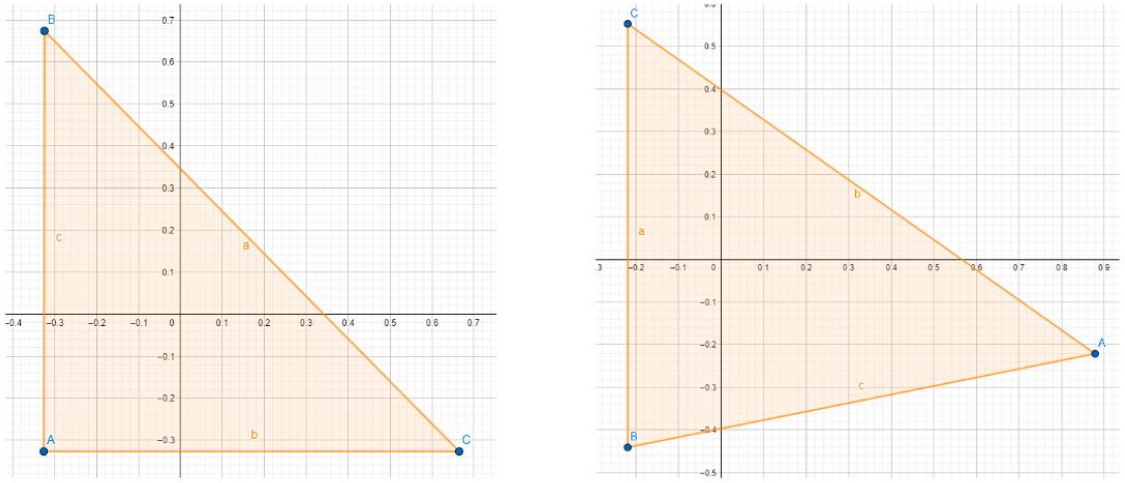

Figure 8: Convex Set of of possible $\mathbf{g}$ weight values depending on $\mathbf{s}_d$.

As a conclusion, notice that all the possible deformations are motivated with variations of $\mathbf{s}_d$ or the $\mathbf{B}^T$ matrix, but $\mathbf{g}$ will be always placed inside a $\mathbb{R}^m$ Convex Set.

5.1.2 R-APA modification: Adaptive weights Projection

In order to take profit from this knowledge, after updating $\mathbf{g}$ from R-APA update equation, $\mathbf{h}$ is going to calculated with the first line of the equation seen at (30), that $\mathbb{R}^n$ point is going to be project into the Convex Set and finally $\mathbf{g}[n+1] \in \mathbb{R}^m$ will be recalculated using the equation (32). The projective algorithm is a modification from [15], a quick overview will be provided.

When the set is convex, there is an unique solution to the Euclidean distance minimization problem. Thus, given an $\mathbf{h}$:

$$\begin{aligned} &\arg\min_{\mathbf{h}_o} \frac{1}{2}||\mathbf{h}_o - \mathbf{h}||^2 \;\; s.t \;\; \mathbf{h}_o^T \mathbf{s}_d = 1, \;\; \mathbf{h}_o \geq 0. \\ &\nabla_{\mathbf{h}_o} \;\; \frac{1}{2}||\mathbf{h}_o - \mathbf{h}||^2 - \gamma(\mathbf{s}_d^T \mathbf{h}_o - 1) - \vartheta^T \mathbf{h}_o = 0. \end{aligned} \tag{33}$$

Notice that in [15] only conceives $\mathbf{s}_d = [1, 1, ..., 1]^T$. Additionally, the $\frac{1}{2}$ scalar does not affect the minimization process and $\vartheta = [\vartheta_1, \vartheta_2, ..., \vartheta_n]^T$ is the list of Lagrangian Coefficients so all the inequality and equalty constraints are met.

Applying KKT conditions at the solution, $\forall i$

- $h_{o_i} - h_i - s_{d_i}\gamma - \vartheta_i = 0.$
- $h_{o_i} \geq 0.$
- $\vartheta_i \geq 0.$
- $h_{o_i}\vartheta_i = 0.$
- $\sum h_{o_i} s_{d_i} = 1.$

By mixing the inequalities, when $h_{o_i} > 0$, then $\vartheta_i = 0$ and $h_{o_i} = h_i + s_{d_i}\gamma$. On the other hand, when $h_{o_i} = 0$, then $h_i + s_{d_i}\gamma = -\vartheta_i$. From these equations it can concluded that h_{o_i} is more positive when h_i is also more positive and the zero values from h_{o_i} are object of h_i smaller weights. Hence, only non 0 values of h_{o_i} affect on the cumulative sum constraint. Taking only the 'n' non zero values from h_o, the last condition can be rewritten as $\sum_{i=1}^{n} s_{d_i}(h_i + s_{d_i}\gamma) = 1$. Then $\gamma = \frac{1}{\sum_{i=1}^{n} {s_{d_i}}^2}(1 - \sum_{i=1}^{n} s_{d_i} h_i)$. (*Notice that s_d is no longer the entire signature array but the values that are bounded to positive h_{o_i} weights*). Once γ is found, the projection is found by calculating $\mathbf{h}_o = \mathbf{h} + \gamma\mathbf{1}$ and setting to 0 all $h_{o_i} < 0$. In order to know the number of positive elements n, we do a loop so 'n' equals the number of 'i' iterations that makes $h_i + \frac{1}{\sum_{i=1}^{n} {s_{d_i}}^2}(1 - \sum_{i=1}^{n} s_{d_i} h_i)$ become a positive scalar. Thus, the algorithm pseudocode is:

Algorithm 1 Projection onto a Convex Set.

1: **Elements**: Inputs: $\mathbf{h}$, $\mathbf{s}_d$,Output: $\mathbf{h}_o$.
2: Sort from max to min: $\mathbf{h}$ along with its corresponding $\mathbf{s}_d$ values.
3: Find the maximum 'i' value that makes: $h_i + \frac{1}{\sum_{i=1}^{n} {s_{d_i}}^2}(1 - \sum_{i=1}^{n} s_{d_i} h_i) > 0$.
4: Calculate $\gamma = \frac{1}{\sum_{i=1}^{n} {s_{d_i}}^2}(1 - \sum_{i=1}^{n} s_{d_i} h_i)$ with n = max 'i' found in the previous step.
5: **Return**: $\mathbf{h}_o$ which its elements equal $max\{h_i + s_{d_i}\gamma, 0\} \;\; \forall\, i$ using the unsorted $\mathbf{h}$.

To end with, the algorithm step-size μ is also modified so it becomes a portion of the distance between $\mathbf{g}$ and the center of the possible $\mathbf{g}$ results set. This middle point can be calculated at the initialization $\mathbf{g}_c = \frac{\mathbf{1}^T}{N}(-\mathbf{B}'\mathbf{I})$. And the step-size can be redefined as $\mu = \alpha|\mathbf{g}_c - \mathbf{g}|$.
Observe that this can be computationally afforded in calibration case, where $\mathbf{s}_d$ is known and as consequence, $\mathbf{B}$ is also known. When miscalibration is produced, every time that a new $\hat{\mathbf{s}}_d$ is aimed, it changes $\mathbf{B}$ and $\mathbf{B}'$ so the set of possible results becomes harder to compute. By making a variable step-size it can be achieved more precise results in our particular scenario as the solution rarely nears in the edges of the Convex Set. Thus, making smaller steps when approaching the center can avoid the algorithm to hop around the points close to the MMSE.

5.2 Data Signature calibration

The signal model in sensor fusion has been precisely defined in section (5.1). Furthermore, the combiner $\mathbf{h}^T$ used for fusing the data is the GSLC schema widely studied in the section (4.5.2). The adaptive algorithm displayed in order to make GSLC converge to BLUE will be firstly the robust APA explained in section (4.4.3). Next, a version where the adaptive weights are projected onto the set of possible results will be given.

All the simulations has been obtained with an experiment with a ten sensor array, so $\mathbf{y}[n], \mathbf{q}^T \in \mathbb{R}^{10}; \mathbf{g}^T \in \mathbb{R}^9$ and $\mathbf{B}^T \in \mathbb{R}^{9\times 10}$. The desired signal x[n] is a sinusoidal waveform. The result of applying R-APA to GSLC in this scenario is.

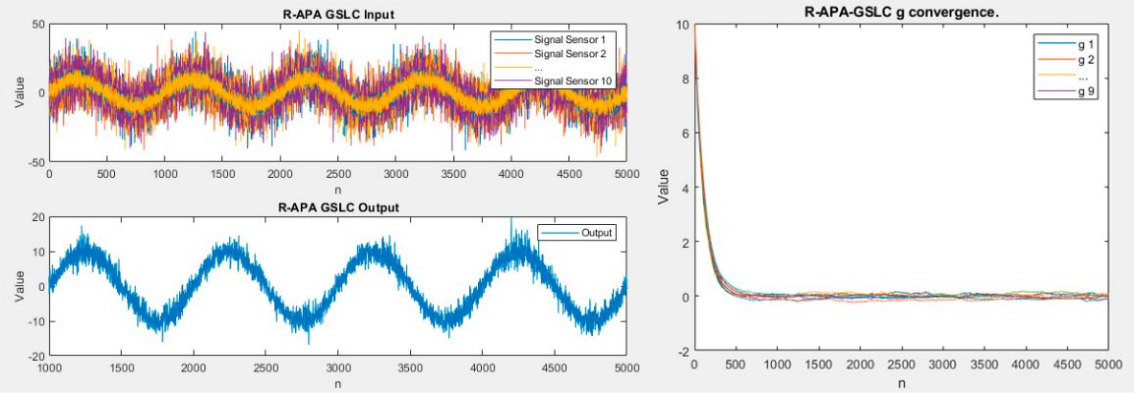

Figure 9: System input, output and adaptive weight convergence rate under 4 Corrupted Sensors, μ = 0.01, projection order = 3 and $\mathbf{s}_d = \mathbf{1}$ with R-APA.

Conclusions will be given later, but from this experiment it is deduced that very high projection orders and step-sizes do not benefit convergence stability.

As it can be seen at figure (16), by changing the algorithm to R-APA it is achieved a convergence rate improvement over LMS. Beyond that, the following algorithm shows the changes after applying the Convex Set projection modification:

Algorithm 2 Projected APA-GSLC in Calibration.

Elements: Input: $\mathbf{y}[n]$, Output: e[n].
Initialize: $\mathbf{s}_d$, $\mathbf{g}^T = \mathbf{10}$, $\mathbf{q}^T = s_d^T/(\mathbf{s}_d^T\mathbf{s}_d)$, Nullspace($\mathbf{B}^T$) $\leftarrow \mathbf{s}_d$, Null memory matrix $\mathbf{X}$ and reference array $\mathbf{d}$, Projection Order $P \in [1, Sensors - 1]$ and small step-size μ.
for $n \in \{samples\}$ **do**
    Update Memory Matrix and reference array as FIFO queues.
    $\mathbf{X}(:,1) = \mathbf{B}^T \cdot \mathbf{y}[n]$
    $\mathbf{d}(1) = \mathbf{q}^T \cdot \mathbf{y}[n]$
    $\mathbf{e} = \mathbf{d} - \mathbf{X}^T\mathbf{g}$
    Update $\mathbf{g}^T$ with the equation: (19)
    Calculate $\mathbf{h}^T$ with the equation: (24)
    Find the projected $\mathbf{g}^T$ as in section: (5.1.2)
Return at every [n]: $\mathbf{e}(1)$.

Applying this algorithm at the same simulation experiment seen at figure (9)

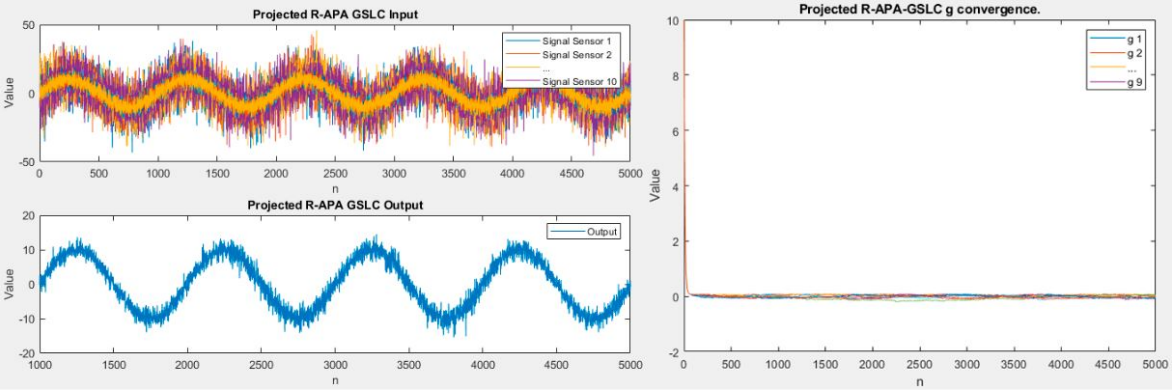

Figure 10: System input, output and adaptive weight convergence rate under 4 Corrupted Sensors, μ = 0.01, projection order = 3 and $\mathbf{s}_d = \mathbf{1}$ with Projected R-APA.

Thus, a huge improvement in the convergence of the algorithm can be seen by delimiting a space of possible $\mathbf{g}$ answers.

Making a final remark of the calibration case, this last simulation represents the impact of changing $\mathbf{s}_d$ to a known but not $\mathbf{1}$ signature and the impact of not knowing this array (miscalibration without changing anything). As can be seen, the Projected APA in GSLC has no problem adapting to other signatures if it knows them, but it becomes completely useless if it doesn't know $\mathbf{s}_d$.

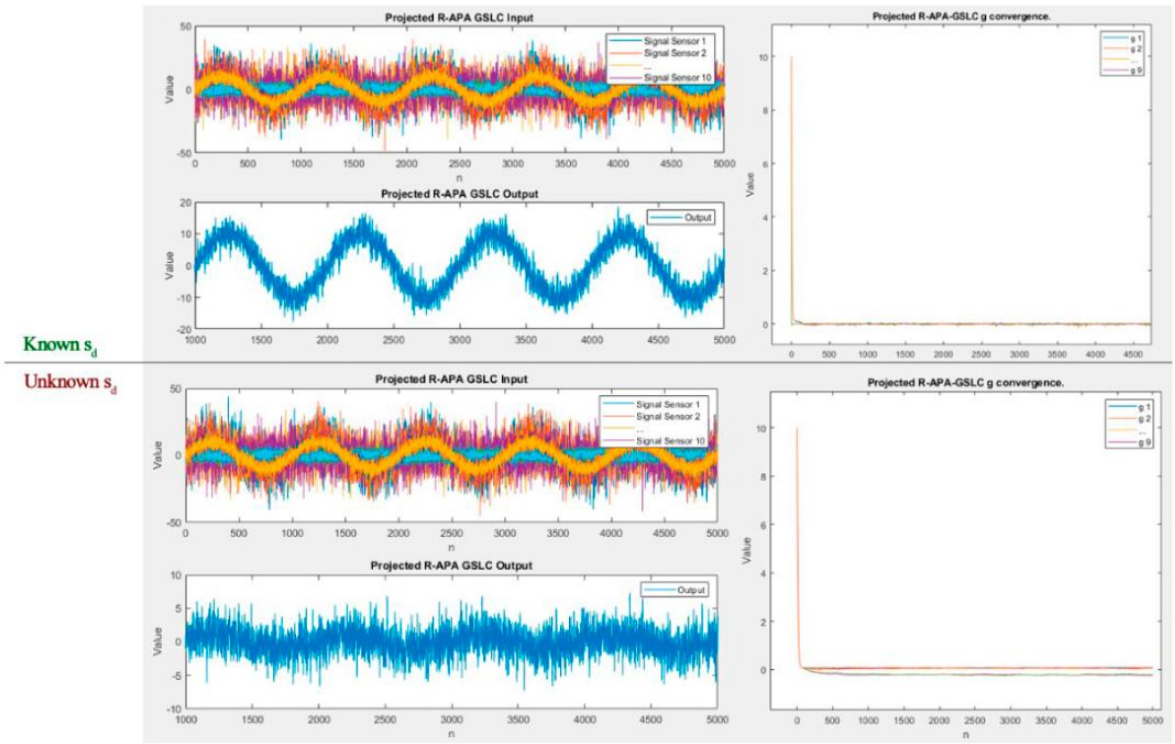

Figure 11: System input, output and adaptive weight convergence rate under or without the knowledge of $\mathbf{s}_d$. Projected R-APA with 4 Corrupted Sensors and μ = 0.01.

5.3 Data Signature miscalibration

Now assuming $\mathbf{s}_d$ is an unknown data signature, the MRC filter that conforms the reference to our adaptive filter cannot be calculated anymore. Furthermore, the matrix $\mathbf{B}^T$ that removes the signal associated to $\mathbf{s}_d$ is also undetermined. Hence, the pointing error is defined as the difference from $\mathbf{s}_d$ and $\hat{\mathbf{s}}_d$. Notice that the system is receiving unknown data differently affected by the sensors attenuation and noise. These two degradation factors can happen simultaneously or not, distorting the desired signal. Still, with some modifications in the algorithm (2), $\hat{\mathbf{s}}_d$ is going to be precise and our system is going to learn which are the worse performing sensors. Recovering the matrix notation, the incoming data can be modeled as:

$$\mathbf{Y} = \begin{bmatrix} | & & | & | \\ y_k[n+M] & \dots & y_k[n+1] & y_k[n] \\ | & & | & | \end{bmatrix} \in \mathbb{R}^{N \times M}.$$

M refers to the Snapshot total arrivals and N the number of sensors. Lets take a simple example with $N = 2$ sensors where one of them is perfectly calibrated and the other one attenuates the signal $\mathbf{s}_d = [1, 0.4]^T$, being this $\mathbf{s}_d$ an unknown array for our system. The desired signal is a zero mean process and the noise has equal variance among all sensors. With M samples, plotting every $\mathbf{y}$ snapshot at $\mathbb{R}^N$ the distribution of every $\mathbf{y}[m]$ with $0 \leq m \leq M-1$ and knowing that the covariance has the form: $\mathbf{s}_d\mathbf{s}_d^T\sigma_x^2 + diag(\sigma_n^2)$ would be as shown in the following figure.

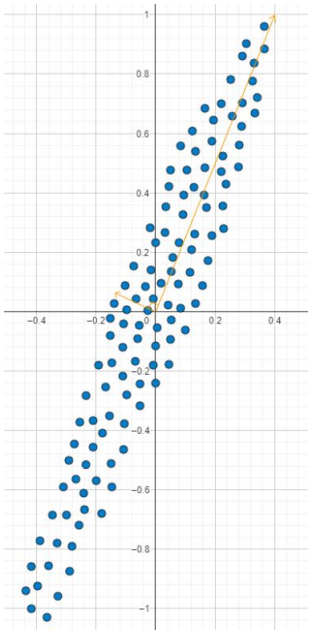

Figure 12: PCA visualization, the two first principal components in $\mathbb{R}^2$ with $s_d = [1, 0.4]^T$.

Even though the data signature is unknown, the way in which the direction of $\hat{\mathbf{s}}_d$ can be calculated from the incoming data structure is by finding its principal component in an adaptive way. After that, $\mathbf{q}^T$ and $\mathbf{B}^T$ can be obtained from $\hat{\mathbf{s}}_d$. A brief summary of how to obtain the first principal component from data will be provided before moving into recursive PCA.

5.3.1 Principal Component Analysis: PCA

The objective is to find the first principal component as it contains the normal direction that has the most variance, which is going to be an attenuated digital signature: $\hat{\mathbf{s}}_d$.

To start with, finding the first principal component direction is formulated as:

$$\hat{\mathbf{s}}_d = \arg\max_{\mathbf{s}} \mathbf{s}^T \mathbf{R}_y \mathbf{s} \;\text{ s.t. }\; \mathbf{s}^T \mathbf{s} = 1. \tag{34}$$

This maximization can be resolved with Lagrangian coefficients.

$$\begin{aligned} \nabla_{\mathbf{s}^T} \;\; \mathbf{s}^T \mathbf{R}_y \mathbf{s} - \gamma(\mathbf{s}^T \mathbf{s} - 1) = 0, \\ \mathbf{R}_y \mathbf{s} = \gamma \mathbf{s}. \end{aligned} \tag{35}$$

Resulting into an equation which can be resolved associating γ as the maximum eigenvalue of $\mathbf{R}_y$ and finding $\mathbf{s}$ as the eigenvector corresponding to this maximum eigenvalue γ.

Beyond this equations, this model requires the assumption of having a centered data and also being able to calculate previously its covariance.

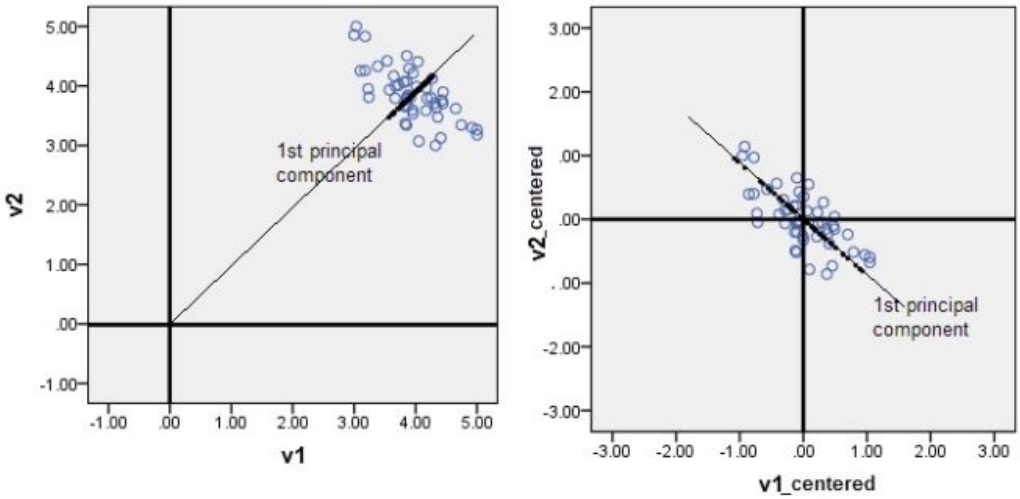

Figure 13: Non zero mean data impact for PCA [2].

As seen in the last figure, not having the data centered may cause PCA to choose incorrectly the first principal component direction. As long as sensors can only obtain real time samples $\mathbf{y}[n]$ and not the entire $\mathbf{Y}$, it is going to be used the adaptive version of PCA studied in [9].

5.3.2 Recursive PCA

Assuming that $\mathbf{R}_n$ cannot be calculated in order to obtain the BLUE combiner, the system will aim at calculating $\hat{\mathbf{R}}_y$ and use it to find the MRC filter in the GSLC model.

Although in the paper they calculate the $\mathbf{Y}$ covariance matrix from singular value decomposition matrices, we can obtain it recursively from samples $\mathbf{y}[n]$ as:

$$\hat{\mathbf{R}}_y[n] = (\frac{n-1}{n})\mathbf{R}_y[n-1] + \frac{1}{n}\mathbf{y}[n]\mathbf{y}[n]^T. \tag{36}$$

$$\hat{\mathbf{R}}_y[n] = (1-\lambda)\hat{\mathbf{R}}_y[n] + \lambda\mathbf{y}[n]\mathbf{y}[n]^T. \tag{37}$$

The first case holds the optimal estimator without a forgetting factor that can be used in stationary environments. With the second option, is it modeled with a lambda between one and zero that give more or less importance to the past samples, adapting the estimation to non-stationary conditions. From that correlation matrix, the eigenvalues and eigenvectors can be extracted to get the first component from PCA.

Before taking the appropriate modifications, notice that the matrix $\mathbf{R}_y$ could also be expressed with its singular value decomposition (SVD) form: $\mathbf{S}\Lambda\mathbf{S}^T$ where $\mathbf{S}$ is a matrix which its columns are the $\mathbf{R}_y$ eigenvectors and Λ is the diagonal matrix holding its eigenvalues:

$$\begin{aligned}(\mathbf{S}\Lambda\mathbf{S}^T)[n] &= (1-\lambda)(\mathbf{S}\Lambda\mathbf{S}^T)[n-1] + \lambda\mathbf{y}[n]\mathbf{y}[n]^T,\\ \mathbf{S}_n\Lambda_n\mathbf{S}_n^T &= \mathbf{S}_{n-1}\Big((1-\lambda)\Lambda_{n-1} + \lambda\mathbf{S}_{n-1}^T\mathbf{y}_n\mathbf{y}_n^T\mathbf{S}_{n-1}\Big)\mathbf{S}_{n-1}^T.\end{aligned} \tag{38}$$

The big parenthesis also contains a matrix that can be decomposed in singular values: $\mathbf{U}\Gamma\mathbf{U}^T$, simplifying the last equation:

$$\begin{aligned}\mathbf{S}_n\Lambda_n\mathbf{S}_n^T &= \mathbf{S}_{n-1}\mathbf{U}\Gamma\mathbf{U}^T\mathbf{S}_{n-1}^T,\\ \mathbf{S}_n &= \mathbf{S}_{n-1}\mathbf{U},\\ \Lambda_n &= \Gamma.\end{aligned} \tag{39}$$

With this, the update equations have been found without calculating the eigenvalues and eigenvectors from the data correlation matrix. The circumstance that can make this SVD useful, is when λ is really small so the matrix in $\Big((1-\lambda)\Lambda_{n-1}+\lambda\mathbf{S}_{n-1}^T\mathbf{y}_n\mathbf{y}_n^T\mathbf{S}_{n-1}\Big)$ is almost diagonal and it can be calculated efficiently its eigenvalues and eigenvectors by doing a further approximation with perturbation matrices. Although this is a witty way to calculate the principal components, there is no need to calculate them at every sample, but it could be implemented for environments with continuous statistical variations.

5.4 Miscalibration Results

In this experiment the calibration method will be used with an unknown signature while estimating the covariance with the equation (37). A user parameter named Aim defined as any sample in which the input covariance matrix has already converged to the solution is going to determinate the moment when the eigenvector associated to the maximum eigenvalue is going to be calculated. Once the eigenvector is obtained, a linear transformation will be applied on it so its values span between 0 and 1. After that, the $\mathbf{q}^T$ and $\mathbf{B}^T$ will be recalculated. Finally is is also going to be computed the projected Convex Set so it ensures that the calibration method will perform correctly using this updated parameters.

Algorithm 3 Miscalibrated APA-GSLC.

Elements: Output: e[n], Input: $\mathbf{y}[n]$.
Initialize: $\hat{\mathbf{s}}_d = \mathbf{1}$, $\mathbf{g}^T = \mathbf{10}$, $\mathbf{q}^T = \hat{\mathbf{s}}_d^T/(\hat{\mathbf{s}}_d^T\hat{\mathbf{s}}_d)$, $\mathbf{B}^T \rightarrow$ Gram-Schmidt $(\hat{\mathbf{s}}_d)$, Null Memory and Covariance$_y$ Matrices, Projection Order $P \in [1, Sensors - 1]$, Small step-size μ, forgetting factor λ and the Aim sample.
for $n \in \{samples\}$ **do**
    $\hat{\mathbf{R}}_y[n] = (1-\lambda)\hat{\mathbf{R}}_y[n] + \lambda\mathbf{y}[n]\mathbf{y}[n]^T$
    if (n != Aim) **then**
        run as $\rightarrow$ Calibrated APA-GSLC
    else
        [CovEigVect,CovEigVal] = eig($\hat{\mathbf{R}}_y[n]$)
        [val,ind] = max(diag(CovEigVal))
        $\hat{\mathbf{s}}_d$ = CovEigVect(:,ind)
        $\hat{\mathbf{s}}_d$ = $\hat{\mathbf{s}}_d$/max($\hat{\mathbf{s}}_d$)
        $\mathbf{q}^T = \hat{\mathbf{s}}_d^T/(\hat{\mathbf{s}}_d^T\hat{\mathbf{s}}_d)$
        $\mathbf{B}^T$: **run** $\rightarrow$ Nullspace($\hat{\mathbf{s}}_d$)
Return:e(1) at every n.

The Matlab Code used for all the simulations can be found in the Appendix.

Simulating the already seen miscalibration attempt in figure (11) with this improved algorithm:

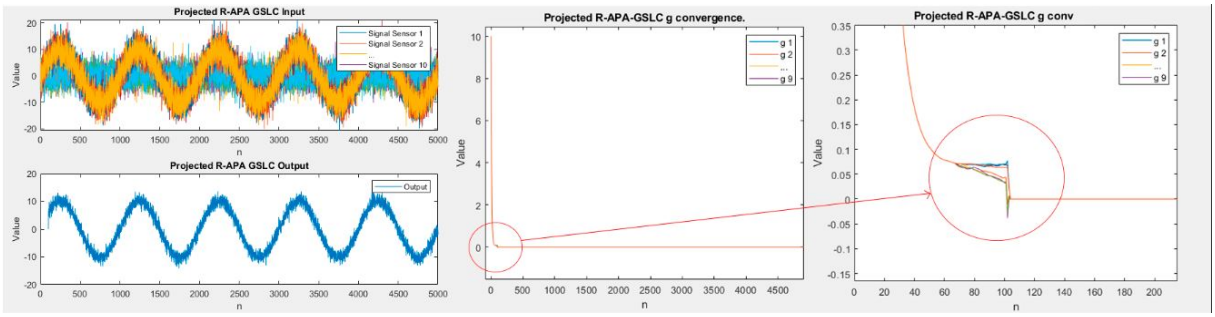

Figure 14: Miscalibration simulation under 4 Corrupted Sensors, μ = 0.01, projection order = 3 , unknown $\mathbf{s}_d$ and $Aim = 100$.

This proves that when the signature is targeted correctly the algorithm converges to the solution of the optimal data merger.

Repeating the simulation when Aim is fixed at the middle of the plot can help to visualize what happens when we properly estimate $\mathbf{s}_d$.

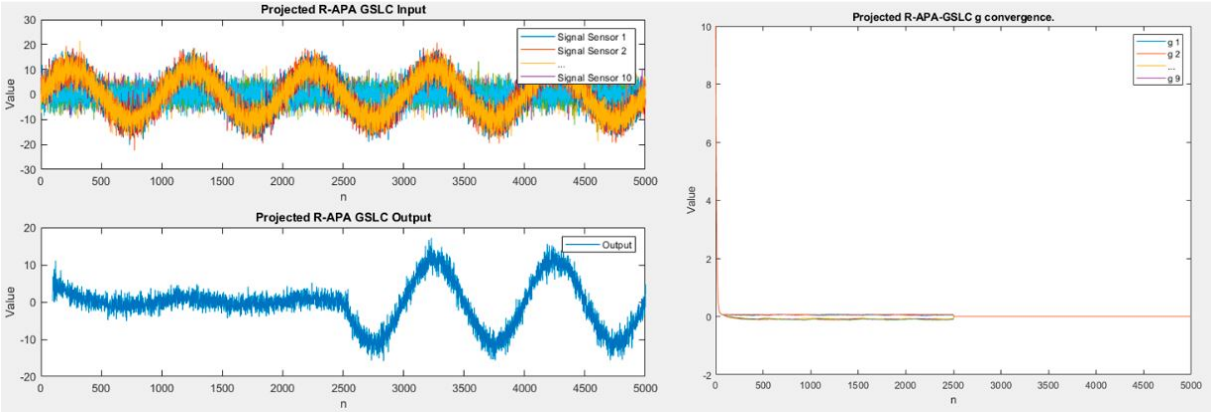

Figure 15: Miscalibration, importance of estimating $\mathbf{s}_d$, μ = 0.01, projection order = 3 , unknown $\mathbf{s}_d$ and $Aim = 2500$.

When $\mathbf{s}_d$ is unknown the matrix $\mathbf{B}^T$ that cancels the desired signal cannot be designed. Thus, Wiener's solution takes as input $\mathbf{y}[n]' = \mathbf{B}^T\mathbf{y}[n]$ which still contains the desired signal and then minimizes $\mathbf{d} - \mathbf{B}^T\mathbf{y}[n]$. As long as the reference also contains the desired signal, the adaptive filter makes the output highly correlated to it, making the desired signals from $\mathbf{y}[n]'$ and $\mathbf{d}$ cancel with each other in order to minimize the error. Once $\mathbf{s}_d$ is aimed, the input $\mathbf{y}[n]'$ no longer contains the desired signal and the R-APA weights only correlate the noise, cleaning the wanted signal as in calibration.

5.5 Result Comparison

A good indicator that the model is working properly is to compare the output's variance with the output's minimum variance that can be obtained by using equation's (27) result in the equation (24). The output's minimum variance is $\sigma^2_{opt} = \frac{1}{\mathbf{s}_d^T \mathbf{C}_n^{-1} \mathbf{s}_d}$.

For the case of calibration, LMS, R-APA and Projected R-APA have been tested. In order to visualize the effect of improving the adaptive model and its impact on convergence and stability, the following figure is the result of plotting the mean value of the converging $\mathbf{g}$ for all three algorithm versions under the same simulation settings.

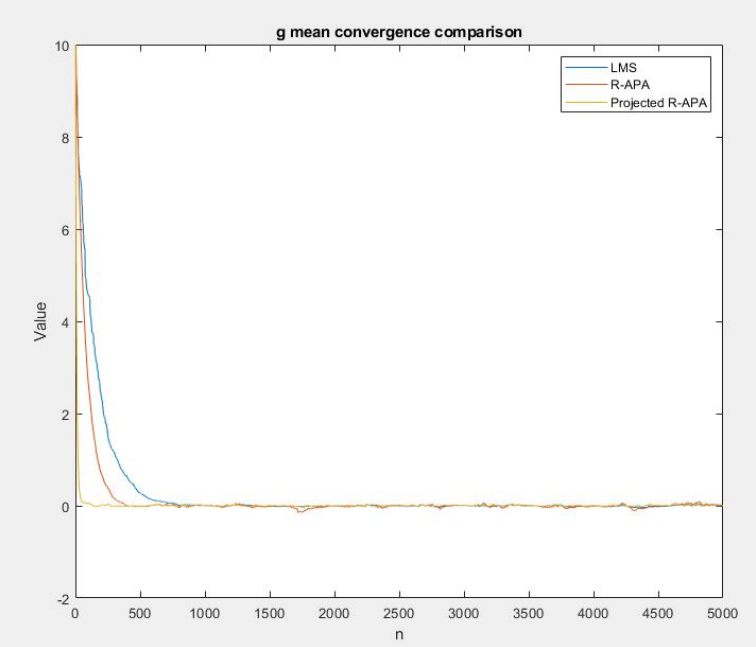

Figure 16: LMS, R-APA and Projected R-APA **g** mean convergence comparison.

Now, focusing on Projected R-APA, it is going to be analyzed how it responds to the changes in the system parameters such as the number of sensors corrupted by noise, the increase of the step-size and the increase of the projection order. As $\mathbf{s}_d$ it will be chosen the all ones signature. After moving into miscalibration, this signature is going to be modified.

Notice that the behavior of the system is to set the adaptive filter weights $\mathbf{g}^T$ so they cancel the same weights of $\mathbf{q}^T$ in which it is receiving the signal from a sensor that has high uncertainty. In the opposite way, it sets the adaptive weights to zero when it is performing well, so the output its only processed by the MRC filter. To end with, as long as the MRC filter is already in the probability simplex, all the adaptive weights steps must have zero mean.

In the calibration case, the following table contains firstly the effect of varying the number of noisy sensors, then increasing the step-size and finally the projection order.

Noisy Sensors	μ	Projection Order	Convergence Rate [n]	$\sigma^2_{opt}\|\sigma^2$
0	0.01	3	67	0.5/3.5
2	0.01	3	54	0.6/3.55
4	0.01	3	49	0.78/4.19
6	0.01	3	50	1.01/4.87
8	0.01	3	47	1.78/4.96
2	1	3	8	0.6/8.27
2	0.5	3	33	0.6/5.31
2	0.075	3	45	0.6/4.48
2	0.01	3	56	0.6/3.62
2	0.001	3	65	0.6/3.33
2	0.01	NLMS	57	0.6/3.69
2	0.01	3	52	0.6/3.66
2	0.01	5	51	0.6/3.35
2	0.01	7	48	0.6/3.52
2	0.01	9	47	0.6/3.91

Table 1: Impact of noisy sensors, μ and projection order at the convergence rate and at system's output σ^2 assuming Calibration.

The convergence rate is highly influenced by the Convex Set projection as it's bounding the possible adaptive weights to that set. This is why taking big step-size values is not the optimal design as it makes less precise for **g** to enclose near to the MMSE. Instead of that, the parameter selection criteria should be to choose a small μ and then increase the projection order until reaching the minimum output's variance.

In the following simulation it can be seen that a higher variance produce weight's to be unsteady, and lower ones make the convergence more stable,

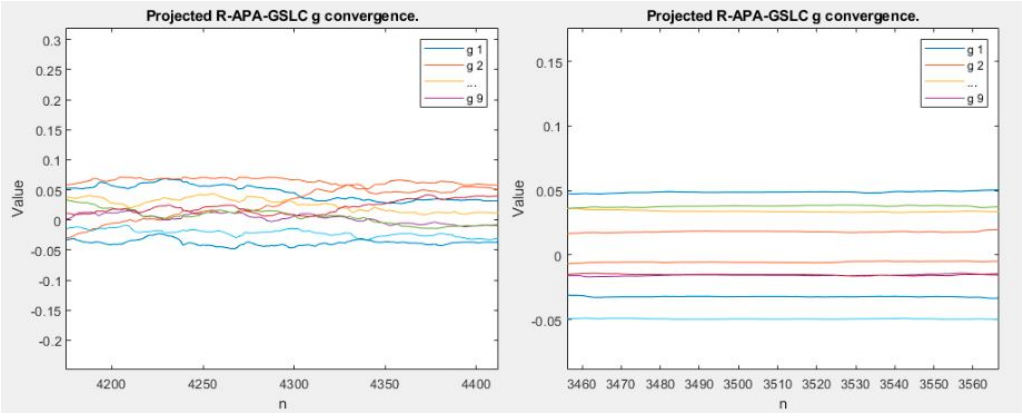

Figure 17: High(left simulation) vs Low(right simulation) **g** adaptive weights variance.

Now moving at the miscalibration situation, there are added as parameters the Aim and the forgetting factor. Knowing that the algorithm performs in the same way as the previous one in spite the scoping process in a single sample, the impact of the step-size and the projection order won't change. Furthermore, the step-size and the projection order have no impact in the self correlation calculation.

The most accurate Aim value is the corresponding to the sample where $\mathbf{R}_y$ is well estimated. The forgetting factor will vary depending on the experiment itself. In our environment, after multiple tries, I've decided to fix it at 0.002 making a proper estimation of the self correlation around 800 samples. Hence, fixing $\mu = 0.01$, the projection order at 5, λ = 0.002 and Aim = 800.

Noisy Sensors	Attenuated Sensors	$dist(\hat{s}_d, s_d)$	σ^2_{opt}/σ^2
0	2	0.04	0.6/4.34
2	2	0.22	0.80/5.05
4	2	0.25	1.13/6.41
6	2	0.21	2.54/7.09
8	2	0.17	6.25/7.67
2	1	0.31	0.69/3.91
2	3	0.33	0.96/4.24
2	5	0.36	1.56/6.32
2	7	0.22	4.16/11.73
0	9	0.04	5.00/8.36

Table 2: Miscalibration, impact of noisy and attenuated sensors at $\hat{s}_d$ and output variance.

From this results, It can be observed that the noise corrupts the self correlation matrix and as a consequence the first component of PCA. If the noise equally affects most or none of the sensors, the algorithm keeps aiming well. In the opposite case, if the noise affects the half of them, then the noise is interpreted as a signal and the non noisy sensors are taken as worse performing sensors. A solution to that could be to set a maximum variance at the input and if the sensors detect that it's being surpassed, drawn the sensor weight to 0. Looking at the attenuation results, if there is no noise added it doesn't matter the number of $s_{d_i} = 0$ sensors, the algorithm will always perform perfectly. The worst scenario is to have a lot of attenuated sensors and from the remaining ones, to have half of them corrupted by noise. Then we should increment the number of total sensors or the noise will be interpreted as a desired signal.

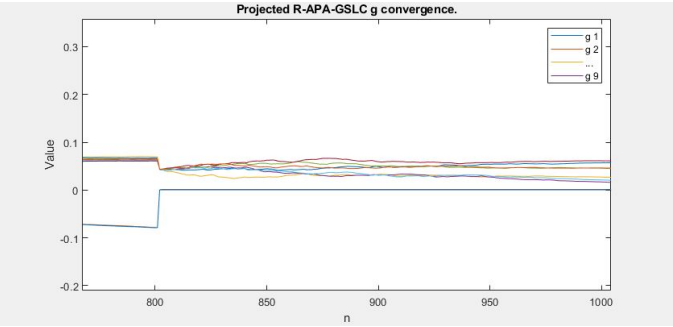

Figure 18: Miscalibration adaptive weights convergence under 2 attenuated sensors and 0 noisy sensors.

6 Conclusions and Future Work

The first conclusion that I have drawn from this thesis is that this research covers only a tiny part of the entire world of signal processing and data science. At the same time, while neural networks occupy the attention of most researchers, there is still utility in using traditional methods to solve current problems.
I would like to start with the work I could do to follow the course of the investigation: The paths are very diverse, but we should start by automating some user parameters based on the statistics provided by the signals of our model. Thus, the Aim value could be calculated from the output's variance stability, the forgetting factor of the input's covariance could also become adaptive and as seen in the State of the Art, there are better versions of the affine projection algorithm that have an adaptive projection order.
Following this trail, we have also seen that there are a lot of adaptive algorithms that can perform in GLSC, so an study where the comparison between convergence and the cost in the performance of the newest and most powerful converging methods such as Kalman Filters family or RLS could be done. It would be also really interesting to do the same study but resolving it with deep learning models. My personal opinion and prediction about these comparisons is that, since we are using the convex hull projection to converge, the solution offered in this thesis has an acceptable complexity and enough potency to compete with these other models. Finally, a list of designs in real life applications would culminate with this line of research.

Moving to more general conclusions, I have found that one of the most difficult challenges is to move from mathematics to the implementation of functional algorithms. Also emphasize the importance of all planning tasks and the difficulty of detecting and avoiding excessive information that leads to dead ends.

This first contact with research has also helped me to decide to continue learning in communications and information technologies, both autonomously and with the institution that has made this work possible, the UPC.

References

[1] Connected papers. URL:https://www.connectedpapers.com/main/8eec186ec80c4b2f55d408b5e28a757c26268fdf/Kalman-Filter%2C-Sensor-Fusion%2C-and-Constrained-Regression%3A-Equivalences-and-Insights/graph 2022.

[2] How does centering the data get rid of the intercept in regression and pca? URL:https://stats.stackexchange.com/q/22331 2016.

[3] Antonio Aguileta, Ramon Brena, Oscar Mayora, Erik Molino Minero Re, and Luis Trejo. Multi-sensor fusion for activity recognition—a survey. *Sensors*, 19, 09 2019.

[4] Nawal Alsaleh, Bilal Daass, Denis Pomorski, and Kamel Haddadi. Fast and real-time sensor-fault detection using shannon's entropy. In *2021 5th International Conference on Control and Fault-Tolerant Systems (SysTol)*, pages 273–278, 2021.

[5] A. Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences - an overview on methods, issues and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9, 09 2017.

[6] Thomas Moore and Daniel Stouch. *A Generalized Extended Kalman Filter Implementation for the Robot Operating System*. Springer International Publishing, Cham, 2016.

[7] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz, and Markus Lienkamp. A deep learning-based radar and camera sensor fusion architecture for object detection. pages 1–7, 10 2019.

[8] Kazuhiko. Ozeki. *Theory of Affine Projection Algorithms for Adaptive Filtering [electronic resource] / by Kazuhiko Ozeki.* Mathematics for Industry, 22. Springer Japan, Tokyo, 1st ed. 2016. edition, 2016.

[9] Hemanth Peddaneni, Deniz Erdogmus, Yadunandana Rao, Anant Hegde, and Jose Principe. Recursive principal components analysis using eigenvector matrix perturbation. volume 2004, pages 83 – 92, 10 2004.

[10] N.S.V. Rao. On fusers that perform better than best sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):904–909, 2001.

[11] Pavel Sakov and Peter Oke. Implications of the form of the ensemble transformation in the ensemble square root filters. *Monthly Weather Review - MON WEATHER REV*, 136:1042–1053, 03 2008.

[12] G. Seco-Granados, J.A. Fernandez-Rubio, and C. Fernandez-Prades. Ml estimator and hybrid beamformer for multipath and interference mitigation in gnss receivers. *IEEE Transactions on Signal Processing*, 53(3):1194–1208, 2005.

[13] Vaseghi SV. Advanced digital signal processing and noise reduction. *Advanced Digital Signal Processing and Noise Reduction*, 01 2008.

[14] Hans Wackernagel. Geir evensen: Data assimilation—the ensemble kalman filter, 2nd edn. *Mathematical Geosciences - MATH GEOSCI*, 42:1001–1002, 11 2010.

[15] Weiran Wang and Miguel Carreira-Perpiñán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. 09 2013.

[16] Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-sensor fusion in automated driving: A survey. *IEEE Access*, PP:1–1, 12 2019.

[17] Christopher Wikle and L. Berliner. A bayesian tutorial for data assimilation. *Physica D: Nonlinear Phenomena*, 230:1–16, 06 2007.

[18] Jan Williams, Olivia Zahn, and J. Nathan Kutz. Data-driven sensor placement with shallow decoder networks, 2022.

[19] Danni Wu, Zichen Liang, and Guang Chen. Lidar and radar-fusion 3d perception: a survey. *Intelligence & Robotics*, 2:105–29, 01 2022.

7 Appendix

Matlab Code:

Projected-RAPA.m

```
clear;

% Defining number of sensors and experiment samples:

sensors = 10;
samples = 1:5000;

% Input signal structure: y(n) = s_d * x(n) + n(n).

% Desired signal:

Data_array = 10*sin(0.002*pi*samples');

% Choose the Data signature:

Data_signature = [1;0;0;1;1;1;1;0;1;1];
Data_signature_aux = Data_signature;

% Init:

inputData_Matrix = zeros(length(Data_array),sensors);
noise_Matrix = zeros(length(Data_array),sensors);

% Choose the Noise variance:

noiseVar_array = sqrt([10;10;10;10;1;1;1;1;1;1]); % normrnd
   demands standard deviation.

% Forming y(n):

for c = 1:sensors

    inputData_Matrix(:,c) = Data_array * Data_signature(c);
    n = normrnd (0, noiseVar_array(c), length(Data_array));
    noise_Matrix(:,c) = n(:,1);
    clear("n");

end
```

```
noise_Matrix = noise_Matrix';
inputData_Matrix = inputData_Matrix' + noise_Matrix;

%% Filters.

% q'

mrc_filter = Data_signature'./(Data_signature'*Data_signature);

% ------Algorithm to finf B' from the data signature------

B_Matrix = [-Data_signature(2:end), eye(sensors-1,sensors-1)]';

interference_cancelation = B_Matrix' * Data_signature;

p=2;
while (mean(interference_cancelation)~=0)

    B_Matrix(p-1,:) = B_Matrix(p,:);
    B_Matrix(p,:) = -cat(2,Data_signature(1:p-1)',Data_
        signature(p+1:end)');
    interference_cancelation = B_Matrix' * Data_signature;
    p = p+1;
end

norms = sqrt(sum(B_Matrix.^2));
B_Matrix = norms .* B_Matrix;

%% Generic Adaptive Parameter Initialization:

adaptive_filter = 10.*ones(sensors-1,1); % g init.
mu_init = 0.004; % Step-Size.

%% APA Algorithm initialization:

APA_P = 5; % Projective order.
memory_input_Matrix = zeros(sensors-1,APA_P); % FIFO's
memory_error_array = zeros(APA_P,1);
memory_reference_array = zeros(APA_P,1);
error_variance = 0;
```

```
% --------- Improved knowledge: Convex Set ---------
g_Convex_Hull_Points = eye(sensors,sensors) * B_Matrix*inv(B_
    Matrix'*B_Matrix);
ConvexHullCenter = round(ones(1,sensors) * g_Convex_Hull_Points
    /sensors)';

% If under misscalibration, init:

Data_signature = [1;1;1;1;1;1;1;1;1;1];
CovarianceMatrix = 0.*eye(sensors, sensors);
FFactor = 0.002; % Forgetting Factor
Aim = 1000; % Sample at which we aim.

% Run the Algorithm:

for n=samples

ploting(:,n) = adaptive_filter'; % Taking the g values
% Ry Estimation:
CovarianceMatrix = (1-FFactor) .* CovarianceMatrix + (FFactor)
    .* inputData_Matrix(:,n) * inputData_Matrix(:,n)';
     % If we are not aiming:
     if n~=Aim
         % Variable StepSize Calculation:
         mu = mu_init*norm(adaptive_filter - ConvexHullCenter);

          % Memory input and reference FIFO update:
         for c=APA_P:-1:2
             memory_input_Matrix(:,c) = memory_input_Matrix(:,c
                 -1);
             memory_reference_array(c) = memory_reference_array(
                 c-1);
         end
         memory_input_Matrix(:,1) = B_Matrix'*inputData_Matrix
             (:,n);
         memory_reference_array(1) = mrc_filter*inputData_Matrix
             (:,n);
         % Error array calculation
         memory_error_array = memory_reference_array - memory_
             input_Matrix'*adaptive_filter;
         % Plot the output:
         Signal_Plot(n) = memory_error_array(1);
```

```
        % APA weight update:
        aux = memory_input_Matrix'*memory_input_Matrix+1e-3*eye
            (APA_P,APA_P);
        adaptive_filter = adaptive_filter + (mu*memory_input_
            Matrix*inv(aux)*memory_error_array);

        %----------Simplex Projection----------
        h = mrc_filter - adaptive_filter' * B_Matrix'; %
            Calculate h
        [u, sortIndx] = sort(h,'descend'); % Sort h
        sorted_signature = Data_signature(sortIndx); % Sort sd
        for j = 1:length(u) % Find maximum n
            p(j) = u(j) + (1/sum(sorted_signature.^2))*(1-sum(u
                *sorted_signature));
            if (p(j)>0)
                auxj = j;
            end
        end
        sum_sign = 0; % Calculate lambda
        sum_weight = 0;
        for k = 1:auxj
            sum_sign = sum_sign + sorted_signature(k)^2;
            sum_weight = sum_weight + sorted_signature(k)*u(k);
        end
        lamb = (1/sum_sign)*(1-sum_weight);

        for i= 1:length(u) % h projection
            if (h(i)+Data_signature(i)*lamb) > 0
                h(i) = h(i)+Data_signature(i)*lamb;
            else
                h(i) = 0;
            end
        end
        % g projection
       adaptive_filter = -inv(B_Matrix'*B_Matrix)*B_Matrix'*h';

    % If we are aiming:
    else
        % Eigenvector calculation
        [CovEigVect,CovEigVal] = eig(CovarianceMatrix);
        [val,ind] = max(diag(CovEigVal));
        Data_signature = abs(CovEigVect(:,ind));
        % Data signature estimation
        Data_signature = Data_signature./(max(Data_signature));
```

```
        % Calculate q
        mrc_filter = Data_signature'./(Data_signature'*Data_
            signature);
        % Calculate B
        B_Matrix = [-Data_signature(2:end), eye(sensors-1,
            sensors-1)]';
        norms = sqrt(sum(B_Matrix.^2));
        B_Matrix = norms .* B_Matrix;
        % Calculate Convex Set center
        g_Convex_Hull_Points = eye(sensors,sensors) * B_Matrix*
            inv(B_Matrix'*B_Matrix);
        ConvexHullCenter = round(ones(1,sensors) * g_Convex_
            Hull_Points/sensors)';
    end
end

% Plotting g weights
figure()
plot(samples,(ploting))
title('g mean convergence comparison')
legend('LMS','R-APA','Projected R-APA');
xlabel('n')
ylabel('Value')

% Plotting input and output
figure()
subplot(2,1,1)
plot(samples,(inputData_Matrix))
title('Projected R-APA GSLC Input')
legend('Signal Sensor 1','Signal Sensor 2','...','Signal Sensor
    10');
xlabel('n')
ylabel('Value')

subplot(2,1,2)
plot(1000:5000,Signal_Plot(1000:end))
title('Projected R-APA GSLC Output')
legend('Output');
xlabel('n')
ylabel('Value')
```