POLITECNICO DI MILANO

SCUOLA DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Specialistica in Ingegneria dell'Automazione

REJECTING ATMOSPHERIC TURBULENCE IN ADAPTIVE OPTICS USING LATTICE FILTERS

Relatore: Prof. MARCO LOVERA

Correlatore: Prof. MICHEL VERHAEGEN

Ing. JACOPO ANTONELLO

Tesi di Laurea di:

ALESSANDRO SCOTTI

Matr. 755248

Anno Accademico 2010–2011

To my Family.

Abstract

This work studies from a theoretical and practical point of view the application of lattice filters, as proposed by S. Gibson (UCLA), for the rejection of a non-stationary disturbance. In Adaptive Optics applications, the effect of turbulence in some layers of the atmosphere severely affects the imaging quality. Such an effect can be modeled as a non-stationary disturbance affecting the phase distribution in the pupil of a ground based telescope. The application of lattice filters in this setting can overcome the common performance limitations usually encountered by employing linear time invariant control in Adaptive Optics implementations. The performance of lattice filters is compared with another adaptive control methodology, originally proposed by Ellerbroek and Rhoadarmer (1998, 2001), where a recursive least square algorithm is applied. Realistic simulations have been carried out with both stationary and non-stationary turbulence. Some experimental validations have also been performed using an optical breadboard with a deformable mirror.

Sommario

Questo lavoro studia da un punto di vista sia teorico sia pratico l'applicazione dei lattice filters, come proposta da S. Gibson (UCLA), per contrastare un disturbo non stazionario. Nelle applicazioni di Adaptive Optics, l'effetto della turbolenza in alcuni livelli dell'atmosfera colpisce gravemente la qualità delle immagini. Questo tipo di effetto può essere modellato come un disturbo non stazionario che interessa la distribuzione della fase nella pupilla di un telescopio terrestre. L'impiego dei lattice filters in questo ambito permette di superare le comuni limitazioni di performance che si incontrano implementando un controllo lineare tempo invariante nelle applicazioni di Adaptive Optics. La performance dei lattice filters viene confrontata con un'altra tecnica di controllo adattativo, originariamente proposta da Ellerbroek e Rhoadarmer (1998, 2001), nella quale viene impiegato un algoritmo recursive least square. Sono state compiute delle simulazioni realistiche con turbolenza sia stazionaria sia non stazionaria. Alcune validazioni sperimentali sono poi state realizzate utilizzando una breadboard ottica equipaggiata di specchio deformabile.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank Prof. Michel Verhaegen for his amazing availability in welcoming me in Delft without knowing me and for his guide in this very interesting Thesis.

Then, I would like to thank my Italian supervisor Prof. Marco Lovera for his support from Italy and for his courtesy.

Another (maybe the biggest one) thanks to the PhD student Jacopo Antonello, my daily supervisor in Delft, that spent a lot of time to guide me and that was always available when I was in troubles, even with his heavy work to do.

A very big thanks also to Rufus Fraanje for his help and his availability for some troubles I had, and to the whole adaptive optics group for the stimulating meetings and presentations.

Of course I want to thank Delft too and all the people I knew during this awesome experience that I will never forget.

And the most important thanks is to my mother and to my father that supported this experience economically and emotionally, and that are the most important thing in my life. Thanks.

Chapter 1

Introduction

For thousands of years, astronomical observations played a crucial role in our attempts to reveal the mysteries of the universe. The introduction of the telescope at the beginning of the seventeenth century, resulted in a big leap in the angular resolution with respect to the naked-eye. Ground-based telescopes are nowadays an indispensable tool in astronomy.

The spherical wavefront phase of a light beam coming from a star can be considered perfectly flat because of the distance. When it arrives to the Earth's atmosphere, it has to go through several layers of atmospheric turbulence that arises from large scale temperature inhomogeneities caused by solar heating: by mixing air of different temperatures, it is responsible for random local fluctuations in the refractive index. It means that the light beam wavefront phase is no longer flat when it arrives to the telescope aperture: some parts of the incoming light beam will be delayed with respect to others parts. We have a distortion in the wavefront phase that get the angular resolution worse. This can lead to twinkling, that is a random intensity variation of light, quivering and spreading of stars. That's the reason why we can't achieve the diffraction limit with ground-based telescopes: we can say them to be seeing limited.

To know more details about angular resolution and the effects of turbulence on images, you can see [1],[2].

Adaptive Optics (AO) is a scientific and engineering discipline which goal is to reduce the wavefront phase distortion by controlling a deformable mirror (DM). To generate the control law, a feedback of the wavefront phase error is needed and it is given by a wavefront sensor (WFS).

In the second chapter of this Thesis the optical problem is introduced. The Classical AO control approach with its limitations is presented. Then the Adaptive AO control approach proposed by Steve Gibson is described in details: this approach relies on lattice filters. In the third chapter lattice filters are described and they are compared with the recursive least square (RLS) algorithm. In the fourth chapter the real Multi Input Multi Output (MIMO) system is described and a technique to decouple the system is proposed. In the fifth chapter the simulations are explained and the most important results are shown. In the sixth chapter the real experiment setting is presented and some experimental results are shown.

Chapter 2

Classical control loop VS adaptive control loop

In this Chapter, we will describe the optical problem and the Classical solution developed by AO. Then, we will see why this classical approach is not enough and why we need an Adaptive approach. We will focus on the one proposed by Steve Gibson.

2-1 The optical problem

To explain the optical problem, consider the schematic drawing in Figure 2-1.

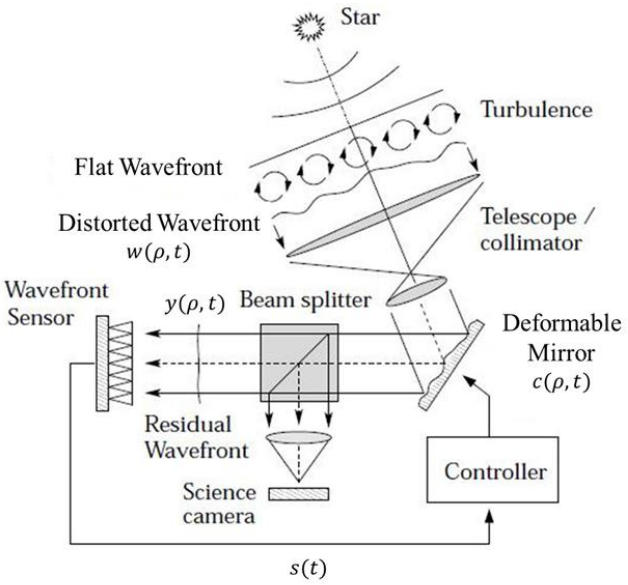


Figure 2-1: optical problem

When the beam coming from a star arrives at the outer layer of the atmosphere, it has a perfectly flat wavefront phase. Then the beam has to go through the atmosphere, where turbulence generates a distortion in the wavefront phase. We denote the distortion $w(\rho, t)$: it is a function of the spatial position in the telescope aperture $\rho \in \mathbb{R}^2$ and of the time t.

The AO system tries to compensate this wavefront phase distortion by actively introducing an opposite distortion with a DM: we denote $c(\rho, t)$ the phase correction induced by the DM. The residual wavefront phase $y(\rho, t)$ is given by the following expression:

$$y(\rho, t) = w(\rho, t) + c(\rho, t) \tag{2-1}$$

After applying the wavefront correction, a beam splitter divides the light beam in two parts: the first part of the corrected light beam leaves the AO system and it is used by the science camera to generate an image of the object of interest; the remainder of the light is directed to the WFS, which provides quantitative information about residual phase.

With the WFS, an estimation of the phase is available. Then we can see that the WFS measure is the feedback for the controller that generates the control law for the DM.

By rejecting the wavefront phase distortions, AO is able to reduce the detrimental effect of atmospheric turbulence on the imaging process: large ground-based telescopes may reach close to diffraction limit. Figure 2-2 is an example of what we can see without AO and how the image improves with AO:

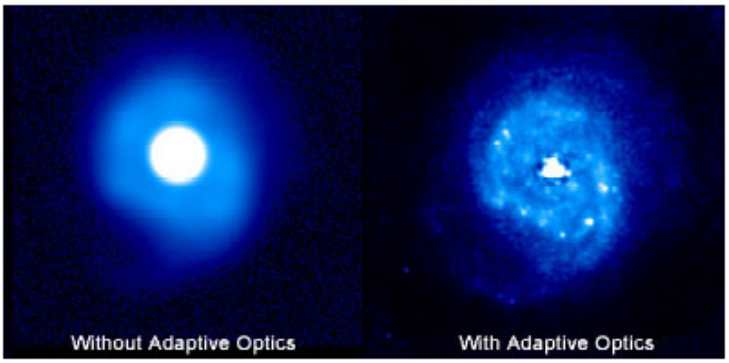


Figure 2-2: example of an image without and with AO

2-2 The Classical AO control approach

The classical AO control approach consists in a linear time invariant (LTI) control loop: the regulator can be a simple proportional integrative derivative (PID) controller or an optimal (LQG - linear quadratic Gaussian - or H_2) controller. Initially we simplify the treatment by considering a Single Input Single Output (SISO) version of the system (The MIMO generalization will be discussed in section 4-1):

- $w(\rho, t) = w(t) \in \mathbb{R}$

- $c(\rho, t) = c(t) \in \mathbb{R}$
- $y(\rho, t) = y(t) \in \mathbb{R}$

Therefore the drawing in Figure 2-1 can be represented with the block diagram in Figure 2-3 where there is the classical controller $R(z)$ (for example an optimal controller), the deformable mirror $D_M(z)$, the wavefront sensor $W_{FS}(z)$, the latency due to the measurement delay z^{-1} and the time-invariant wavefront phase distortion w. This distortion is induced by the atmospheric turbulence, which is modeled by the stationary transfer function $T(z)$. d is a zero-mean white noise process.

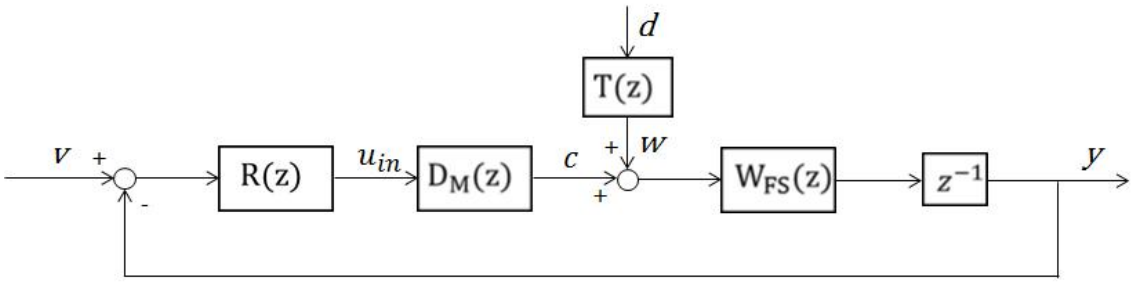


Figure 2-3: classical AO loop

The main delay is the reading time of the WFS, that's the reason why this is the only delay considered in the scheme.

We can define $G(z)$ as the closed-loop transfer function from the signal v to the residual phase y and $S(z)$ as the transfer function from the disturbance w to the residual phase y:

$$G(z) = \frac{R(z)D_M(z)W_{FS}(z)z^{-1}}{1 + R(z)D_M(z)W_{FS}(z)z^{-1}} \tag{2-2}$$

$$S(z) = \frac{W_{FS}(z)z^{-1}}{1 + R(z)D_M(z)W_{FS}(z)z^{-1}} \tag{2-3}$$

Notice that this control loop is not adaptive in the sense in which the term *adaptive* is used in control and filtering literature, where it normally refers to the practice of updating the control and/or filter gains in real time.

By defining (2-2) and (2-3) we can represent the block diagram in Figure 2-3 in a more schematic way (Figure 2-4).

The new signal $\omega \in \mathbb{R}$ in Figure 2-4 represents the disturbance signal in the new diagram: it was between the DM and the WFS (Figure 2-3) , while now we have moved it after the plant $G(z)$. We can write that:

$$y = G(z)v + \omega \tag{2-4}$$

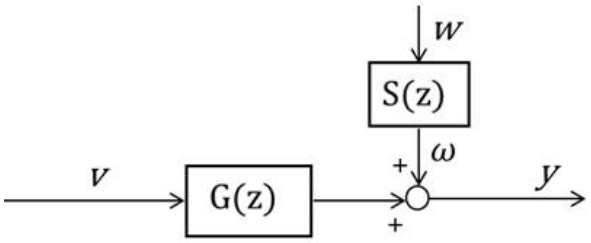


Figure 2-4: classical AO loop - schematic diagram

2-3 The classical approach limitation

The atmospheric stationary turbulence w can be represented in several ways [3]. We will consider an Auto Regressive (AR) model of order M. However, the most interesting signal is ω (not w), that is the disturbance in the new block diagram (Figure 2-4). This signal is generated by $S(z)$ which is a time-invariant transfer function (TF). Therefore we can also model ω as an AR process, and we can write:

$$\omega(k) = \sum_{j=1}^{M} A_j \omega(k-j) + d(k) \tag{2-5}$$

Where $d(k)$ is a zero-mean white noise process and A_j are the AR coefficients used to model the turbulence.

In the Kolmogorov model, the atmospheric turbulence can be described by four parameters, that are represented by the AR coefficients A_j:

- L_0: is the *outer scale of turbulence*.
- r_0: is the *Fried parameter*;
- v_x: is the wind velocity in x direction;
- v_y: is the wind velocity in y direction;

The goal of the classical control loop is to find, for example, the optimal controller $R(z)$ that minimizes this cost function:

$$J = E\left\{y^T(k)y(k) + u_{in}^T(k)Qu_{in}(k)\right\}, \tag{2-6}$$

where Q is a positive-defined regularization matrix that represents the weight for the control variable.

Now suppose we have two different turbulence parameters vectors:

- $\boldsymbol{p_1} = [L_0^{(1)}, r_0^{(1)}, v_x^{(1)}, v_y^{(1)}]$ at time t_1;

- $\boldsymbol{p_2} = [L_0^{(2)}, r_0^{(2)}, v_x^{(2)}, v_y^{(2)}]$ at time $t_2 \neq t_1$;

At time t_1, we can find [1] the optimal controller $R_1(z)$ that minimizes the cost function J for the turbulence parameters vector $\boldsymbol{p_1}$. Likewise, at time t_2, we can find the optimal controller $R_2(z)$ that minimizes the cost function J for the turbulence parameters vector $\boldsymbol{p_2}$.

The problem of the classical control scheme is found when turbulence is non-stationary, because it is time invariant. If we have tuned the optimal controller $R_1(z)$ to take into account the first turbulence parameters vector $\boldsymbol{p_1}$, as the turbulence parameters tend to $\boldsymbol{p_2}$, the controller $R_1(z)$ is non optimal anymore.

The most advanced concepts about optimal controllers rely upon *a priori* knowledge of the turbulence statistics to improve their performance. The required *a priori* knowledge is obtained by matching physical parameters to the expected turbulence conditions or are acquired in a separate identification experiment [1].

Since atmospheric parameters like the wind velocity and the atmospheric turbulence strength change on time scales of minutes, it becomes necessary to constantly update the controller to the changing atmospheric statistics in order to prevent a loss in performance. To account for non-stationary turbulence and track the changes, we need to introduce an adaptive control loop.

2-4 The Adaptive control approach

Different adaptive control schemes have been proposed. The one proposed by Ellerbroek and Rhoadarmer (1998, 2001) relies on a RLS algorithm to adaptively optimize the wavefront reconstructor of an AO system on the basis of closed-loop WFS measurements [1]. Another adaptive control strategy is based on lattice filtering: this approach relies on Internal Model Control (IMC), because we have a reconstruction of the open-loop WFS signal ($\hat{\omega}$) by subtracting the influence of the DM wavefront corrections from the closed-loop measurements y (this is clearer in Figure 2-8).

In this Thesis we will present and analyze in details the second one, proposed by Gibson [4], [5], [6], [7], [8], [9], [10], [11], [12].

This control strategy employs augmentation of a LTI classical feedback loop with an adaptive loop. Therefore we obtain the following block diagram (Figure 2-5):

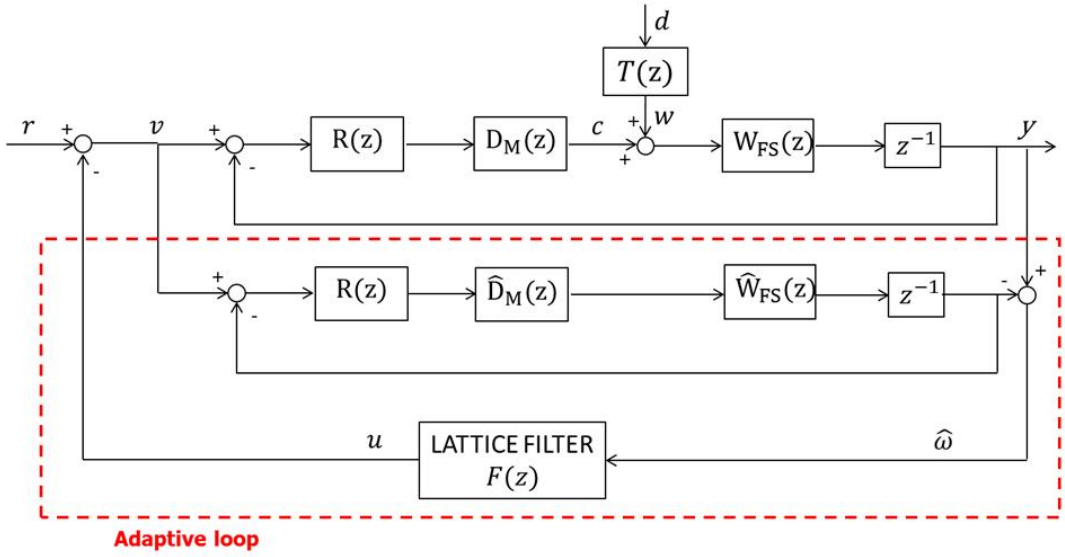


Figure 2-5: classical loop augmented by adaptive loop

Where $\hat{D}_M(z)$ is a first-order model of the deformable mirror, $\hat{W}_{FS}(z)$ is a first-order model of the wavefront sensor, and $F(z)$ is the lattice filter and represents the adaptive controller of the loop. $R(z)$ can be tuned like a simple proportional integrative (PI) controller.

Remembering how we defined $G(z)$, we can observe that the closed-loop with $R(z)$, $\hat{D}_M(z)$, $\hat{W}_{FS}(z)$ and z^{-1}, is an estimation of the closed-loop TF from v to y: so we can call this loop $\hat{G}(z)$ (Figure 2-6).

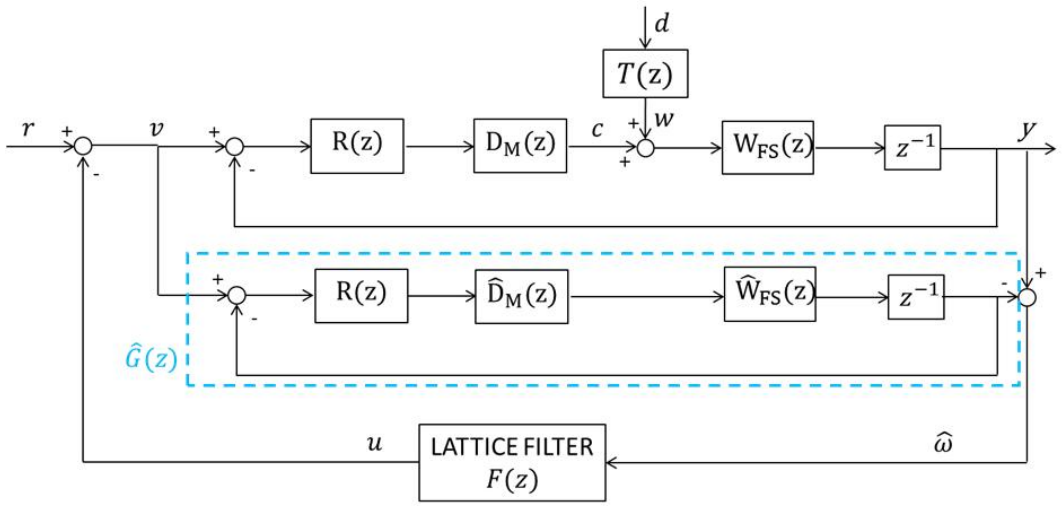


Figure 2-6: classical loop augmented by adaptive loop - $\hat{G}(z)$

To highlight the delay in the next block diagrams, we can shift the delay block before the lattice filter and rework the block diagram like in Figure 2-7, but we have to redefine $G(z)$ and $S(z)$ before:

$$G(z) = \frac{R(z)D_M(z)W_{FS}(z)}{1 + R(z)D_M(z)W_{FS}(z)z^{-1}} \tag{2-7}$$

$$S(z) = \frac{W_{FS}(z)}{1 + R(z)D_M(z)W_{FS}(z)z^{-1}} \tag{2-8}$$

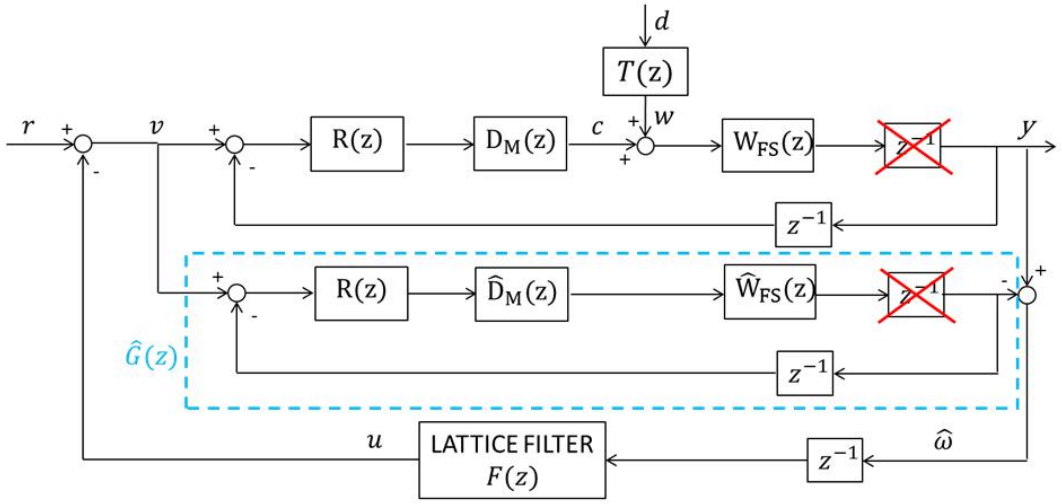


Figure 2-7: classical loop augmented by adaptive loop - shifted delay

Then it is possible to represent the classical AO loop augmented by the adaptive loop with a compact diagram (Figure 2-8). We can observe that the input of the lattice filter is an

estimation of the atmospheric turbulence ω. Indeed, we have that $y = G(z)v + \omega$; it follows that $y - \hat{G}(z)v = \hat{\omega}$. The output of the lattice filter is the control signal u. The reference r is zero because we want a flat residual phase.

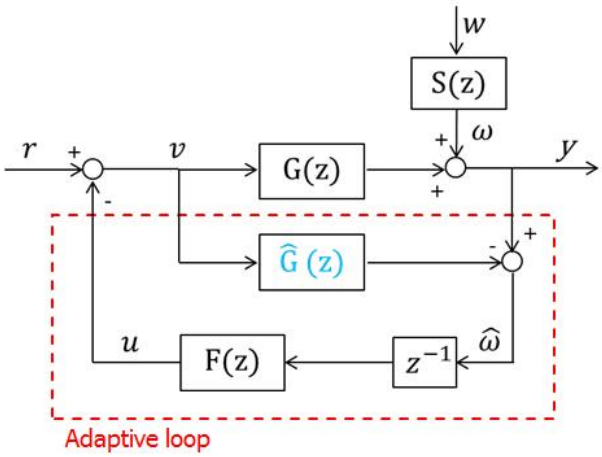


Figure 2-8: classical loop augmented by adaptive loop - compact scheme

The lattice filter block $F(z)$ is actually a copy of another identical block in the bottom part of the diagram, whose gains are updated by a lattice algorithm (Figure 2-9):

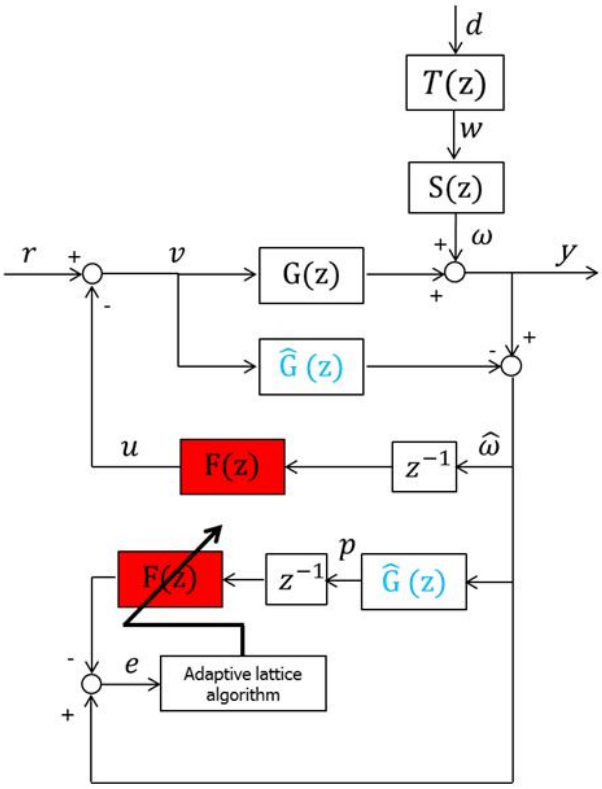


Figure 2-9: adaptive loop

The goal of the lattice algorithm is to minimize the signal error e that is given by this expression:

$$e = \hat{\omega} - z^{-1}F(z)\hat{G}(z)\hat{\omega} = \hat{\omega}(1 - z^{-1}F(z)\hat{G}(z)) \tag{2-9}$$

We can observe that if there wasn't the delay z^{-1}, it would be enough to have $F(z) = \hat{G}(z)^{-1}$ to minimize the error signal e. But because of the delay, we need an adaptive lattice filter whose gains are updated to replace $\hat{G}(z)^{-1}$ but also to predict the atmospheric turbulence $\omega(i)$ because the transfer function $T(z)$ is a non-stationary TF and the delay in the loop allows only to know $\omega(i-1)$. This technique is called Adaptive Inverse Control [13].

We can now define the TF from r to y and the TF from ω to y

$$H_{ry}(z) = \frac{G(z)}{1 + z^{-1}F(z)(G(z) - \hat{G}(z))} \tag{2-10}$$

$$H_{\omega y}(z) = \frac{1 - z^{-1}F(z)\hat{G}(z)}{1 + z^{-1}F(z)(G(z) - \hat{G}(z))} \tag{2-11}$$

Notice that if the model $\hat{G}(z)$ is identical to $G(z)$, then the lattice filter $F(z)$ does not affect the stability of the system. But if the modeling error $G(z) - \hat{G}(z)$ is sufficiently large, the adaptive loop can cause the system to be unstable [8]. If lattice filter works perfectly and the modeling error is small, we manage to obtain $H_{wy} = 0$ (i.e. total turbulence rejection).

In the next chapter we will discuss about lattice filters to better understand how they work and what is their difference from RLS.

Chapter 3

Lattice Filters

In the previous Chapter we chose to use lattice filter instead of RLS in the adaptive control loop. In this Chapter, we will describe how lattice filters work and the features that make them interesting to compare with RLS. We are going to explain the *a posteriori*-based lattice filter. In literature you can also find the *a priori*-based lattice filter, but it is not useful for our application. To know more about the *a priori* lattice filters, you can refer to [14]. Actually, there are several ways to implement lattice filter (like normalized lattice filter or array-based lattice filter), but they are only different representations of the same methodology [14]. The *a posteriori*-based is the simplest one.

3-1 Exponentially-weighted recursive least-square algorithm (λ-RLS)

We have already seen that stationary turbulence can be modeled with an AR process of order M. If we consider a non-stationary turbulence, we can write:

$$\omega(k) = \sum_{j=1}^{M} A_j(k)\omega(k-j) + d(k) \tag{3-1}$$

where the coefficients $A_1(k), A_2(k), ..., A_M(k)$ are time-variant, $d(k) \approx WN(0, \lambda^2)$.

Now suppose we want to minimize the error signal e with an exponentially-weighted RLS algorithm, not with a lattice filter (Figure 3-1). Notice that we can use RLS algorithm because the least square (LS) problem has been developed to be used with AR processes, and $F(z)$ is used to learn how to reject ω.
The exponentially-weighted LS problem seeks the parameters vector $\boldsymbol{\vartheta}_i$ at time i that solves this problem:

$$\min_{\vartheta} \left[\lambda^{i+1} \boldsymbol{\vartheta}^T \Pi \boldsymbol{\vartheta} + \sum_{j=1}^{i} \lambda^{i-j} \left| \hat{\omega}(j) - \boldsymbol{\psi}_j \boldsymbol{\vartheta} \right|^2 \right] \tag{3-2}$$

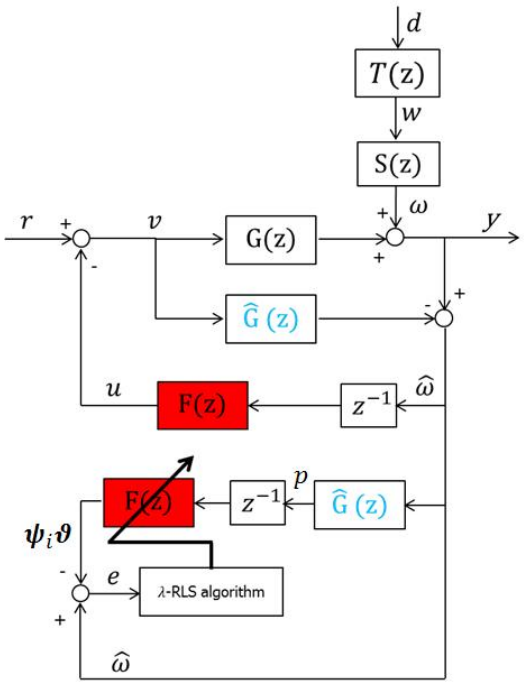


Figure 3-1: adaptive control with only a λ-RLS algorithm

Where Π is a positive-defined regularization matrix, and λ is the forgetting factor ($0 \ll \lambda \leq 1$). The signal error we want to minimize is e and it is defined (at time i) by

$$e(i) = \hat{\omega}(i) - \boldsymbol{\psi}_i \boldsymbol{\vartheta}_i \tag{3-3}$$

The signal $\hat{\omega}(i)$ is the AR process we don't know, and it is a number (estimation of the real disturbance $\omega(i)$). We assume that $\hat{\omega}(i)$ can be modeled as:

$$\hat{\omega}(i) = A_1(i)\hat{\omega}(i-1) + A_2(i)\hat{\omega}(i-2) + ... + A_M(i)\hat{\omega}(i-M) + d(i) \tag{3-4}$$

Then we can observe that the signal $\hat{\omega}$ is filtered by a time-invariant known TF ($\hat{G}(z)$). So we can call p the output of $\hat{G}(z)$ in the bottom part of the diagram in Figure 3-1. Observe that p is related to $\hat{\omega}$ and we assume p can be modeled with an AR process of order M. Now, we can define $\boldsymbol{\psi}_i \boldsymbol{\vartheta}_i$ as:

$$\begin{aligned} \boldsymbol{\psi}_i \boldsymbol{\vartheta}_i &= C_1(i)p(i-1) + C_2(i)p(i-2) + ... + C_M(i)p(i-M) = \\ &= \begin{bmatrix} p(i-1) & p(i-2) & \cdots & p(i-M) \end{bmatrix} \begin{bmatrix} C_1(i) \\ C_2(i) \\ \vdots \\ C_M(i) \end{bmatrix} \end{aligned} \tag{3-5}$$

where $\boldsymbol{\psi}_i$ is the observation vector at time i. The goal of the RLS algorithm is to find the parameters vector $\boldsymbol{\vartheta}_i = [C_1(i)C_2(i)...C_M(i)]^T$ that minimizes the error signal e at time i (equation (3-3)) , starting by $\boldsymbol{\vartheta}_{i-1}$.

The λ-RLS algorithm has this expression:

$$\boldsymbol{\vartheta}_i = \boldsymbol{\vartheta}_{i-1} + P_i \boldsymbol{\psi}_i^T \left[\hat{\omega}(i) - \boldsymbol{\psi}_i \boldsymbol{\vartheta}_{i-1}\right] \tag{3-6}$$

$$P_i = \frac{1}{\lambda}\left(P_{i-1} - \frac{P_{i-1}\boldsymbol{\psi_i}^T \boldsymbol{\psi_i} P_{i-1}}{\lambda + \boldsymbol{\psi_i} P_{i-1} \boldsymbol{\psi_i}^T}\right) \tag{3-7}$$

where P_i is a square matrix of dimension $M \times M$ and $P_0 = \Pi^{-1}$.
This formula shows that we can only time-update the parameters vector $\boldsymbol{\vartheta}$ from time $t = i$ to time $t = i + 1$. This is one of the RLS algorithm's limitations, it is a fixed-order algorithm; the other limitation is the computational cost: it is $O(M^2)$ at every time step.

3-2 *A posteriori*-based Lattice Filters description

Now, consider the following exponentially-weighted LS cost function

$$\min_{\vartheta_M} \left[\lambda^{i+1} \boldsymbol{\vartheta}_M^T \Pi_M \boldsymbol{\vartheta}_M + \sum_{j=1}^{i} \lambda^{i-j} \left| \hat{\omega}(j) - \boldsymbol{\psi}_{M,j} \boldsymbol{\vartheta}_M \right|^2 \right] \tag{3-8}$$

that can be also written in a vectorial form:

$$\min_{\vartheta_M} \left[\lambda^{i+1} \boldsymbol{\vartheta}_M^T \Pi_M \boldsymbol{\vartheta}_M + (\boldsymbol{Y}_i - H_{M,i} \boldsymbol{\vartheta}_M)^T \Lambda_i (\boldsymbol{Y}_i - H_{M,i} \boldsymbol{\vartheta}_M) \right] \tag{3-9}$$

where:

$$\Lambda_i = \text{diag}\left(\lambda^i, \lambda^{i-1}, ..., \lambda, 1 \right) \tag{3-10}$$

$$\Pi_M = \eta^{-1} \text{diag}\left(\lambda^{-2}, \lambda^{-3}, ..., \lambda^{-(M+1)} \right) \tag{3-11}$$

and η is a positive scalar that weighs the a priori estimation of the parameters vector $\boldsymbol{\vartheta}_M$.

This cost function is the same of (3-2), but we have changed the notation highlighting the order M of the LS problem. Indeed, to analyze order-recursive problems (like lattice filters), it is necessary to indicate both the size of a vector/matrix and the time instant at which it becomes available.

For example, if we consider the parameters vector $\boldsymbol{\vartheta}_i$ at time i that has size M (i.e. solution of a LS problem of order M), we shall write $\boldsymbol{\vartheta}_{M,i}$. In the same way we shall write $H_{M,i}$ to refer to an observations matrix with column dimension M and with data up to time $i-1$. Similarly, we shall write Π_M to refer to an $M \times M$ regularization matrix.

In the cost function (3-8) we define:

$$\boldsymbol{Y}_i = \begin{bmatrix} \hat{\omega}(1) \\ \hat{\omega}(2) \\ \vdots \\ \hat{\omega}(i) \end{bmatrix} \tag{3-12}$$

$$H_{M,i} = \begin{bmatrix} \boldsymbol{\psi}_{M,1} \\ \boldsymbol{\psi}_{M,2} \\ \vdots \\ \boldsymbol{\psi}_{M,i} \end{bmatrix} = \begin{bmatrix} p(0) & 0 & \cdots & 0 \\ p(1) & p(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p(i-1) & p(i-2) & \cdots & p(i-M) \end{bmatrix} \tag{3-13}$$

Now we need to define:

$\hat{\boldsymbol{Y}}_{M,i} = H_{M,i} \boldsymbol{\vartheta}_{M,i} =$ the order M estimation of $\boldsymbol{Y}_i$

$\hat{\omega}_M(i) = \boldsymbol{\psi}_{M,i}\boldsymbol{\vartheta}_{M,i}$ = the order M estimation of $\hat{\omega}(i)$, that is the last entries of $\boldsymbol{Y}_i$

$\hat{\omega}_{M+1}(i) = \boldsymbol{\psi}_{M+1,i}\boldsymbol{\vartheta}_{M+1,i}$ = the order $M+1$ estimation of $\hat{\omega}(i)$

where $\boldsymbol{\vartheta}_{M+1,i}$ is the solution of the following LS problem:

$$\min_{\vartheta_{M+1}} \left[\lambda^{i+1}\boldsymbol{\vartheta}_{M+1}^T \Pi_{M+1}\boldsymbol{\vartheta}_{M+1} + \sum_{j=1}^{i} \lambda^{i-j} \left| \hat{\omega}(j) - \boldsymbol{\psi}_{M+1,j}\boldsymbol{\vartheta}_{M+1} \right|^2 \right] \tag{3-14}$$

Notice that in this situation, the real signal is $\hat{\omega}$ and the order M estimation of $\hat{\omega}$ is $\hat{\omega}_M$.

The *a posteriori* error vector is defined as:

$$\boldsymbol{r}_{M,i} = \boldsymbol{Y}_i - H_{M,i}\boldsymbol{\vartheta}_{M,i} \tag{3-15}$$

So the last element of the *a posteriori* error vector will be the *a posteriori* error:

$r_M(i) = \hat{\omega}(i) - \hat{\omega}_M(i)$ = the order M *a posteriori* error;

$r_{M+1}(i) = \hat{\omega}(i) - \hat{\omega}_{M+1}(i)$ = the order $M+1$ *a posteriori* error;

Lattice filters are not concerned with updating $\boldsymbol{\vartheta}_{M,i}$ to $\boldsymbol{\vartheta}_{M+1,i}$: the purpose of lattice filters is to update the *a posteriori* error from $r_M(i)$ to $r_{M+1}(i)$ with a direct recursive formula. This kind of lattice is called the *a posteriori*-based lattice filter:

$$\boxed{r_{M+1}(i) = r_M(i) - \kappa_M(i)b_M(i)}$$

This is an important result as it means that you can recursively calculate the *a posteriori* error from order 1 to M with the same computational burden that you have if you only calculate the order M *a posteriori* error: the computational cost is $O(M)$ at every time step.

Once you have calculated $r_1(i), r_2(i), ..., r_M(i)$, you can solve this problem:

$$\min \begin{bmatrix} r_1(i) & r_2(i) & \cdots & r_M(i) \end{bmatrix}$$

And find the optimal order $\bar{j} \in (1, M)$ of the parameters vector $\boldsymbol{\vartheta}_{\bar{j},i}$ that minimizes the error between $\hat{\omega}(i)$ and its estimation $\boldsymbol{\psi}_{\bar{j},i}\boldsymbol{\vartheta}_{\bar{j},i}$, where:

$$\boldsymbol{\psi}_{\bar{j},i}\boldsymbol{\vartheta}_{\bar{j},i} = \begin{bmatrix} p(i-1) & p(i-2) & \cdots & p(i-\bar{j}) \end{bmatrix} \begin{bmatrix} C_1(i) \\ C_2(i) \\ \vdots \\ C_{\bar{j}}(i) \end{bmatrix} \tag{3-16}$$

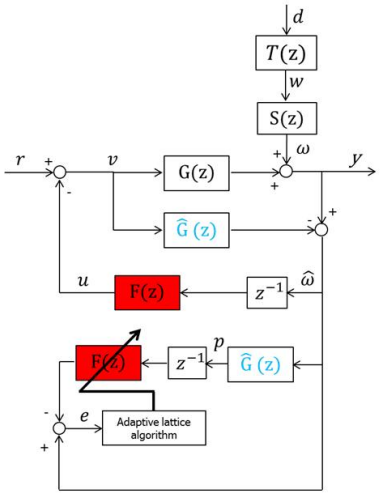


Figure 3-2: adaptive control with lattice filter

3-2-1 Lattice filter recursive formula

The recursive formula to update the *a posteriori* error from $r_M(i)$ to $r_{M+1}(i)$ is:

$$\boxed{r_{M+1}(i) = r_M(i) - \kappa_M(i) b_M(i)}$$

In this section it will be explained what the terms $\kappa_M(i)$ and $b_M(i)$ are.

First of all, we need to consider these several equivalent partitions of the observation matrix $H_{M+1,i}$:

$$\begin{aligned} H_{M+1,i} = \begin{bmatrix} \boldsymbol{\psi}_{M+1,1} \\ \boldsymbol{\psi}_{M+1,2} \\ \vdots \\ \boldsymbol{\psi}_{M+1,i} \end{bmatrix} &= \begin{bmatrix} p(0) & 0 & \cdots & 0 \\ p(1) & p(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p(i-1) & p(i-2) & \cdots & p(i-M-1) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{x}_{0,i} & \boldsymbol{x}_{1,i} & \cdots & \boldsymbol{x}_{M,i} \end{bmatrix} \\ &= \begin{bmatrix} H_{M,i} & \boldsymbol{x}_{M,i} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{x}_{0,i} & \bar{H}_{M,i} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{x}_{0,i} & \bar{H}_{M-1,i} & \boldsymbol{x}_{M,i} \end{bmatrix} \end{aligned}$$

We can now define:

- $\boldsymbol{b}_{M,i} = \boldsymbol{x}_{M,i} - H_{M,i}\boldsymbol{\vartheta}^b_{M,i} = a\ posteriori$ backward prediction error vector in estimating $\boldsymbol{x}_{M,i}$ with $H_{M,i}$

where $\boldsymbol{\vartheta}^b_{M,i}$ is the solution of this LS problem:

$$\min_{\vartheta^b_M} \left[\lambda^{i+1}\boldsymbol{\vartheta}^{bT}_M \Pi_M \boldsymbol{\vartheta}^b_M + (\boldsymbol{x}_{M,i} - H_{M,i}\boldsymbol{\vartheta}^b_M)^T \Lambda_i (\boldsymbol{x}_{M,i} - H_{M,i}\boldsymbol{\vartheta}^b_M)\right] \tag{3-17}$$

and $b_M(i)$ is the last entries of the vector $\boldsymbol{b}_{M,i}$, it means that

$$b_M(i) = p(i-M-1) - \begin{bmatrix} p(i-1) & p(i-2) & \cdots & p(i-M) \end{bmatrix} \boldsymbol{\vartheta}^b_{M,i}$$

and we can calculate it with a recursive formula:

$$\boxed{b_{M+1}(i) = b_M(i-1) - \kappa^b_M(i) f_M(i)}$$

- $\boldsymbol{f}_{M,i} = \boldsymbol{x}_{0,i} - \bar{H}_{M,i}\boldsymbol{\vartheta}^f_{M,i}$ = *a posteriori* forward prediction error vector in estimating $\boldsymbol{x}_{0,i}$ with $\bar{H}_{M,i}$

  where $\boldsymbol{\vartheta}^f_{M,i}$ is the solution of this LS problem:

$$\min_{\vartheta^f_M} \left[\lambda^{i}\boldsymbol{\vartheta}^{fT}_M \Pi_M \boldsymbol{\vartheta}^f_M + (\boldsymbol{x}_{M,i} - H_{M,i}\boldsymbol{\vartheta}^f_M)^T \Lambda_i (\boldsymbol{x}_{M,i} - H_{M,i}\boldsymbol{\vartheta}^f_M)\right] \tag{3-18}$$

  and $f_M(i)$ is the last entries of the vector $\boldsymbol{f}_{M,i}$, it means that

$$f_M(i) = p(i-1) - \begin{bmatrix} p(i-2) & p(i-3) & \cdots & p(i-M-1) \end{bmatrix} \boldsymbol{\vartheta}^f_{M,i}$$

  and we can calculate it with a recursive formula:

$$\boxed{f_{M+1}(i) = f_M(i) - \kappa^f_M(i) b_M(i-1)}$$

- $\kappa_M(i), \kappa^b_M(i), \kappa^f_M(i)$ are called **reflection coefficients** and are defined as follows:

$$\boxed{\kappa_M(i) = \frac{\rho^T_M(i)}{\zeta^b_M(i)} \qquad \kappa^b_M(i) = \frac{\delta_M(i)}{\zeta^f_M(i)} \qquad \kappa^f_M(i) = \frac{\delta^T_M(i)}{\zeta^b_M(i-1)}}$$

- $\zeta^b_M(i), \zeta^f_M(i)$ are (modified) **cost variables**, and have both a recursive formula to time-updating:

$$\boxed{\zeta^b_M(i) = \lambda\zeta^b_M(i-1) + \frac{|b_M(i)|^2}{\gamma_M(i)} \qquad \zeta^f_M(i) = \lambda\zeta^f_M(i-1) + \frac{|f_M(i)|^2}{\gamma_M(i-1)}}$$

- $\gamma_M(i)$ is called **conversion factor** and has a recursive formula to order-updating:

$$\boxed{\gamma_{M+1}(i) = \gamma_M(i) - \frac{|b_M(i)|^2}{\zeta_M^b(i)}}$$

- $\rho_M(i), \delta_M(i)$ have a recursive formula to time-updating:

$$\boxed{\rho_M(i) = \lambda\rho_M(i-1) + \frac{r_M^T(i)b_M(i)}{\gamma_M(i)} \qquad \delta_M(i) = \lambda\delta_M(i-1) + \frac{f_M^T(i)b_M(i-1)}{\gamma_M(i-1)}}$$

Now we can write the complete algorithm with the initialization:

1. from $m = 0$ to $m = M - 1$, set:
   $\delta_m(-1) = \rho_m(-1) = 0$;
   $\gamma_m(-1) = 1$;
   $b_m(-1) = 0$;
   $\zeta_m^f(-1) = \eta^{-1}\lambda^{-2}$;
   $\zeta_m^b(-1) = \eta^{-1}\lambda^{-m-2}$;
2. for $i \geq 0$, repeat:
   - set:
     $\gamma_0(i) = 1$;
     $b_0(i) = f_0(i) = p(i-1)$;
     $r_0(i) = \hat{\omega}(i)$;
   - from $m = 0$ to $m = M - 1$, repeat:
     $\zeta_m^f(i) = \lambda\zeta_m^f(i-1) + \frac{|f_m(i)|^2}{\gamma_m(i-1)}$
     $\zeta_m^b(i) = \lambda\zeta_m^b(i-1) + \frac{|b_m(i)|^2}{\gamma_m(i)}$
     $\delta_m(i) = \lambda\delta_m(i-1) + \frac{f_m^T(i)b_m(i-1)}{\gamma_m(i-1)}$
     $\rho_m(i) = \lambda\rho_m(i-1) + \frac{r_m^T(i)b_m(i)}{\gamma_m(i)}$
     $\gamma_{m+1}(i) = \gamma_m(i) - \frac{|b_m(i)|^2}{\zeta_m^b(i)}$
     $\kappa_m(i) = \frac{\rho_m^T(i)}{\zeta_m^b(i)}$
     $\kappa_m^b(i) = \frac{\delta_m(i)}{\zeta_m^f(i)}$
     $\kappa_m^f(i) = \frac{\delta_m^T(i)}{\zeta_m^b(i-1)}$
     $b_{m+1}(i) = b_m(i-1) - \kappa_m^b(i)f_m(i)$
     $f_{m+1}(i) = f_m(i) - \kappa_m^f(i)b_m(i-1)$
     $r_{m+1}(i) = r_m(i) - \kappa_m(i)b_m(i)$

In Figure 3-3 is shown the block diagram of the *a posteriori*-based lattice filter algorithm.

The lattice filter algorithm has 2 parameters that can be tuned by the user:

- λ, that is the forgetting factor ($0 \ll \lambda \leq 1$) and it has the same function as in RLS: if we want to track a time-varying signal, we have to set $\lambda < 1$.
- η, that appears in the equation of the regularization matrix Π_M (3-11): it is a positive scalar (usually large) and it influences transients in learning stage.

To know more details about *a posteriori*-based lattice filter, you can refer to [14].

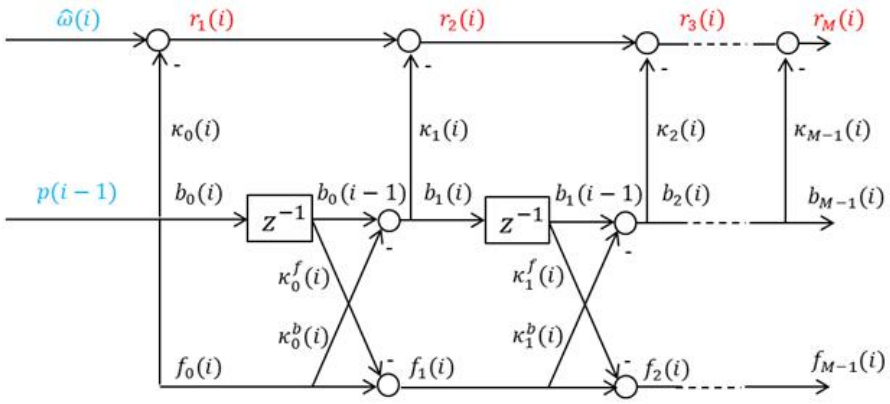


Figure 3-3: the *a posteriori*-based lattice filter

3-2-2 How to calculate parameters vector at time i

Once we have found the order $\bar{j} \in (1, M)$ of the AR model that minimizes the error signal e at time i, we need to calculate the parameters vector $\boldsymbol{\vartheta}_{\bar{j},i}$ at time step i.

Starting from the initial conditions $\boldsymbol{\vartheta}_{0,i} = 0$, $\boldsymbol{\vartheta}^b_{0,i} = 0$ and up to order $\bar{j}$ we have that [14]:

$$\boldsymbol{\vartheta}_{\bar{j},i} = \begin{bmatrix} \boldsymbol{\vartheta}_{\bar{j}-1,i} \\ 0 \end{bmatrix} + \kappa_{\bar{j}-1}(i) \begin{bmatrix} -\boldsymbol{\vartheta}^b_{\bar{j}-1,i} \\ 1 \end{bmatrix} \qquad ,\bar{j} \geq 1 \tag{3-19}$$

That can be also written [14] as:

$$\boldsymbol{\vartheta}_{\bar{j},i} = \sum_{l=0}^{\bar{j}-1} \kappa_l(i) \begin{bmatrix} -\boldsymbol{\vartheta}^b_{l,i} \\ 1 \end{bmatrix} \qquad ,\bar{j} \geq 1 \tag{3-20}$$

Hence, we have to calculate the parameters vector $\boldsymbol{\vartheta}_{\bar{j},i}$ of order $\bar{j} \in (1, M)$, where $\bar{j}$ is the order found optimal by the *a posteriori*-based lattice filter. To do that, we need to know:

- the reflection coefficients $\kappa_l(i)$ from $l = 0$ up to $l = \bar{j} - 1$.
  We already know these coefficients because we need them to evaluate the *a posteriori* error $r_M(i)$ in the lattice algorithm (actually, to order update $b_M(i)$ and $f_M(i)$).
- the vector $\boldsymbol{\vartheta}^b_{l,i}$ from $l = 1$ to $l = \bar{j} - 1$ (because $\boldsymbol{\vartheta}^b_{0,i} = 0$) that is the solution of the LS problem in (3-17).
  We don't know this vector yet.

We could evaluate $\boldsymbol{\vartheta}^b_{l,i}$ by using an exponentially-weighted RLS algorithm, but we would lose the advantage in using lattice filter instead of RLS. Notice that in that case it would be computationally heavier than RLS because we would need to solve a LS problem for every order l up to $l = \bar{j} - 1$. We need to find a more computationally efficient way.

We can observe that at time i the parameters vector that solves the LS problem is the vector $\boldsymbol{\vartheta}_{l,i}^b$ that appears in the following formula:

$$b_l(i) = p(i-l-1) - \begin{bmatrix} p(i-1) & p(i-2) & \cdots & p(i-l) \end{bmatrix} \boldsymbol{\vartheta}_{l,i}^b \tag{3-21}$$

It suggests that, if we manage to write the signal $b_l(i)$ in an explicit form that reveals the coefficients of the parameters vector $\boldsymbol{\vartheta}_{l,i}^b$, we have an alternative way to find the solution to the problem (3-17).

We also need to remind the vector $\boldsymbol{\vartheta}_{l,i}^f$, that is the solution of the LS problem in (3-18) and that appears in the following formula:

$$f_l(i) = p(i-1) - \begin{bmatrix} p(i-2) & p(i-3) & \cdots & p(i-l-1) \end{bmatrix} \boldsymbol{\vartheta}_{l,i}^f \tag{3-22}$$

Remember that the recursive expressions to order-update the *a posteriori* backward and forward prediction errors are:

$$b_l(i) = b_{l-1}(i-1) - \kappa_{l-1}^b(i) f_{l-1}(i)$$

$$f_l(i) = f_{l-1}(i) - \kappa_{l-1}^f(i) b_{l-1}(i-1)$$

In our application, we know that the initial backward and forward prediction errors conditions are (section 3-2-1):

$$b_0(i) = f_0(i) = p(i-1)$$

Let's consider first $\underline{l=1}$. We have:

$$b_1(i) = b_0(i-1) - \kappa_0^b(i) f_0(i) = p(i-2) - \kappa_0^b(i) p(i-1) \tag{3-23}$$

that is identical to (3-21) if $l = 1$ and here we have the value of the parameters vector (that is a scalar if $l = 1$):

$$\boldsymbol{\vartheta}_{1,i}^b = \kappa_0^b(i) \tag{3-24}$$

We also need to calculate $f_1(i)$ to solve the next equations. We have:

$$f_1(i) = f_0(i) - \kappa_0^f(i) b_0(i-1) = p(i-1) - \kappa_0^f(i) p(i-2) \tag{3-25}$$

that is identical to (3-22) so that we can say:

$$\boldsymbol{\vartheta}_{1,i}^f = \kappa_0^f(i) \tag{3-26}$$

Now, we can try to do the same for $\underline{l=2}$. We have that:

$$\begin{aligned} b_2(i) = b_1(i-1) - \kappa_1^b(i) f_1(i) &= p(i-3) - \kappa_0^b(i-1)p(i-2) - \kappa_1^b(i)\left(p(i-1) - \kappa_0^f(i)p(i-2)\right) \\ &= p(i-3) - p(i-2)\left(\kappa_0^b(i-1) - \kappa_1^b(i)\kappa_0^f(i)\right) - p(i-1)\kappa_1^b(i) \\ &= p(i-3) - \begin{bmatrix} p(i-1) & p(i-2) \end{bmatrix} \begin{bmatrix} \kappa_1^b(i) \\ \kappa_0^b(i-1) - \kappa_1^b(i)\kappa_0^f(i) \end{bmatrix} \end{aligned}$$

where we can recognize the same structure of (3-21) if $l = 2$. It means that we can find the parameters vector with this simple expression:

$$\boldsymbol{\vartheta}_{2,i}^b = \begin{bmatrix} \kappa_1^b(i) \\ \kappa_0^b(i-1) - \kappa_1^b(i)\kappa_0^f(i) \end{bmatrix} \tag{3-27}$$

In the same way we can find the expression of the vector $\boldsymbol{\vartheta}_{2,i}^f$:

$$\boldsymbol{\vartheta}_{2,i}^f = \begin{bmatrix} \kappa_0^f(i) - \kappa_1^f(i)\kappa_0^b(i-1) \\ \kappa_1^f(i) \end{bmatrix} \tag{3-28}$$

We also show the expressions of the vectors if $\underline{l = 3}$ (the procedure is similar to the previous ones):

$$\boldsymbol{\vartheta}_{3,i}^b = \begin{bmatrix} \kappa_2^b(i) \\ \kappa_1^b(i-1) - \kappa_2^b(i)\kappa_0^f(i) + \kappa_2^b(i)\kappa_1^f(i)\kappa_0^b(i-1) \\ \kappa_0^b(i-2) - \kappa_1^b(i-1)\kappa_0^f(i-1) - \kappa_2^b(i)\kappa_1^f(i) \end{bmatrix} \tag{3-29}$$

$$\boldsymbol{\vartheta}_{3,i}^f = \begin{bmatrix} \kappa_0^f(i) - \kappa_1^f(i)\kappa_0^b(i-1) - \kappa_2^f(i)\kappa_1^b(i-1) \\ \kappa_1^f(i) - \kappa_2^f(i)\kappa_0^b(i-2) + \kappa_2^f(i)\kappa_1^b(i-1)\kappa_0^f(i-1) \\ \kappa_2^f(i) \end{bmatrix} \tag{3-30}$$

We could continue to find every $\boldsymbol{\vartheta}_{l,i}^b$ (and $\boldsymbol{\vartheta}_{l,i}^f$) up to $l = \bar{j} - 1$: we will see that these vectors are function of the reflection coefficients $\kappa_l^b(t)$ and $\kappa_l^f(t)$ from $l = 0$ to $l = \bar{j} - 2$ and from $t = i - \bar{j} + 2$ to $t = i$, that are already known by the computation of the Lattice Filter algorithm. It means that we can find the parameters vector $\boldsymbol{\vartheta}_{\bar{j},i}$ with the same computational cost we have to evaluate the optimal order $\bar{j}$, and it is $O(M)$, one order magnitude less than RLS.

Actually, we have two recursive equations for the parameters vectors $\boldsymbol{\vartheta}^b_{l,i}$ and $\boldsymbol{\vartheta}^f_{l,i}$ that allow the practical implementation [15].
Starting from the initial conditions $\boldsymbol{\vartheta}^b_{0,i-1} = 0$, $\boldsymbol{\vartheta}^f_{0,i} = 0$, we have:

$$\boldsymbol{\vartheta}^b_{l,i} = \begin{bmatrix} 0 \\ \boldsymbol{\vartheta}^b_{l-1,i-1} \end{bmatrix} - \kappa^b_{l-1}(i) \begin{bmatrix} -1 \\ \boldsymbol{\vartheta}^f_{l-1,i} \end{bmatrix} \qquad , l \geq 1 \tag{3-31}$$

$$\boldsymbol{\vartheta}^f_{l,i} = \begin{bmatrix} \boldsymbol{\vartheta}^f_{l-1,i} \\ 0 \end{bmatrix} - \kappa^f_{l-1}(i) \begin{bmatrix} \boldsymbol{\vartheta}^b_{l-1,i-1} \\ -1 \end{bmatrix} \qquad , l \geq 1 \tag{3-32}$$

3-3 Generation of the control signal

Once we have found the optimal parameters vector $\boldsymbol{\vartheta}_{\bar{j},i}$ of order $\bar{j} \in (1, M)$ at time i, we can finally generate the control law $u(i)$ that is the output of the lattice filter $F(z)$:

$$u(i) = \begin{bmatrix} \hat{\omega}(i-1) & \hat{\omega}(i-2) & \cdots & \hat{\omega}(i-\bar{j}) \end{bmatrix} \boldsymbol{\vartheta}_{\bar{j},i} \tag{3-33}$$

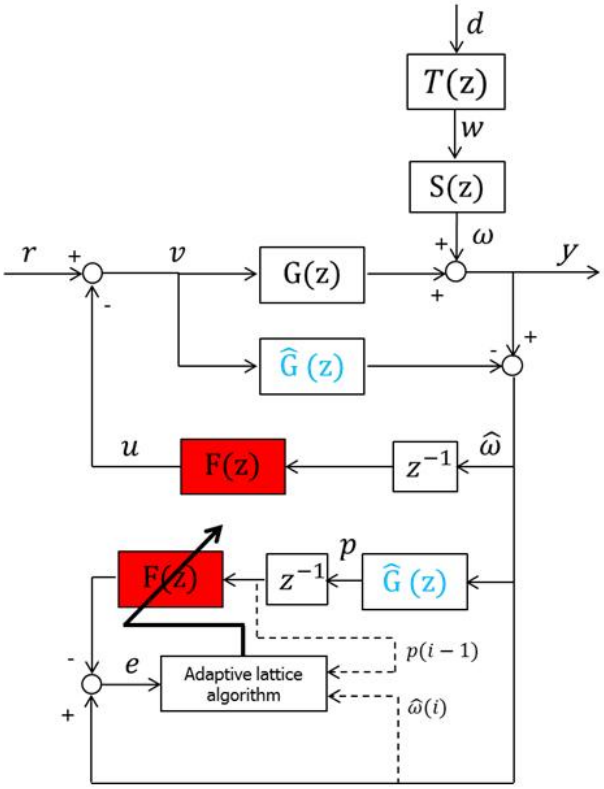


Figure 3-4: adaptive control with lattice filter

With this control variable, if $\hat{G}(z) = G(z)$ and the lattice algorithm works perfectly, we can fully reject the turbulence wavefront phase distortion ω. In real situations, $\hat{G}(z) \neq G(z)$ and we can't achieve a perfect reconstruction of the turbulence at time-step i, but this adaptive control allows us to reduce the wavefront phase distortion and to have an almost flat residual phase y.

3-4 Parameters vector convergence: Lattice Filter VS RLS

To verify that lattice filters work well, we have to do a convergence analysis and compare it with the one of RLS.

To do that, we consider several random fixed-order AR(M) processes. To make the simulation not too complicated, we chose M=4. Then, we have generated 12 different AR(4) processes like this:

$$y(k) = a_1^{(i)}y(k-1) + a_2^{(i)}y(k-2) + a_3^{(i)}y(k-3) + a_4^{(i)}y(k-4) + \xi(k) \tag{3-34}$$

where ξ is a white-Gaussian-noise of mean 0 and variance 1 ($\xi \approx WGN(0,1)$), and $i \in [1,12]$ represents the AR model we are considering. In Tabel 3-1 are listed the parameters $a_1^{(i)}, a_2^{(i)}, a_3^{(i)}, a_4^{(i)}$ for all the AR processes we used:

index i	$a_1^{(i)}$	$a_2^{(i)}$	$a_3^{(i)}$	$a_4^{(i)}$
1	0.2	0.18	0.1	0.3
2	0.4	0.25	0.04	0.2
3	0.7	0.12	0.08	0.02
4	0.01	0.6	0.2	0.17
5	0.1	0.2	0.3	0.35
6	0.35	0.3	0.2	0.1
7	0.05	0.01	0.08	0.06
8	-0.3	0.09	-0.17	0.2
9	-0.2	0.4	-0.08	0.3
10	0.2	-0.3	-0.15	-0.23
11	-0.1	-0.2	-0.3	-0.4
12	0.8	0.02	0.04	0.08

Table 3-1: list of the different AR parameters

The parameters vector is defined as: $\boldsymbol{\theta}^{(i)} = [a_1^{(i)} \quad a_2^{(i)} \quad a_3^{(i)} \quad a_4^{(i)}]^T$.

Now, if we choose a RLS order 4 algorithm or a lattice order 4 algorithm to estimate each of the AR processes described by (3-34), we have the following prediction model:

$$\hat{y}(k) = \hat{a}_1^{(i)}y(k-1) + \hat{a}_2^{(i)}y(k-2) + \hat{a}_3^{(i)}y(k-3) + \hat{a}_4^{(i)}y(k-4) \tag{3-35}$$

If we consider a simulation time of N steps and we use RLS, for the convergence property of RLS we know that

$$if \quad N \to \infty, \quad then \quad \hat{a}_j^{(i)} \to a_j^{(i)} \tag{3-36}$$

where $j \in [1,4]$.

We have done a simulation with N=15000 and we have tried to estimate all the AR processes in Table 3-1 with an order 4 lattice filter algorithm with $\lambda = 0.9999$ and $\eta = 1000$.

For instance, the estimation of the first AR process ($i = 1$) produces the following trend of the AR parameters (Figure 3-5):

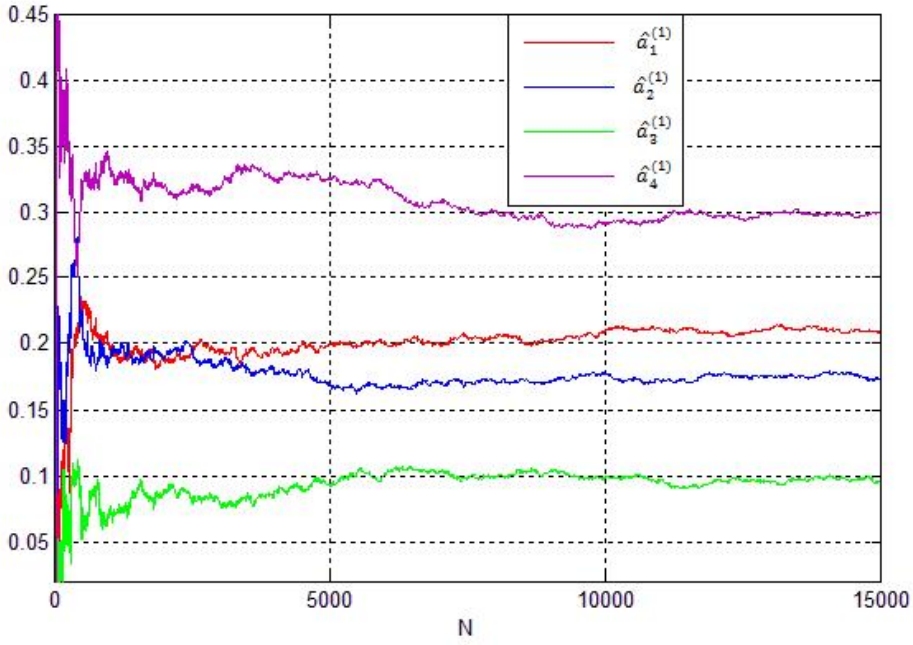


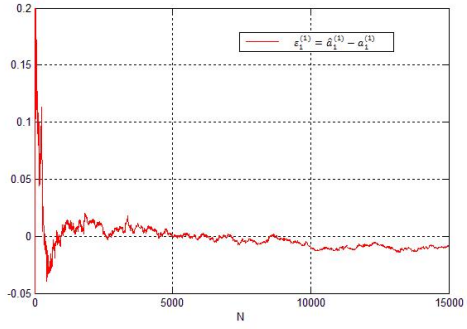
Figure 3-5: parameters trend in estimating the first AR (i=1) with order 4 lattice filter

We can observe that there is an initial transient in which the value of the parameters oscillates around the real value $a_j^{(1)}$, and then the parameters converge to the real value.

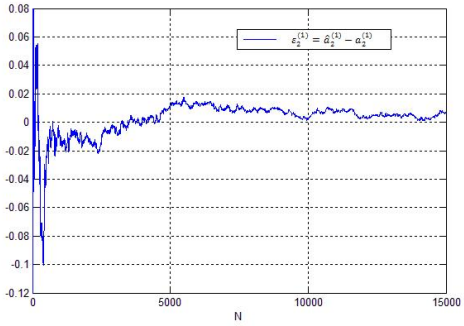
Now, we can do (for each of the i processes) the following operation to highlight the error trend in estimating the AR parameters:

$$\varepsilon_j^{(i)} = a_j^{(i)} - \hat{a}_j^{(i)} \tag{3-37}$$

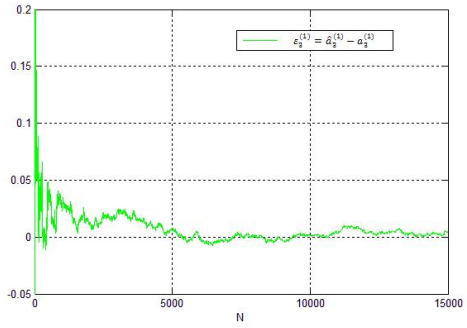
where $i \in [1, 12]$, $j \in [1, 4]$.



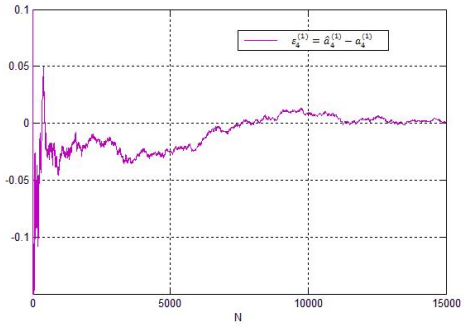
(a) error trend in estimating $a_1^{(1)}$



(b) error trend in estimating $a_2^{(1)}$



(c) error trend in estimating $a_3^{(1)}$



(d) error trend in estimating $a_4^{(1)}$

Figure 3-6: error trend in estimating AR parameter $a_j^{(1)}$

For example, if $i = 1$, we have $\varepsilon_j^{(1)} = a_j^{(1)} - \hat{a}_j^{(1)}$, and in Figure 3-6 is shown what we obtain for each j.

Then, for each time-step N, we can evaluate the following mean error:

$$\bar{\varepsilon}_j(N) = \frac{1}{12}\sum_{i=1}^{12}\varepsilon_j^{(i)}(N) = \frac{1}{12}\left(\varepsilon_j^{(1)}(N) + \varepsilon_j^{(2)}(N) + \cdots + \varepsilon_j^{(12)}(N)\right) \quad , j \in [1,4] \tag{3-38}$$

So that we will have 4 different mean errors, one for each j parameter of the AR process:

- $\bar{\varepsilon}_1(N) = \frac{1}{12}\sum_{i=1}^{12}\varepsilon_1^{(i)}(N)$;
- $\bar{\varepsilon}_2(N) = \frac{1}{12}\sum_{i=1}^{12}\varepsilon_2^{(i)}(N)$;
- $\bar{\varepsilon}_3(N) = \frac{1}{12}\sum_{i=1}^{12}\varepsilon_3^{(i)}(N)$;
- $\bar{\varepsilon}_4(N) = \frac{1}{12}\sum_{i=1}^{12}\varepsilon_4^{(i)}(N)$;

with $N = 1, 2, \cdots, 15000$.

We show these values in the following figures (Figures 3-7/3-10), where the yellow lines in each figure are the lower and upper bound of the errors (representing the variance of the error). It means that they represent, at each time-step N, the maximum and the minimum value of $\left[\varepsilon_j^{(1)}, \varepsilon_j^{(2)}, \cdots, \varepsilon_j^{(12)}\right]$,with $j \in [1, 4]$:

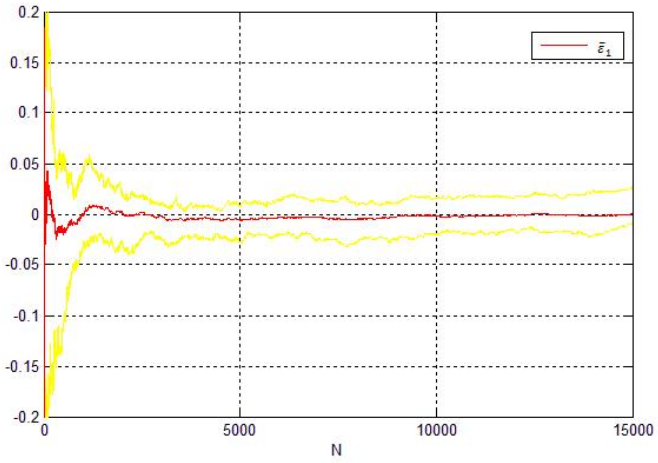


Figure 3-7: trend of $\bar{\varepsilon}_1$ with his upper and lower bound (lattice)

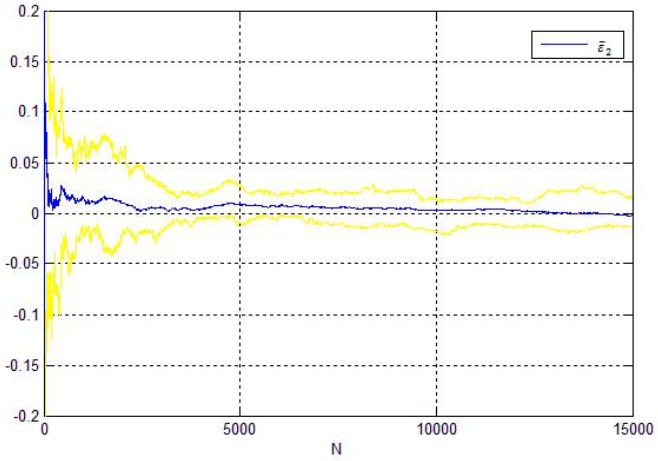


Figure 3-8: trend of $\bar{\varepsilon}_2$ with his upper and lower bound (lattice)

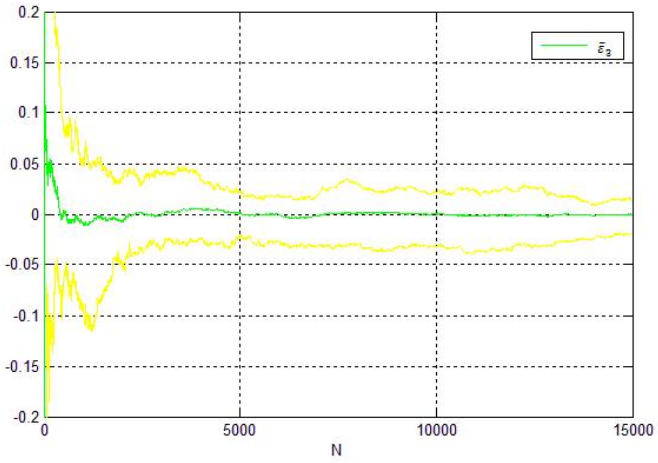


Figure 3-9: trend of $\bar{\varepsilon}_3$ with his upper and lower bound (lattice)

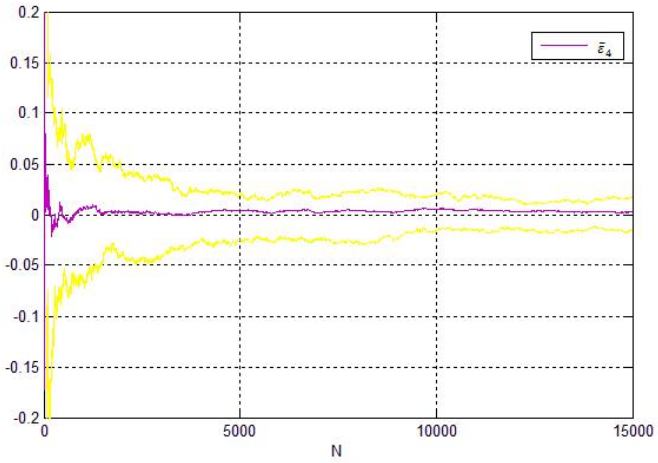


Figure 3-10: trend of $\bar{\varepsilon}_4$ with his upper and lower bound (lattice)

From these figures we can conclude that also in lattice filter we have that:

$$if \quad N \to \infty, \quad then \quad \hat{a}_j^{(i)} \to a_j^{(i)}. \tag{3-39}$$

Notice that at the beginning we have a bigger variance, that reduces itself with the increasing of the time N.

To make a comparison, we have done the same thing for RLS (we have tried to estimate the 12 AR processes in Table 3-1 with a RLS algorithm, and then we have evaluated the mean error $\bar{\varepsilon}_j$, $j \in [1, 4]$), and these are the results:

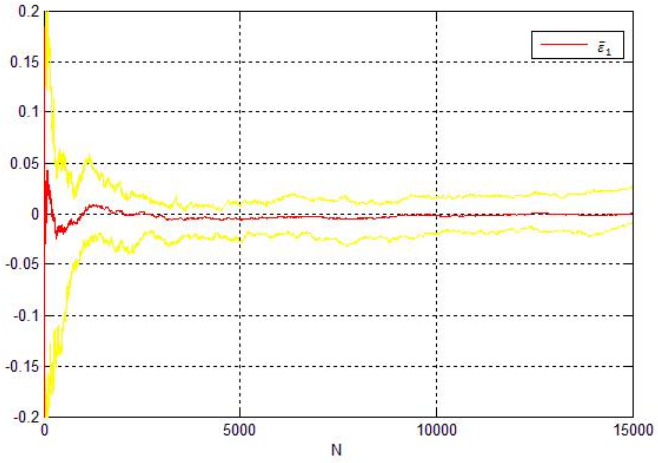


Figure 3-11: trend of $\bar{\varepsilon}_1$ with his upper and lower bound (RLS)

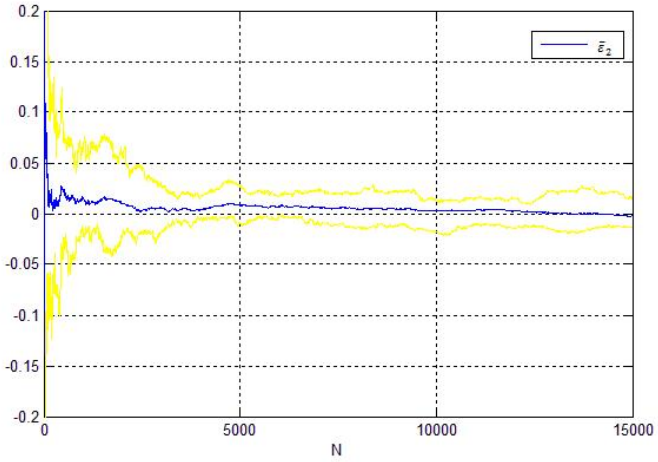


Figure 3-12: trend of $\bar{\varepsilon}_2$ with his upper and lower bound (RLS)

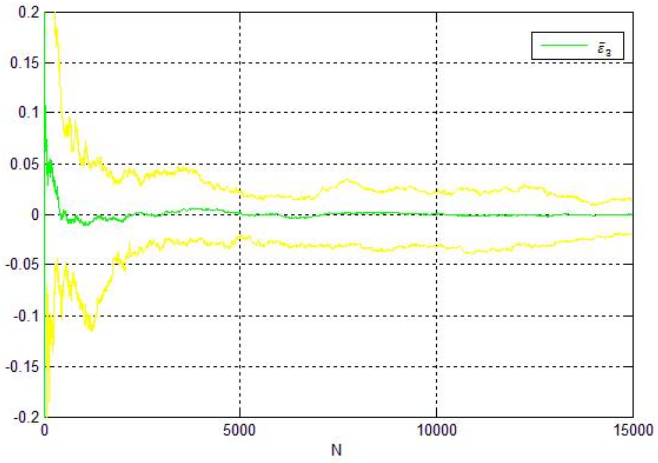


Figure 3-13: trend of $\bar{\varepsilon}_3$ with his upper and lower bound (RLS)

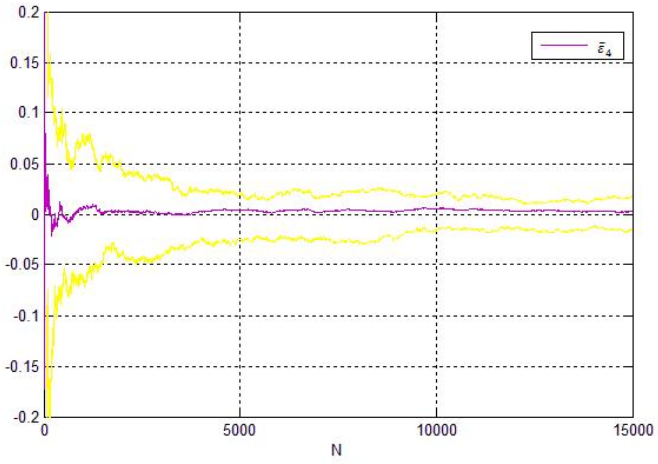


Figure 3-14: trend of $\bar{\varepsilon}_4$ with his upper and lower bound (RLS)

We can say that lattice filter and RLS have almost the same convergence properties for the AR processes we have considered in this evaluation. Actually, lattice filters have the same performance of RLS only if they are implemented with infinite-precision arithmetic operations. In finite-precision implementation each algorithm will perform differently [15]. Because RLS and lattice minimize the same cost function (see Section 3.1-3.2), we already expected that they would had the same performance. But lattice filter has a limitation: it could lead to divergence in some particular situation. To know more about this topic you can see [16].

Notice that to realize the convergence analysis we used the **forced** lattice filter (i.e. we have forced the order to be fixed at 4 during the all simulation and not to change from 1 to 4 as it should be in the lattice algorithm). In fact, the normal lattice filter changes order at every time step: it means that it changes the parameters vector dimension at every time step. For this reason, the estimated parameters $\hat{a}_j^{(i)}$ do not converge but continue jumping between

some values close by the convergence value $a_i^{(i)}$.

For example, if we consider the first AR process in Table 3-1 and we try to estimate it with a normal order 4 lattice filter, we have this trend of the parameter $\hat{a}_1^{(1)}$ (Figure 3-15):

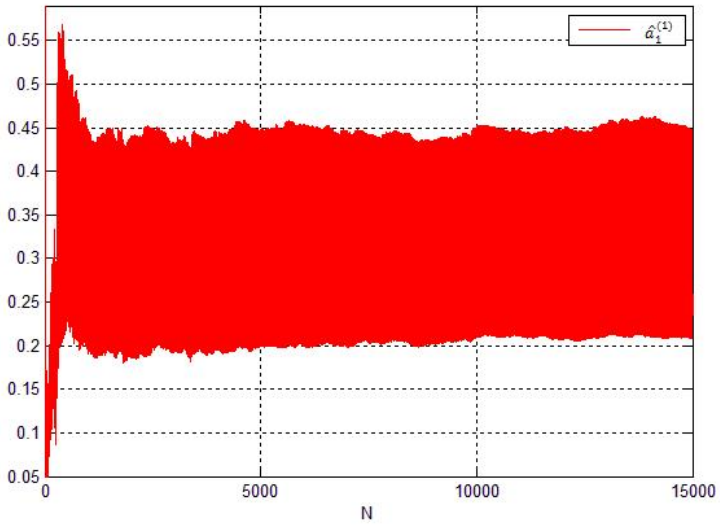


Figure 3-15: trend of $\hat{a}_1^{(1)}$ using normal lattice filter

Even if we have not parameters convergence in normal lattice filter, we manage to obtain a great process reconstruction, even better than the one produced by the forced lattice filter. In Figure 5-10 we have the first AR process in Table 3-1 (blue), the normal lattice estimation (green) and the forced lattice estimation (red) in a short time-horizon:

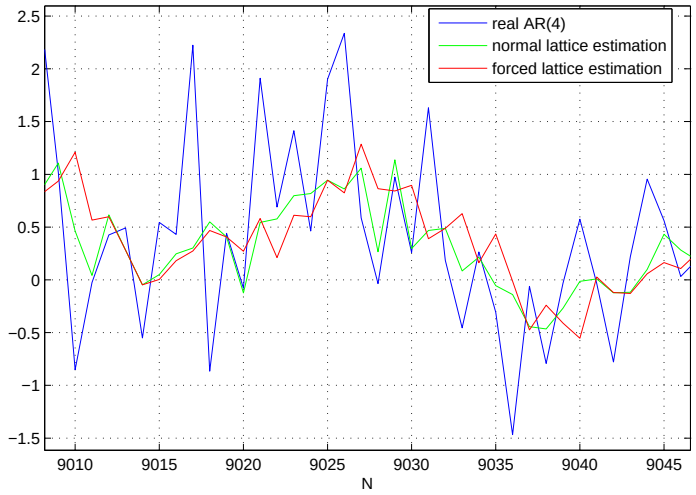


Figure 3-16: estimation of AR process with normal and forced lattice filters

We can observe that the normal lattice filter works better than the forced one in this particular case. To proof that, we have evaluated the root mean square (RMS) value between:

- the real AR(4) $y(k)$ and the normal lattice estimation $\hat{y}_{L,N}(k)$:

$$RMS_{L,N}^{(1)} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y(k) - \hat{y}_{L,N}(k))^2} = 0.8303 \tag{3-40}$$

- the real AR(4) $y(k)$ and the forced lattice estimation $\hat{y}_{L,F}(k)$:

$$RMS_{L,F}^{(1)} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y(k) - \hat{y}_{L,F}(k))^2} = 0.9853 \tag{3-41}$$

Then, we have evaluated these 2 RMS values for all the AR processes in Table 3-1, where $RMS^{(i)}$ is the RMS value related to the AR process with index i in Table 3-1:

RMS values - normal VS forced		
index i	$RMS_{L,N}^{(i)}$	$RMS_{L,F}^{(i)}$
1	0.8303	0.9853
2	0.8516	0.9972
3	0.9174	0.9915
4	0.8518	1.0064
5	0.8484	0.9871
6	0.8546	0.9984
7	0.9578	1.0079
8	0.8792	1.0091
9	0.9321	0.9998
10	0.8394	1.0029
11	0.8226	0.9932
12	0.9508	1.0087

Table 3-2: list of RMS values in normal and forced lattice filter in estimating AR(4) processes

From this table we can observe that normal lattice filter works always better than the forced one in our simulations. This cannot lead to any conclusion, we can only observe what happens in our specific situation.

The RMS value if we use RLS ($RMS_{RLS}^{(i)}$) is very close to the RMS of the forced lattice filter.

3-5 Order VS performance in RLS and in Lattice

We have seen that order 4 forced lattice filter and order 4 RLS have the same performance. Then, we have observed that (in our examples) normal lattice filter works better than forced lattice filter.

Now, we want to evaluate the performance of the algorithms increasing their order from 1 to 20. What we are going to evaluate is again the RMS value between the real AR(M) process and its estimation by RLS, normal lattice and forced lattice. To make this evaluation more general, we have considered 5 different AR(M) processes for each order M, and we have calculated the mean between their RMS values to generate the order M RMS value.

Assuming $k \in [1, 5]$ as the number of different AR realizations of the same order M, and $M = [1, 2, 3, 4, 5, 6, 8, 10, 20]$ as the order of the AR process, we can define:

- $RMS^M_{L,N,k}$ as the RMS value between the k-th realization of the process AR(M) and its estimation with normal lattice filter;
- $RMS^M_{L,F,k}$ as the RMS value between the k-th realization of the process AR(M) and its estimation with forced lattice filter;
- $RMS^M_{RLS,k}$ as the RMS value between the k-th realization of the process AR(M) and its estimation with RLS;

and then, we can define:

- $\bar{RMS}^M_{L,N}$ as the mean of the k RMS values evaluated with order M normal lattice;
- $\bar{RMS}^M_{L,F}$ as the mean of the k RMS values evaluated with order M forced lattice;
- $\bar{RMS}^M_{RLS}$ as the mean of the k RMS values evaluated with order M RLS;

In Table 3-3 we show the mean RMS values:

$\bar{RMS}$ values increasing the order			
index M	$R\bar{M}S^M_{L,N}$	$R\bar{M}S^M_{L,F}$	$R\bar{M}S^M_{RLS}$
1	0.9967	0.9967	0.9967
2	0.8929	1.0008	1.0008
3	0.8878	1.0024	1.0024
4	0.8851	0.9972	0.9972
5	0.9137	0.9993	0.9993
6	0.9100	0.9983	0.9983
8	0.8936	0.9968	0.9968
10	0.8909	0.9949	0.9949
20	0.8467	0.9888	0.9888

Table 3-3: list of RMS values increasing the order of the algorithms

In Figure 3-17 we plot the values in Table 3-3:

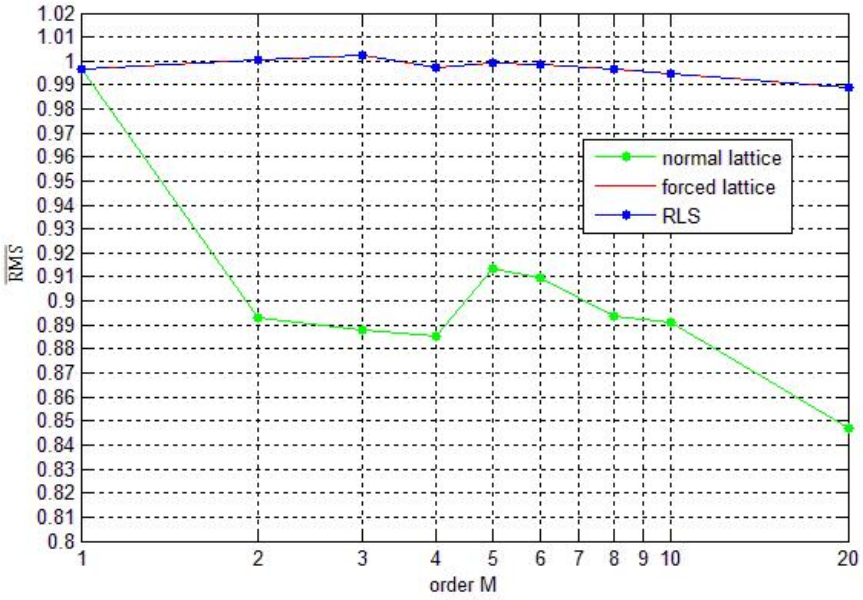


Figure 3-17: $R\bar{M}S$ values in Table 3-3

In this Figure 3-17 we can see once more that RLS and forced lattice filter have the same performance and that order 4 lattice filter is a good compromise between complexity and performance. Also we can observe once more that, in our simulations, the normal lattice works better than the forced one. The strange decreasing trend of the $R\bar{M}S$ value after order $M = 5$ is probably due to numerical errors.

Chapter 4

The MIMO system: channels decoupling

In the previous Chapters we have considered a SISO system with scalar disturbance and scalar output. This is the easiest situation because we only have one channel, but it is not the real situation. In this Chapter we will generalize to the MIMO situation and we will show a configuration that allows to decouple the channels of the system.

4-1 From SISO to MIMO

Let's consider again the classical control loop without the adaptive loop. For now, we neglect the WFS delay. In the real situation, the disturbance $w = w(\rho, t)$ is a surface (function of $\mathbb{R}^2$) and we have a MIMO system like in Figure 4-1, where w has been transformed in a vector for the simulation:

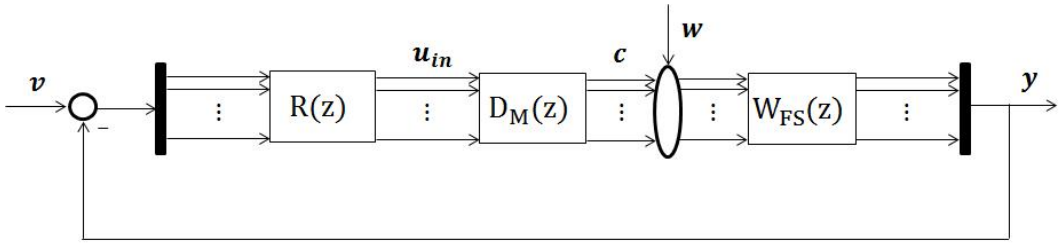


Figure 4-1: MIMO system

With MIMO system things become more difficult: the DM block is represented by a $P \times P$ matrix, where P is the number of channels. This DM matrix is not diagonal, it is a full matrix where every element is a TF, and the consequence is that every control command $[u_{in_1}, u_{in_2}, \cdots u_{in_P}] \in \boldsymbol{u_{in}}$ has an influence on every DM actuator command $[c_1, c_2, \cdots c_P] \in \boldsymbol{c}$ (Figure 4-2):

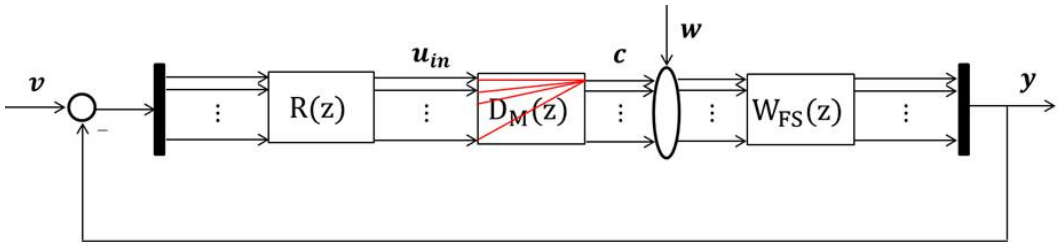


Figure 4-2: MIMO system - channels coupling

For example, if we consider $P = 2$, the the DM matrix is a 2×2 matrix:

$$D_M(z) = \begin{bmatrix} D_{M_{11}}(z) & D_{M_{12}}(z) \\ D_{M_{21}}(z) & D_{M_{22}}(z) \end{bmatrix} \tag{4-1}$$

where $D_{M_{i,j}}$ is a TF, with $(i, j) \in [1, 2]$.

We can say that the DM command vector $\boldsymbol{c}$ is given by:

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} D_{M_{11}}(z) & D_{M_{12}}(z) \\ D_{M_{21}}(z) & D_{M_{22}}(z) \end{bmatrix} \begin{bmatrix} u_{in_1} \\ u_{in_2} \end{bmatrix} \tag{4-2}$$

We can represent the system like in Figure 4-3, where we can well notice the channels coupling:

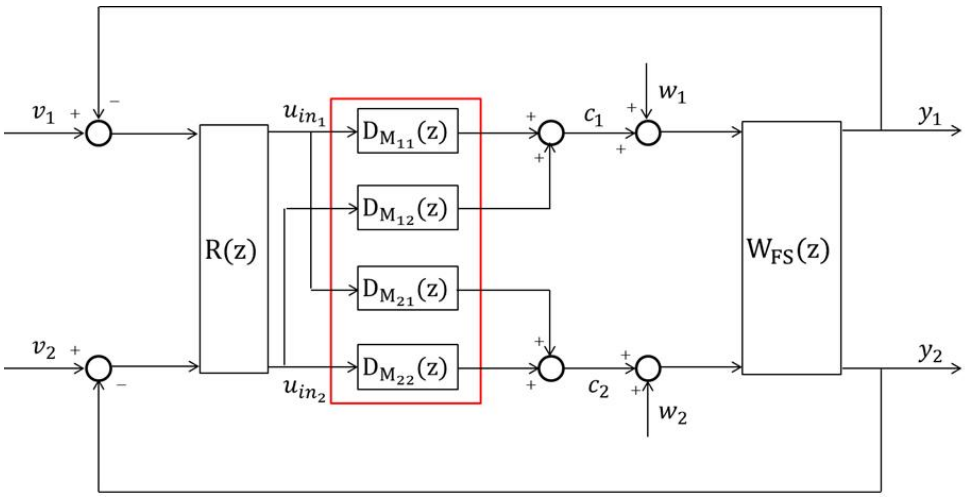


Figure 4-3: MIMO system with $P = 2$

4-2 MIMO channels decoupling

To decouple system's channels we can realize the following configuration of the classical (internal) control loop [12]:

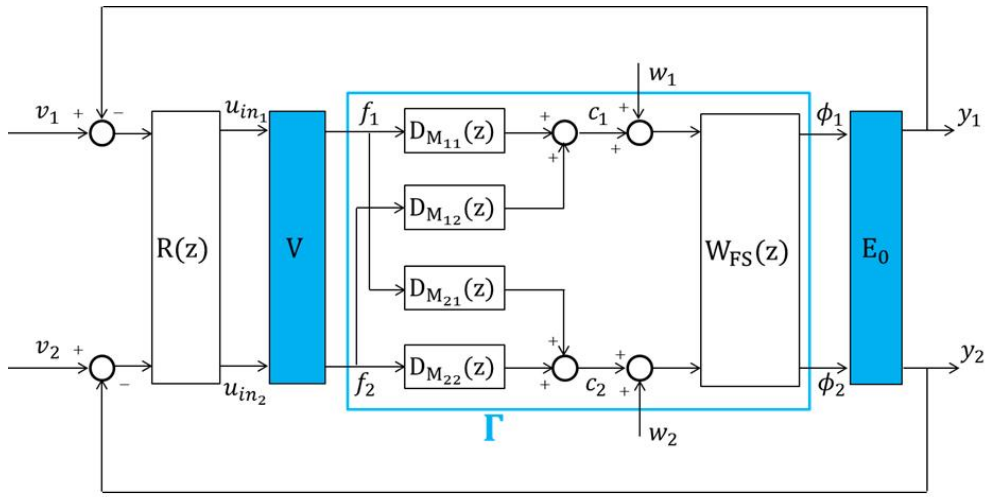


Figure 4-4: MIMO system decoupling configuration with $P = 2$

In the same way, we can extend this configuration to the generic $P \times P$ situation:

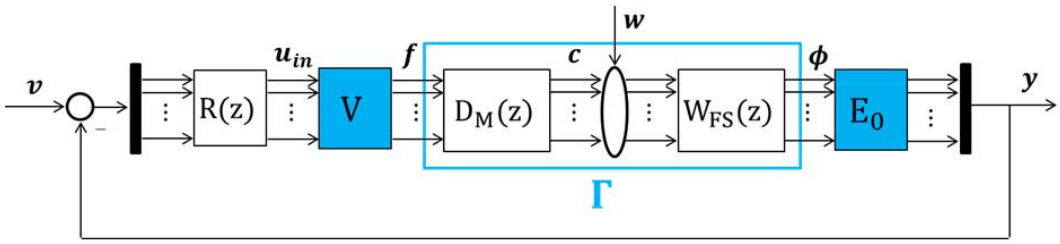


Figure 4-5: MIMO system decoupling configuration

Where:

- Γ is called the Poke Matrix and represents the mapping from $\boldsymbol{f}$ to $\boldsymbol{\phi}$. This poke matrix can be identified with an experiment on the system [4].
- The matrix V defines a parametrization of actuator space: the columns of V represent DM modes that are commanded independently by the control loop.

  To define this matrix, we can realize a singular value decomposition (SVD) of the Poke Matrix Γ:

  $$\Gamma = Q_\Gamma \Sigma V_\Gamma^T \tag{4-3}$$

  and to set $V = V_\Gamma$.
- The reconstructor matrix E_0 is chosen to satisfy

  $$E_0 \Gamma V = I \tag{4-4}$$

it means that E_0 is chosen as the pseudo-inverse of ΓV.

Proof. We have that:

$$\boldsymbol{y} = E_0 \Gamma V \boldsymbol{u_{in}} = E_0 (Q_\Gamma \Sigma V_\Gamma^T) V \boldsymbol{u_{in}}.$$

If we choose $V = V_\Gamma$, we have:

$$\boldsymbol{y} = E_0 (Q_\Gamma \Sigma V_\Gamma^T) V_\Gamma \boldsymbol{u_{in}}$$

But we know that, because of the SVD, $(V V_\Gamma^T) = (V_\Gamma^T V) = I$. It means that

$$\boldsymbol{y} = E_0 (Q_\Gamma \Sigma) \boldsymbol{u_{in}}$$

To decouple the channels, we want that $E_0 Q_\Gamma \Sigma = I$, so we need

$$E_0 = (Q_\Gamma \Sigma)^\dagger = (\Gamma V)^\dagger$$

□

Under this condition, the control channels represented by the components of the vector $\boldsymbol{u_{in}}$ are uncoupled. The control command represented by a single element of $\boldsymbol{u_{in}}$ affects only the error signal represented by the corresponding element of the error vector $\boldsymbol{y}$.

Now, we can augment the classical loop with the adaptive loop, but we need to redefine what we called $G(z)$ and $S(z)$ respectively in equation (2-7) and (2-8). Indeed the classical loop contains 2 more elements (V and E_0) and $R(z)$ is a diagonal matrix:
$R(z)$ =diag$(R_1(z)R_2(z)\cdots R_P(z))$, where P is the number of channels and $R_i(z)$ is the controller for the channel i. Furthermore $D_M(z)$ and $W_{FS}(z)$ are full matrices of TF.

Therefore, reminding us of the WFS delay, we can now redefine:

$$G(z) = \frac{E_0 W_{FS}(z) D_M(z) V R(z)}{1 + z^{-1} E_0 W_{FS}(z) D_M(z) V R(z)} \tag{4-5}$$

$$S(z) = \frac{E_0 W_{FS}(z)}{1 + z^{-1} E_0 W_{FS}(z) D_M(z) V R(z)} \tag{4-6}$$

and in Figure 4-6 we have the complete MIMO block diagram with channels decoupling.

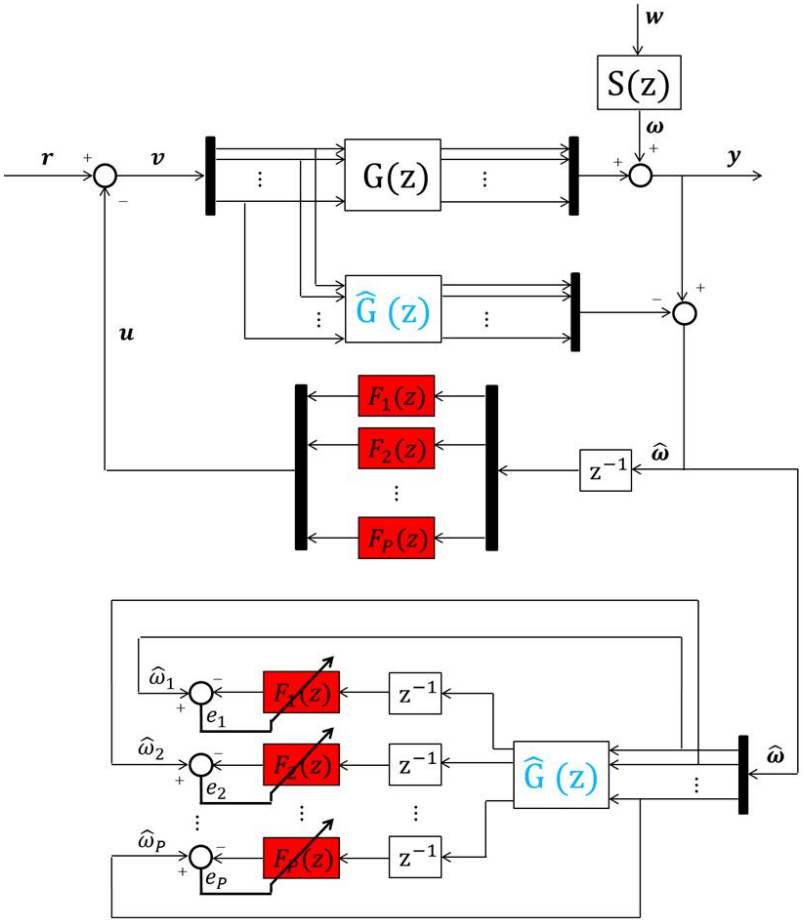


Figure 4-6: MIMO system with adaptive loop

Chapter 5

Simulations

In Chapter 4 we have described how we can decouple the MIMO system using the matrices V and E_0. Now, we have to describe how we can represent the DM and the WFS in the simulation to make it realistic and how we can realize a realistic turbulence for the simulation.

5-1 The simulation of the atmospheric turbulence

5-1-1 Stationary turbulence

In this section we explain very briefly how the turbulence can be generated. The goal of this section about turbulence is not to be exhaustive. An extensive treatment of this topic can be found in [17], [18], [19], [20].

The turbulence correlation matrix

We can start saying that the atmospheric turbulence w is a $N_x \times N_y$ grid at every time instant. In our situation we have $N_x = 14, N_y = 7$. The Figure 3-7 is an example of the 3D representation of the stationary turbulence wavefront phase.

As we told in Chapter 2, the atmospheric turbulence can be described by 4 parameters:

- L_0: is the *outer scale of turbulence*. This is the characteristic size of the large-scale turbulence inhomogeneities. It is expressed in meters $[m]$;
- r_0: is the *Fried parameter*. It is a measure of the optical quality of the atmosphere and it indicates the size of a telescope which can operate at the diffraction limit. It is expressed in meters $[m]$;
- v_x: is the wind velocity in x direction. It is expressed in $[m/s]$;
- v_y: is the wind velocity in y direction. It is expressed in $[m/s]$.

If we consider this parameters fixed, we are talking about stationary atmospheric turbulence. In Figure 5-1 we have these parameters: $L_0 = 2[m], r_0 = 0.56[m], v_x = 4[m/s], v_y = 0[m/s]$:

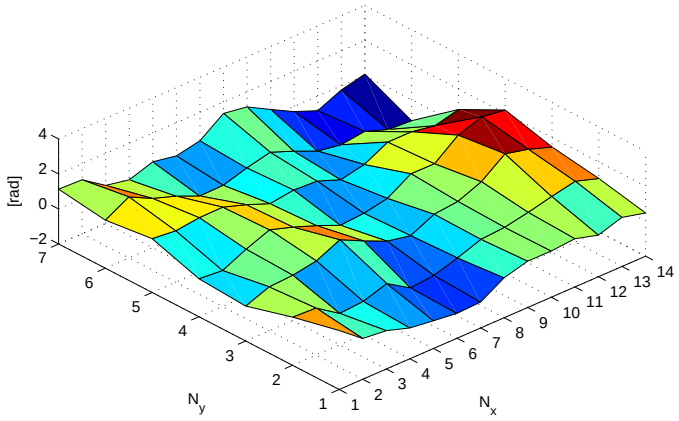


Figure 5-1: atmospheric turbulence at a random time step

For stationary atmospheric turbulence the spatial covariance of the phase between two points at distance r is given by:

$$C_\phi(r) = \frac{\Gamma(11/6)}{2^{5/6}/\pi^{8/3}} \left[\frac{24}{5}\Gamma(6/5)\right]^{5/6} \left(\frac{L_0}{r_0}\right)^{5/3} \left(\frac{2\pi r}{L_0}\right)^{5/6} K_{5/6}\left(\frac{2\pi r}{L_0}\right) \tag{5-1}$$

where $K()$ is the MacDonal function, and Γ is the Gamma function [17], [18].

Subspace identification to find a turbulence state space model

Let $y(k)$ be a vector containing the values of the stationary turbulent phase on a column of the phase grid. Then, we can represent $y(k)$ as the output of the following linear state space model:

$$\begin{aligned} x(k+1) &= Ax(k) + Ke(k) \\ y(k) &= Cx(k) + e(k) \end{aligned} \tag{5-2}$$

where $x \in \mathbb{R}^n, y \in \mathbb{R}^{N_y}, A \in \mathbb{R}^{n \times n}, C \in \mathbb{R}^{N_y \times n}, K \in \mathbb{R}^{n \times N_y}$ and $e(k)$ is a zero mean with noise process.

It is possible, with a subspace identification method, to estimate matrices A, C, K, starting from the function $C_\phi(r)$ [17].

First of all, let $\Lambda_k = E[y(i+k)y(i)^T]$ be the covariance matrix of the output y between two output samples $y(i+k)$ and $y(i)$ (it has dimension $N_y \times N_y$). It can be found from $C_\phi(r)$.

The state space model in 5-2 can be also written in a different way:

$$\begin{aligned} \bar{x}(k+1) &= A\bar{x}(k) + Gu(k) \\ \Lambda_k &= C\bar{x}(k) + Du(k) \end{aligned} \tag{5-3}$$

where $\bar{x} \in \mathbb{R}^{n \times N_y}$, and $u(k)$ is an impulsive input matrix of dimension $N_y \times N_y$ that is defined as:

$$u(k) = \begin{cases} I, & k = 0 \\ 0, & k \neq 0 \end{cases} \tag{5-4}$$

Assuming as initial condition $x(0) = 0$, we directly have that $D = \Lambda_0$, so that we can rewrite it as:

$$\begin{aligned} \bar{x}(k+1) &= A\bar{x}(k) + Gu(k) \\ \Lambda_k &= C\bar{x}(k) + \Lambda_0 u(k) \end{aligned} \tag{5-5}$$

Now, starting from the initial condition, we can find the following Markov parameters [19]:

$$\begin{aligned} &\Lambda_1 = C\bar{x}(1) = CG \\ &\Lambda_2 = C\bar{x}(2) = CA\bar{x}(1) = CAG \\ &\Lambda_3 = C\bar{x}(3) = CA\bar{x}(2) = CA^2\bar{x}(1) = CA^2G \\ &\vdots \\ &\Lambda_i = \cdots = CA^{i-1}G \end{aligned} \tag{5-6}$$

and we can construct the following block Hankel matrix:

$$\begin{aligned} \underbrace{\begin{bmatrix} \Lambda_1 & \Lambda_2 & \cdots & \Lambda_s \\ \Lambda_2 & \Lambda_3 & \cdots & \Lambda_{s+1} \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda_s & \Lambda_{s+1} & \cdots & \Lambda_{2s-1} \end{bmatrix}}_{Y_{1,s,s}} &= \begin{bmatrix} CG & CAG & \cdots & CA^{s-1}G \\ CAG & CA^2G & \cdots & CA^sG \\ \vdots & \vdots & \ddots & \vdots \\ CA^{s-1}G & CA^sG & \cdots & CA^{2s-2}G \end{bmatrix} \\ &= \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-1} \end{bmatrix} \begin{bmatrix} G & AG & \cdots & A^{s-1}G \end{bmatrix} = \mathcal{O}\mathcal{C} \end{aligned} \tag{5-7}$$

where $\mathcal{O} \in \mathbb{R}^{N_y s \times n}, \mathcal{C} \in \mathbb{R}^{n \times N_y s}$. The integer s is chosen by the user. Notice that, if we do i realizations of Λ_k ($k \in [1, i]$), we need to have $2s - 1 \leq i$.

Now, we need to find matrices $\mathcal{O}$ and $\mathcal{C}$ in order to find out A, C, G. If we do a SVD of the matrix $Y_{1,s,s}$ that has dimension $N_y s \times N_y s$, we obtain [20]:

$$Y_{1,s,s} = \begin{bmatrix} U & \bar{U} \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & \bar{\Sigma} \end{bmatrix} \begin{bmatrix} V^T \\ \bar{V}^T \end{bmatrix} = U\Sigma V^T + \bar{U}\bar{\Sigma}\bar{V}^T \tag{5-8}$$

but $\Sigma \in \mathbb{R}^{n \times n}$ has the largest singular values of $Y_{1,s,s}$. Hence, we can do this approximation:

$$Y_{1,s,s} \simeq U\Sigma V^T \tag{5-9}$$

where we can choose $n = \text{rank}(Y_{1,s,s})$, and $U \in \mathbb{R}^{N_y s \times n}, V^T \in \mathbb{R}^{n \times N_y s}$.

This means that, from the SVD, we can find an estimation of $\mathcal{O}$ and $\mathcal{C}$:

$$\begin{aligned} \hat{\mathcal{O}} &= \bar{U}\bar{\Sigma}^{1/2} \\ \hat{\mathcal{C}} &= \bar{\Sigma}^{1/2}\bar{V}^T \end{aligned} \tag{5-10}$$

$\hat{\mathcal{O}}$ and $\hat{\mathcal{C}}$ can be chosen up to a similarity transformation T: $\hat{\mathcal{O}}T = \mathcal{O}$ and $T^{-1}\hat{\mathcal{C}} = \mathcal{C}$.

Then, we can finally find out an estimation of matrices A, C, G:

$$\begin{aligned} &\hat{C} = \hat{\mathcal{O}}(1:l, 1:n); \\ &\hat{A} = \hat{\mathcal{C}}^{\dagger}\hat{\mathcal{O}}(l+1:2l, 1:n) \quad \leftarrow \quad \hat{C}\hat{A} = \hat{\mathcal{O}}(l+1:2l, 1:n); \\ &\hat{G} = \hat{\mathcal{C}}(1:n, 1:l); \end{aligned} \tag{5-11}$$

Now, we can estimate the state covariance matrix P, that is the solution of the Riccati equation:

$$\hat{P} = \hat{A}\hat{P}\hat{A}^T + (\hat{G} - \hat{A}\hat{P}\hat{C}^T)(\Lambda_0 - \hat{C}\hat{P}\hat{C}^T)^{-1}(\hat{G} - \hat{A}\hat{P}\hat{C}^T) \tag{5-12}$$

and then find out also an estimation of the Kalman gain K:

$$\hat{K} = (\hat{G} - \hat{A}\hat{P}\hat{C}^T)(\Lambda_0 - \hat{C}\hat{P}\hat{C}^T)^{-1} \tag{5-13}$$

Therefore, we have found a state space Kalman model equivalent to the one in 5-2 to generate the turbulence column y in simulation:

$$\begin{aligned} x(k+1) &= \hat{A}x(k) + \hat{K}e(k) \\ y(k) &= \hat{C}x(k) + e(k) \end{aligned} \tag{5-14}$$

Moreover, because we assume a frozen flow turbulence, we can evaluate the other columns of the grid by augmenting the model in formula 5-2 saying that:

$$y_{l+1}(k+1) = y_l(k) \tag{5-15}$$

where l is the number of the column, $l \in (1, 13)$. Hence, we have now a linear state space model of the stationary turbulence w. It will be in radians. To know more about the identification of the state space model you can see also [19],[20].

5-1-2 Non-Stationary turbulence

To generate a non-stationary turbulence wavefront phase, we have used the following trick: we have changed the turbulence parameters in a uniform way and we have generated a covariance matrix of the set of parameters. From each of these covariance matrices a state space model was identified. Then, we have generated the non-stationary turbulence by switching between the several stationary models at predetermined time instants. Every time the system matrices were changed, the state vector x was kept constant. It was hoped that by preserving the state, the transition between different stationary turbulence regimes would be smoother.

5-2 Realistic Simulation Scheme

Remembering how we have decoupled the system, this is the MIMO block diagram for the matrix $G(z)$ that we have already seen in the previous Chapter:

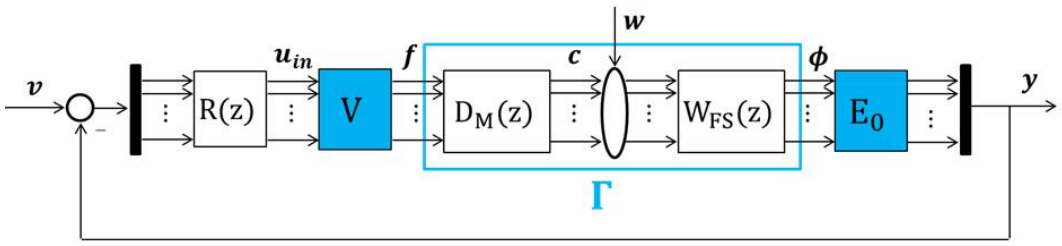


Figure 5-2: MIMO system decoupling configuration

where $D_M(z)$ represents the real DM and $W_{FS}(z)$ represents the real WFS. Unfortunately, in the simulation we can only use models of both DM and WFS.

The easiest way to describe them is with a static model (i.e. without considering their dynamics), and this is what we have done to implement the simulation.
To even more simplify it, we only have considered an open-loop $G(z)$ without the internal controller $R(z)$ and without the feedback of the internal loop.

5-2-1 Realistic scheme with open-loop $G(z)$

Figure 5-3 shows the block diagram in the open-loop configuration (with no feedback and no internal controller), without the disturbance w and without the WFS delay z^{-1}:

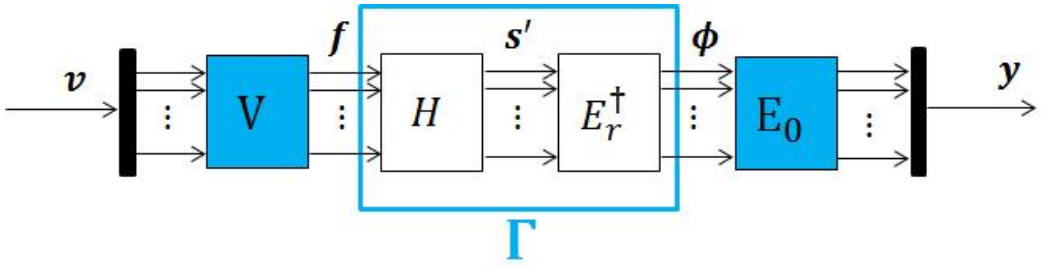


Figure 5-3: open-loop $G(z)$ real simulation

In this system we have that the following relations hold:

- $\boldsymbol{s'} = H\boldsymbol{f}$ where H represents the mapping from voltages $\boldsymbol{f}$ to slopes $\boldsymbol{s'}$. This matrix represents both the DM and the WFS;
- $\boldsymbol{s'} = E_r\boldsymbol{\phi}$ where E_r represents the mapping from phase $\boldsymbol{\phi}$ to slopes $\boldsymbol{s'}$. Matrix E_r describes the phase reconstruction;

It means that we have:

$$\boldsymbol{s'} = E_r\boldsymbol{\phi} = H\boldsymbol{f} \tag{5-16}$$

Hence:

$$\boldsymbol{\phi} = E_r^{\dagger} H \boldsymbol{f} = \Gamma \boldsymbol{f} \tag{5-17}$$

Matrix H is identified from the real system. Matrix E_r was computed using the Hudgin geometry [21].

This MIMO system has 37 channels, because the DM has 37 independent actuators. It means that $\boldsymbol{v}$ is a column vector of dimension 37×1. The matrices that appear in the block diagram are:

- V: has dimension 37×37 ($\boldsymbol{f}$ is a column vector 37×1)
- H: has dimension 254×37 ($\boldsymbol{s'}$ is a column vector 254×1)
- $E_r^{\dagger}$: has dimension 98×254 ($\boldsymbol{\phi}$ is a column vector 98×1)
- E_0: has dimension 37×98 ($\boldsymbol{y}$ is a column vector 37×1)

Notice that Γ has dimension 98×37.

Now we have to add the turbulence w. Because w is the distorted phase, to correctly add the disturbance we have to add it to the signal $\boldsymbol{\phi}$, that is the reconstructed phase. It means that $\boldsymbol{w}$ is a column vector of dimension 98×1 (it follows from the 14×7 size of the phase screen):

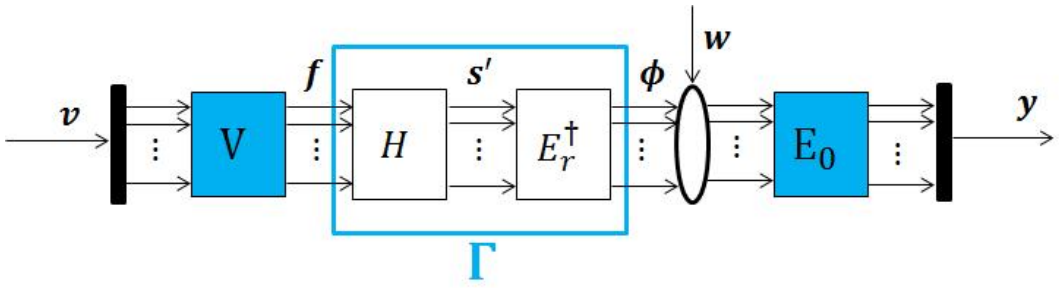


Figure 5-4: open-loop $G(z)$ real simulation with turbulence $\boldsymbol{w}$

Hence, in this simulation, we have that the closed-loop TF $G(z)$ from $\boldsymbol{v}$ to $\boldsymbol{y}$ is actually an open-loop TF:

$$\boldsymbol{y} = G(z)\boldsymbol{v} = (E_0 E_r^\dagger H V)\boldsymbol{v} = (E_0 \Gamma V)\boldsymbol{v} \tag{5-18}$$

The TF from the disturbance $\boldsymbol{w}$ to the residual phase $\boldsymbol{y}$ is:

$$S(z) = E_0 \tag{5-19}$$

Notice that all the TF are static. The only dynamic element is the turbulence $\boldsymbol{w}$. Anyway, it is important to remark that the goal of the adaptive control loop is to predict the estimated turbulence $\hat{\boldsymbol{\omega}}$ at time i because of the delay z^{-1} in the loop (Figure 5-5).

We can then augment the classical open-loop $G(z)$ with the adaptive control loop and obtain the complete block diagram for the MIMO system with also the WFS delay z^{-1} (Figure 5-5):

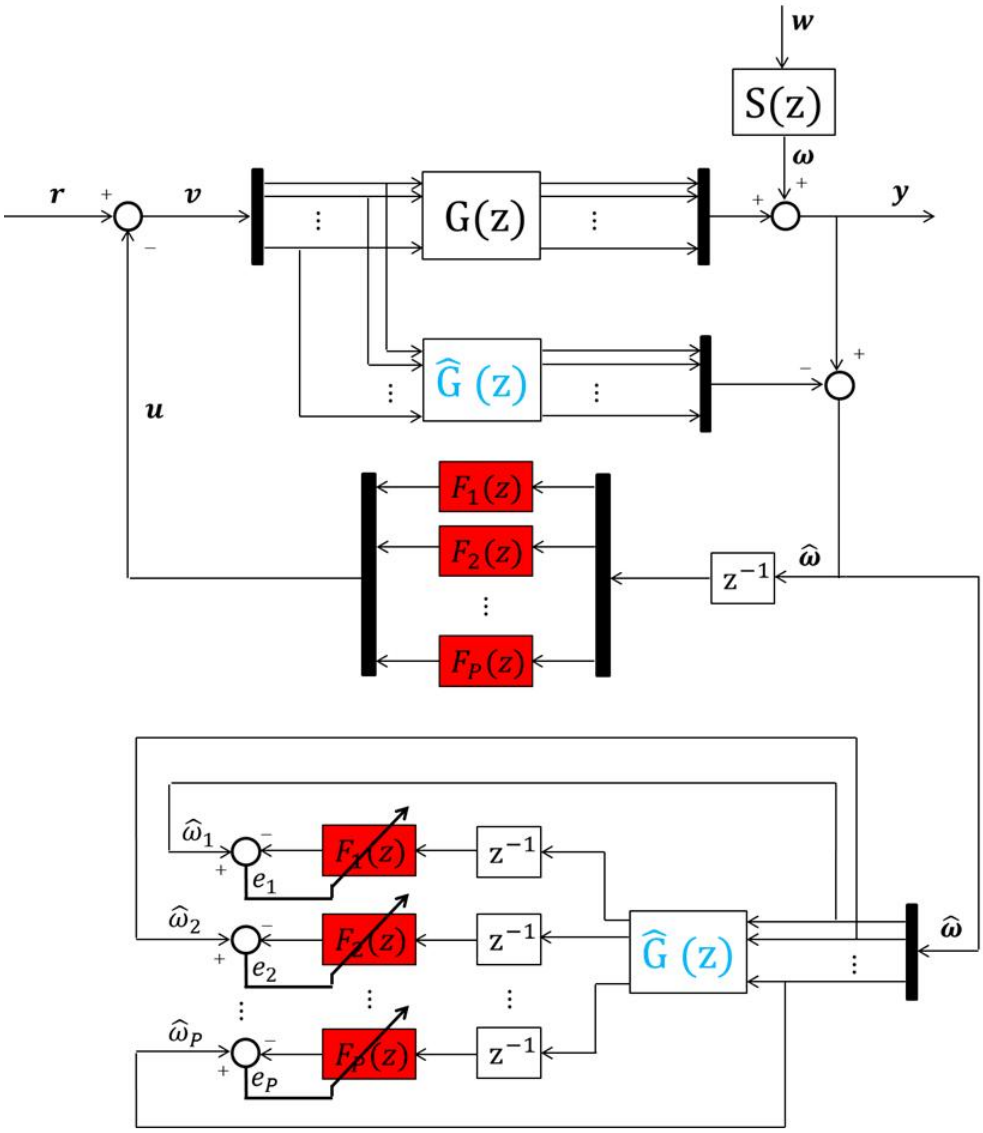


Figure 5-5: realistic MIMO block diagram adaptive control

In this diagram, $\boldsymbol{w}$ is a column vector of dimension 98×1, but $S(z) = E_0$ has dimension 37×98. It means that the new disturbance $\boldsymbol{\omega}$ is a column vector of dimension 37×1.
To realize the adaptive control of the MIMO system, we need 37 lattice filters, one for each channel.

Actually, the matrix $\hat{G}(z)$ can be considered identical to $G(z)$ or we can suppose that $\hat{G}(z) = G(z) + \Delta G$, where ΔG is a diagonal matrix of the same dimension as $G(z)$ that symbolizes the difference that we have between $G(z)$ and $\hat{G}(z)$. Now we have only considered $\hat{G}(z) = G(z)$, but in the real experiment this equality is never verified: if this modeling error is too large, we can cause the system to be unstable with the adaptive loop closed [8].

5-3 Simulation results with stationary turbulence

Now, we can show the results we obtained in rejecting a stationary turbulence with $\hat{G}(z) = G(z)$ and with an order 4 normal lattice filer. But first, we have to explain some details about the simulation to make every thing clear.

Our turbulence $\boldsymbol{w}$ is in radians [rad] while the reconstructed phase $\boldsymbol{\phi}$ is in meters [m]. It means that, to add the disturbance $\boldsymbol{w}$ to $\boldsymbol{\phi}$, we have to convert it to meters:

$$[m] = [rad]\frac{\lambda}{2\pi} \tag{5-20}$$

where $\lambda = 632.8 \times 10^{-9}$ and it is the light beam wavelength.

Then, we have the following block diagram:

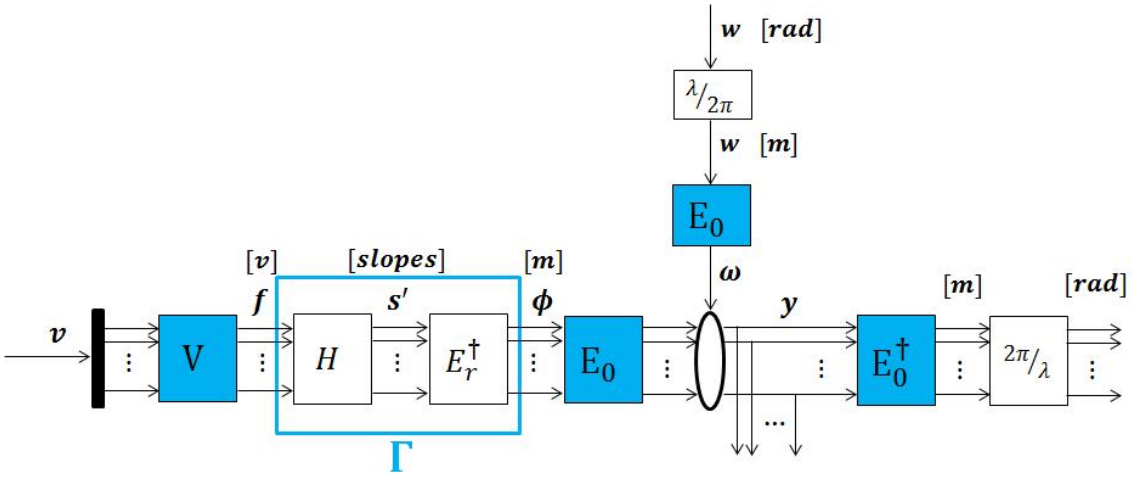


Figure 5-6: open-loop real simulation scheme

Because $E_0 \Gamma V = I$ (see equation 4-4), when we multiply $\boldsymbol{\phi}$ by the matrix E_0, we obtain $\boldsymbol{y}$ in the same measure unit as the signal $\boldsymbol{v}$, that is absolutely not meters. Even $\boldsymbol{\omega}$ has the same measure unit. It means that, the residual phase $\boldsymbol{y}$ that we feedback is not meters neither radians. In order to evaluate the performance of the system, the RMS value of $\boldsymbol{y}$ will be evaluated in radians. That's the reason why we convert $\boldsymbol{y}$ in radians in Figure 5-6.

To avoid this procedure, we can easily implement the following equivalent block diagram for our simulation:

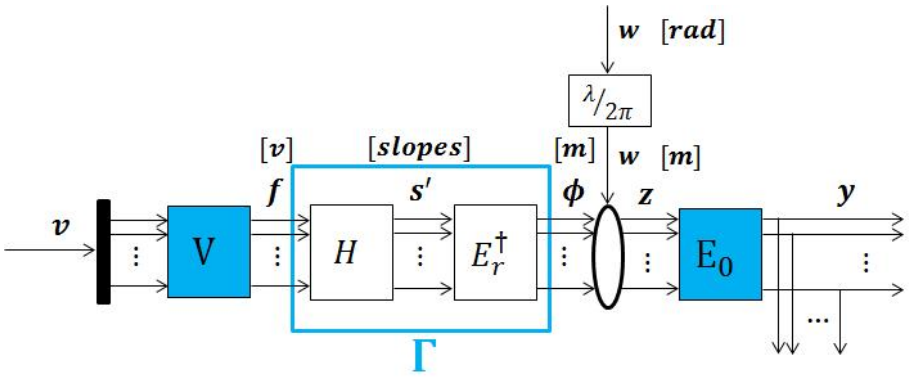


Figure 5-7: open-loop real simulation scheme - second version

With this diagram, we have only to convert the turbulence from $[rad]$ to $[m]$, then we can measure the signal $\boldsymbol{z}$, that will be in meters. To convert it to radians we only have to applicate the conversion in formula 5-20. Then, the residual phase $\boldsymbol{y}$ is the feedback for the adaptive loop.

Now, we can evaluate the RMS values of $\boldsymbol{w}$ and $\boldsymbol{z}$ (the residual phase) over the all grid (that is a 14×7 grid) at every time step, on a time horizon of N=1000 secs, and we have also plotted the rejection in percentage at every time step on the entire grid:

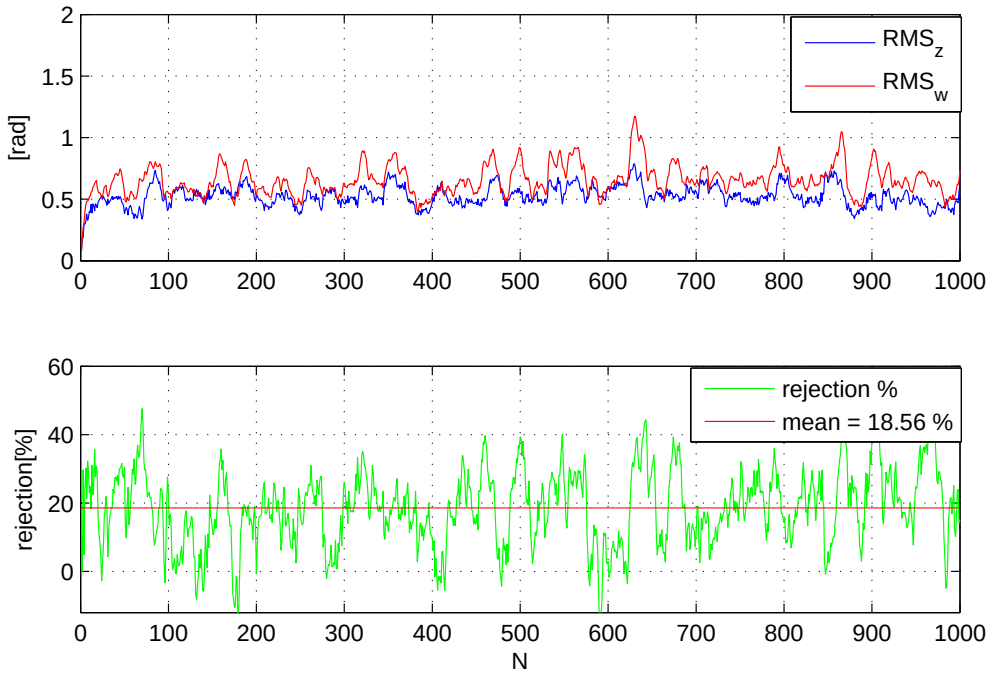


Figure 5-8: RMS_z, RMS_w and stationary turbulence rejection

We can see that the blue line is almost always below the red line, it means that there is a rejection of the turbulence most of the time. The mean rejection is equal to 18.56%.

Then, we have compared the performance of different order lattice filters to see which one works better with this kind of turbulence:

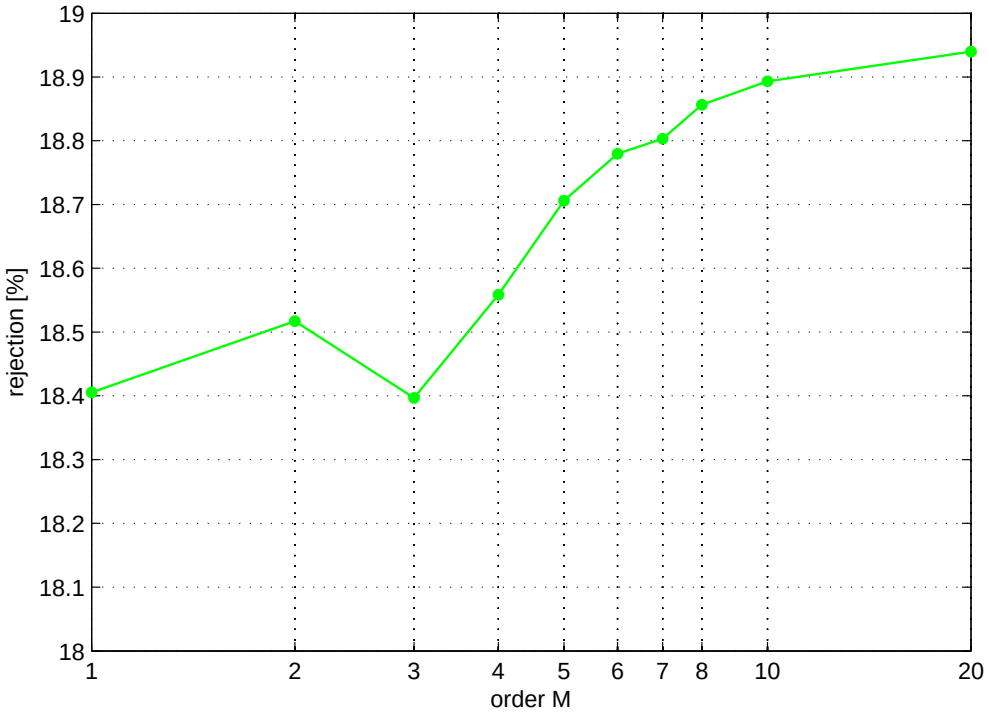


Figure 5-9: rejection VS order in lattice filters

From this figure we decided to choose $M = 4$ for our lattice filter. Actually, we can see that it is not the best performance we can achieve, but the performance increases very slightly if we choose an order between 5 and 10. We assumed that the less computational burden we have if we take $M = 4$ is more important than the increasing of performance we have if we take an order between 5 and 10.

5-4 Simulation results with non-stationary turbulence

Here we have generated a non-stationary turbulence as we explained in 5-1-2. To generate it, we decided to change only the parameter r_0 every 200 secs:

- $r_0(t <= 200) = 0.6 \quad [m]$;
- $r_0(t <= 400) = 0.59 \quad [m]$;
- $r_0(t <= 600) = 0.58 \quad [m]$;
- $r_0(t <= 800) = 0.57 \quad [m]$;
- $r_0(t <= 1000) = 0.56 \quad [m]$.

We have tried to reject this non-stationary turbulence with our order 4 lattice filter, and this is what we obtained:

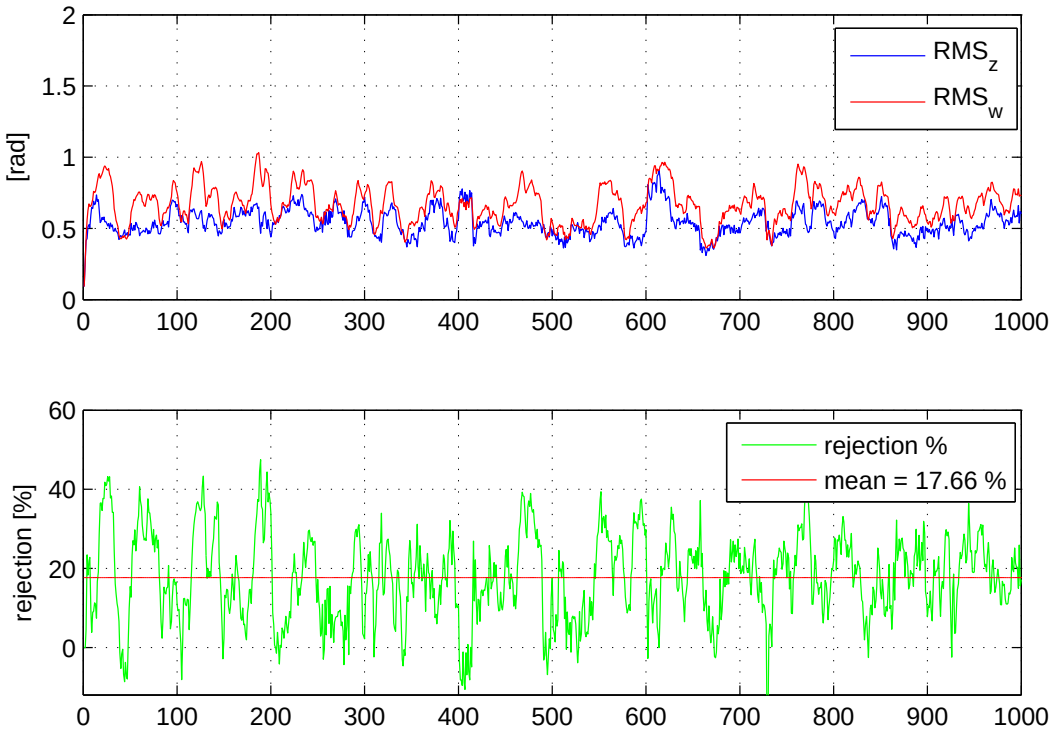


Figure 5-10: RMS_z, RMS_w and non-stationary turbulence rejection

What we can observe is that the performance remain very close to those ones we have with stationary turbulence. Actually this is a slow and not realistic variation of the turbulence, mainly because of the frozen flow assumption: future improvements could focus on understand how a more realistic non-stationary turbulence can be generated. Anyway, we can see that in this example our lattice filter manage to reject the turbulence even if it is non-stationary.

Chapter 6

Experimental Setting and Results

In this Chapter, we present the real system and some experimental results we obtained applying the adaptive configuration. First we have tried to reject a static aberration, then a stationary atmospheric turbulence.

6-1 The Experimental setup

6-1-1 Description of the experiment

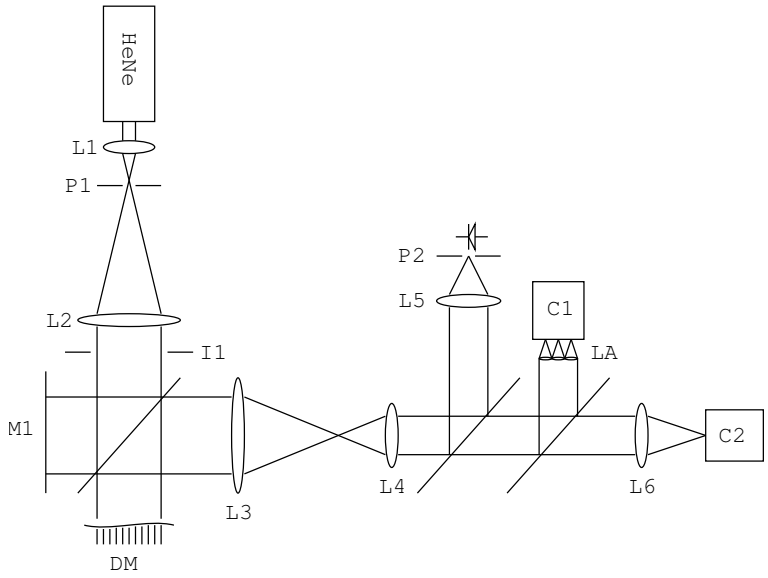


Figure 6-1: experiment

Figure 6-1 shows the optical setup used in the experiments. Light from a HeNe laser source with a wavelength of 632.8nm is spatially filtered by focusing it with a 11mm lens L_1 into a

30μm pinhole P_1. A flat wavefront is then created by collimating the output of the spatial filter with a 500mm lens L_2. An iris I_1 limits the diameter of the beam to 10mm. The membrane of the deformable mirror DM (MMDM37, OKOTech, The Netherlands) is reimaged by L_3 (200mm) and L_4 (100mm) into a microlens array LA (127 microlenses with a focal distance of 18mm). Voltage to the electrodes of the deformable mirror is supplied by a high voltage amplifier (OKOTech, The Netherlands) with 40 channels. The amplifier has an output range of 0 up to 300V, nonetheless, a power supply delivers 150V for safe operation of the deformable mirror. The image of the microlens array is recorded with a camera C_1 (svs340, SYS-VISTEK, Germany). Lens L_5 (200mm) images the pupil into a 50μm pinhole P_2 followed by a photodiode (TSL250R-LF, TAOS, Korea). Lens L_6 (500mm) also images the pupil into a camera C_2 (svs340, SYS-VISTEK, Germany). The system is operated by a desktop PC running Linux. The high voltage amplifier is driven by a 16 bit analog output card (PD2-AO-96/16A, United Electronic Industries, The United States). Whereas the voltage from the photodiode is acquired with a 16 bit analog input card (PCI-6220, National Instruments, The United States). Two framegrabbers cards (Leonardo CL Full, Arvoo, The Netherlands) are used to acquire the signal from C_1 and C_2. Custom software and MATLAB (Version R2011a, The MathWorks, The United States) are used to run the experiments.

This is a picture of the real setup:

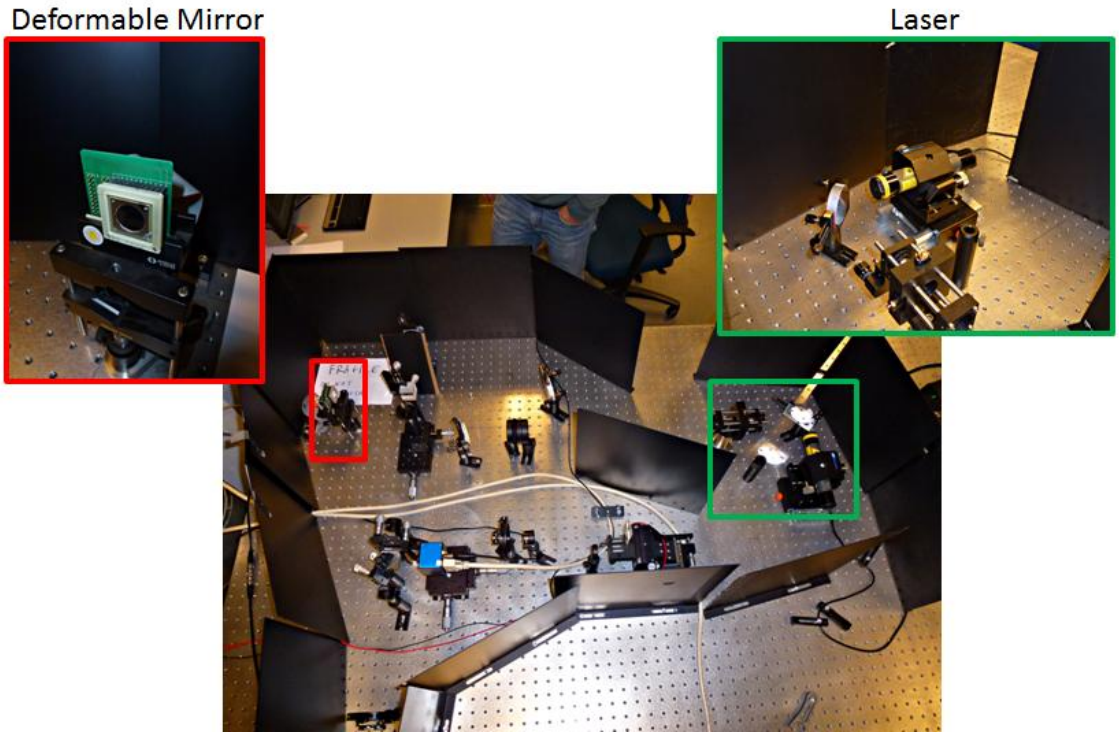


Figure 6-2: picture of the real system

6-1-2 The block diagram configuration

In section 5-2 we have discussed the block diagram for the realistic simulation. Now, we need to describe the block diagram we have in the experiment. The open-loop $G(z)$ in the realistic simulation (Figure 5-4) is this:

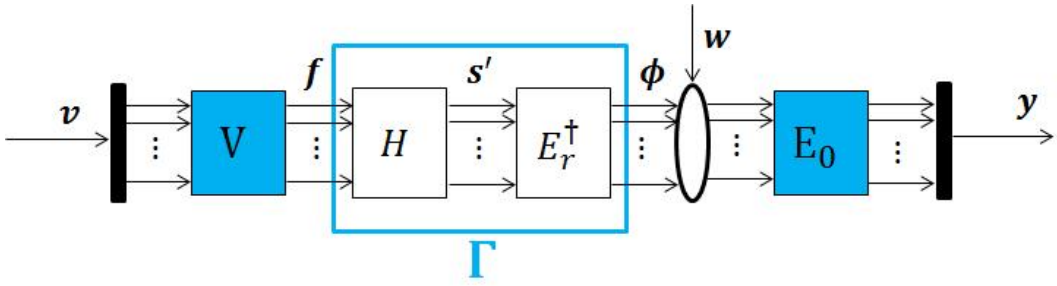


Figure 6-3: open-loop $G(z)$ real simulation with turbulence $\boldsymbol{w}$

First of all, we don't have H anymore, but we have H_{REAL} that represents both the real DM and the real WFS with their dynamics. We don't know H_{REAL}. Then, we need matrix $E_r^\dagger$ to reconstruct the phase from the slopes vector $\boldsymbol{s'}$.

The open-loop $G(z)$ used for the experiment is in Figure 6-4:

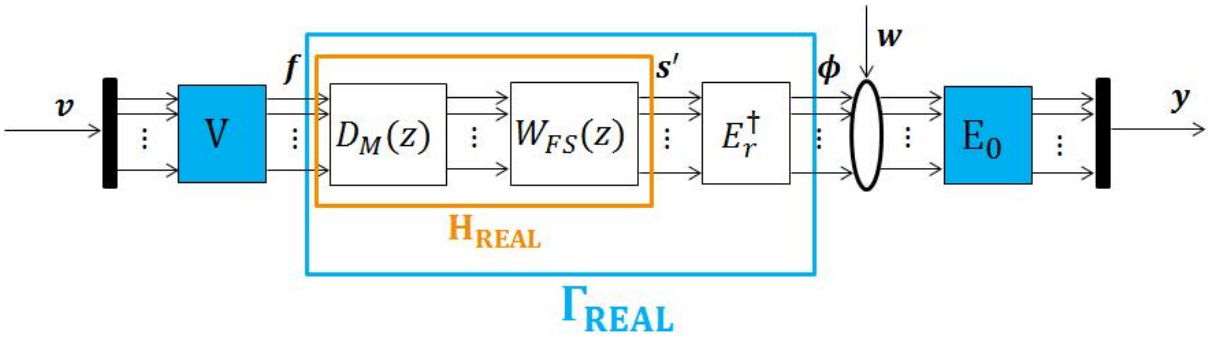


Figure 6-4: open-loop $G(z)$ real experiment with turbulence $\boldsymbol{w}$

that we can also represent like:

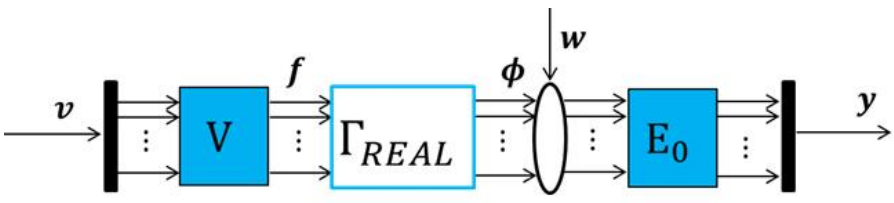


Figure 6-5: open-loop $G(z)$ real experiment with turbulence $\boldsymbol{w}$ - compact

It means that $G(z)$ and $S(z)$ can be defined in this way (we don't know $G(z)$):

$$G(z) = E_0 \Gamma_{REAL} V = E_0 E_r^{\dagger} H_{REAL} V \tag{6-1}$$

$$S(z) = E_0 \tag{6-2}$$

To represent the adaptive control configuration, we also need to define $\hat{G}(z)$. Notice that in the real experiment we always have $\hat{G}(z) \neq G(z)$:

$$\hat{G}(z) = E_0 \Gamma V = E_0 E_r^{\dagger} H V \tag{6-3}$$

that is exactly what is shown in Figure 5-4: it is what we called $G(z)$ in the realistic simulation, where we don't make any difference between $G(z)$ and $\hat{G}(z)$.

Now that we have defined all the matrices, we can show the top part of the block diagram of the real experiment with the WFS delay, without showing the bottom part in which lattice parameters are updated:

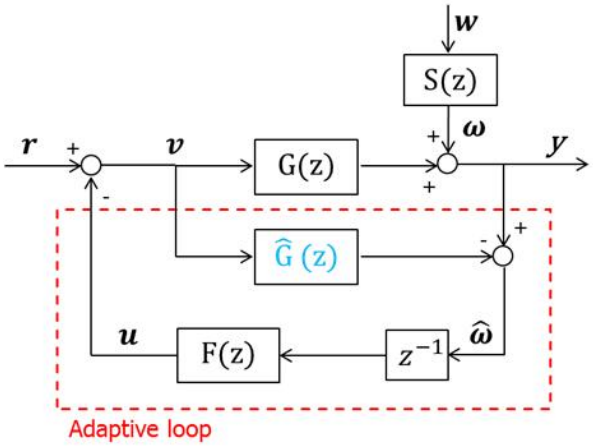


Figure 6-6: top part of adaptive control block diagram - open loop $G(z)$

Problem: in the system in Figure 6-1 however there is no real turbulence source. Therefore Gibson's scheme needs to be modified and the disturbance is added in a different way. Since only the DM can be used as a turbulence generator, we only can introduce $\boldsymbol{w}$ in a voltage form. It means that the turbulence has to be added to the signal $\boldsymbol{v}$ and we need to modify the block diagram like in Figure 6-7:

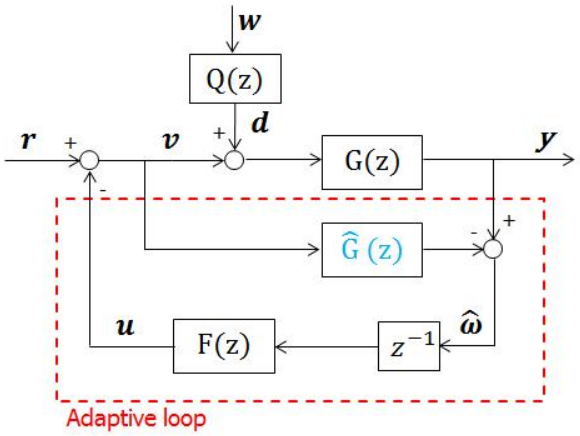


Figure 6-7: top part of adaptive control block diagram for the experiment - open loop $G(z)$

where $Q(z)$ is a matrix of TF that should be chosen to maintain the TF from $\boldsymbol{w}$ to $\boldsymbol{y}$ equal to $S(z) = E_0$. It means that we have to choose $Q(z)$ to satisfy this equation:

$$G(z)Q(z) = S(z) \tag{6-4}$$

But we only know $\hat{G}(z)$, it means that:

$$Q(z) = \hat{G}(z)^{-1}S(z) = (E_0\Gamma V)^{-1}E_0 = (\Gamma V)^{-1} \tag{6-5}$$

and $\boldsymbol{d} = Q(z)\boldsymbol{w}$ is the turbulence in the new configuration.

In this new configuration, we have that

$$\boldsymbol{y} = G(z)(\boldsymbol{v} + \boldsymbol{d}) = G(z)\boldsymbol{v} + G(z)\boldsymbol{d}. \tag{6-6}$$

But $\boldsymbol{d} = Q(z)\boldsymbol{w} = \hat{G}(z)^{-1}S(z)\boldsymbol{w}$. So, if we assume that the modeling error $G(z) - \hat{G}(z)$ is small, we have that:

$$\boldsymbol{y} = G(z)\boldsymbol{v} + G(z)\hat{G}(z)^{-1}S(z)\boldsymbol{w} \simeq G(z)\boldsymbol{v} + S(z)\boldsymbol{w} = G(z)\boldsymbol{v} + \boldsymbol{\omega} \tag{6-7}$$

Hence, we have that this equation holds:

$$\boldsymbol{y} - \hat{G}(z)\boldsymbol{v} = \hat{\boldsymbol{\omega}} \tag{6-8}$$

that is the same we have in Figure 6-6. We can say that Figure 6-6 and in Figure 6-7 are different representations of the same scheme (assuming a small modeling error). So, we can use the block diagram in Figure 6-7 in our experiment.

6-2 The Experimental results

6-2-1 Results with static aberration

First of all, we have tried to reject a static aberration. This disturbance is due to the non flatness of the deformable mirror. It means that, even if we don't add any turbulence to the system, we have an intrinsic aberration due to the initial distortion of the DM. This is a static disturbance, like an offset. If we try to reject this kind of aberration with our lattice adaptive loop with an order 4 lattice filter and we measure the RMS value of the residual phase $\boldsymbol{y}$, this is what we obtain:

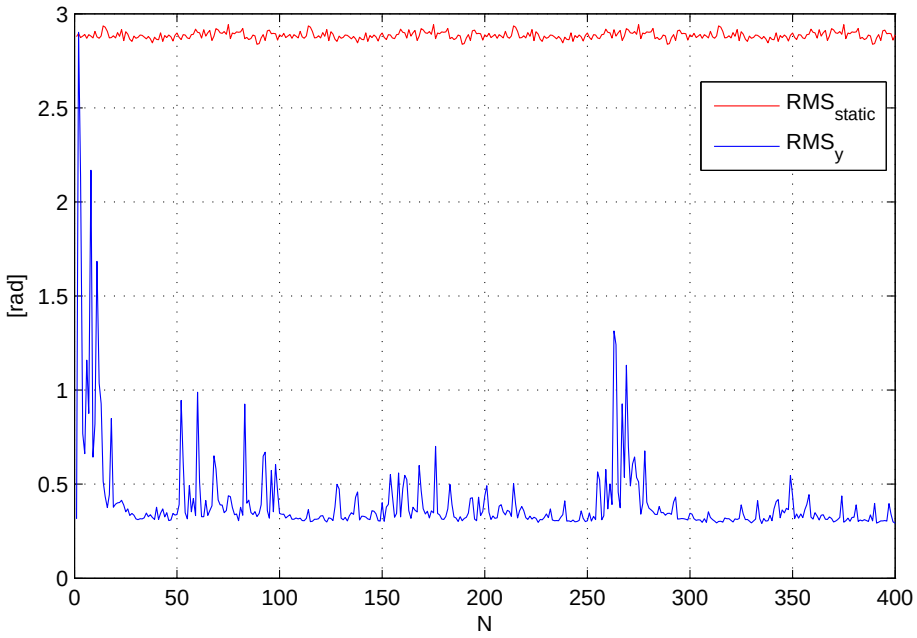


Figure 6-8: RMS_{static} and RMS_y with static aberration due to the DM

where the red line is the RMS of the static aberration. We can observe that we have a really acceptable rejection (more than 75% rejection) if we have a static aberration. What we can observe from Figure 6-8 is that we have an initial learning period, which is around 20 secs (very fast). Then, we manage to reject the aberration even if this rejection is not constant as we expected.

6-2-2 Results with stationary turbulence

In the second experiment we have tried to reject a stationary atmospheric turbulence identical to the one we have tried to reject in the simulation Chapter (see Section 5-3). The difference is that now we have the stationary turbulence $\boldsymbol{w}$ and also the static aberration due to the DM. The RMS of the residual phase we obtain is the blue one:

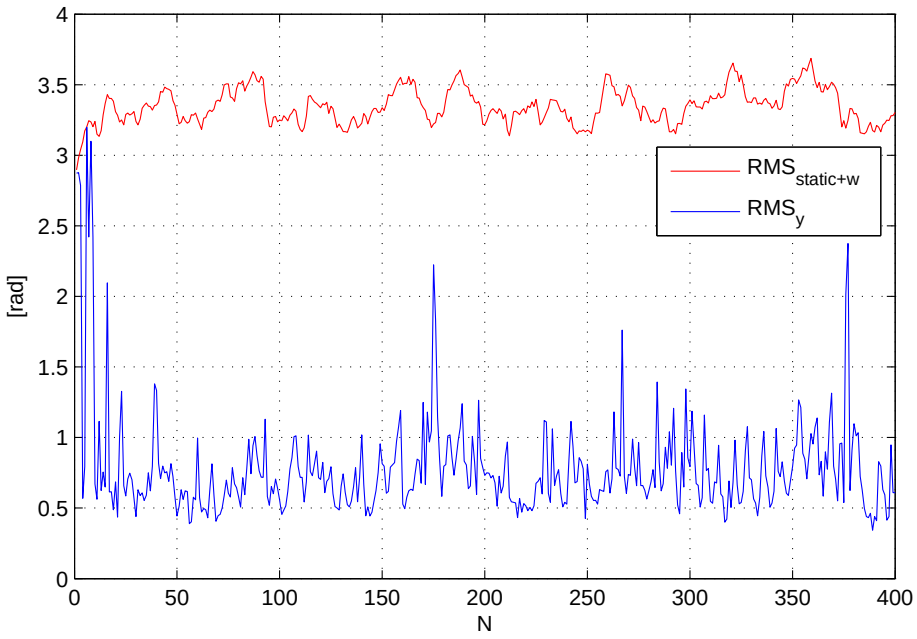


Figure 6-9: $RMS_{static+w}$ and RMS_y with stationary turbulence

We can observe the initial learning period as before. Then, we manage to reject quite well the new atmospheric turbulence even if the main rejection is related to the static aberration. The turbulence $\boldsymbol{w}$ is not actually so well rejected. Probably this is because the model of the DM and of the WFS (that is the matrix H) is not a good estimation of the real model. It means that we have a quite large modeling error $G(z) - \hat{G}(z)$ that often amplifies the turbulence.

Chapter 7

Conclusions and Future Work

In this Thesis we have analyzed the adaptive control loop proposed by Steve Gibson to reject the non-stationary atmospheric turbulence. We have seen that lattice filters have a computational cost that is one order of magnitude less than RLS. From the theory we know that the performance of lattice filters is identical to that of RLS when infinite precision arithmetics operations are performed. Then, we have seen that the forced lattice filters have the same performance as RLS. However, in finite-precision implementation each algorithm will perform differently [15]. We have also seen that lattice filters have some divergence problems in some situation. This observation has not been investigated in this thesis, see [16] for more information. We have done some realistic simulations to evaluate the performance of this adaptive configuration and we have seen that we manage to reject almost the 20% of the turbulence, even without the internal controller $R(z)$ (i.e. with an open-loop $G(z)$). We have seen that it is theoretically possible to reject both stationary and non-stationary turbulence. In our simulations we have tried to generate some realistic non-stationary turbulence by alternating different linear state space models identified from the spatial covariance of different stationary turbulence regimes. It was found that generating non-stationary turbulence in this way has some limitations. Finally, we have done some simple experiment on the real system, where it can be seen that the static aberration due to the initial shape of the DM is effectively canceled. Experiments with a realistic non-stationary turbulence did not provide the expected performance. This might be due to the modeling error $G(z) - \hat{G}(z)$. Nonetheless, due to the limited time available, the cause for this poor performance was not fully investigated.

There are a lot of future improvements that can be realized. For example:

- an internal controller $R(z)$ (for example a PI controller) could be tuned and a closed-loop $G(z)$ could be used to increase the performance;
- the issue of generating realistic non-stationary turbulence should be further investigated to realize more realistic simulations and experiments;
- a more accurate identification of the real system could be realized. If we manage to decrease the modeling error $G(z) - \hat{G}(z)$ we can improve the performance in the experiments;

- limitations of lattice filters could be studied deeply and some corrective measures could be taken;
- a comparison with a similar adaptive loop in which Extended Kalman Filter is used can be performed;
- some useful criteria to select the order of the forced lattice filter $\bar{j}$ at every time step could be found: for example, it could be taken a low order at the beginning or when the turbulence changes (in case of non-stationary turbulence) to have a faster transient, and then the order could be increased to have best performance.

Appendix A

Lattice Filters Matlab Code

```
function [theta_j]  = lattice_filter(omega_hat_i, p_im1, i, k)

% This is the function to implement lattice filters in Matlab

% inputs of the function:
%  -   omega_hat_i is the signal we want to estimate
%  -   p_im1 is the signal we want equal to omega_hat_i
%  -   i is the time step
%  -   k is the lattice filter order

% output of the function:
%  -   theta_j is the parameters vector


M      = k+1;
eta    = 1000;   % this is the weight for the initial estimation
    parameters vector
lambda = 0.9999; % forgetting factor (it has to be major than 0.96)
t      = M;
c      = 2;
j      = 1;


% this is the initialization of the variables that are reset at every
    time step
kappa     = zeros (k,1);
f         = zeros (t,c);
r         = zeros (t,c);
vett_r    = zeros (1,M-1);
theta_i_b = zeros(k,k);
theta_i_f = zeros(k,k);

% the following variables have to be passed from time i to time i+1
```

```
32    persistent zeta_f
33    persistent zeta_b
34    persistent delta
35    persistent gamma
36    persistent rho
37    persistent b
38    persistent theta_im1_b
39    persistent kappa_b
40    persistent kappa_f
41
42
43     if  isempty(zeta_f)
44
45         % this is the initialization at time t=0
46         zeta_f      = zeros (t,c);
47         zeta_b      = zeros (t,c);
48         delta       = zeros (t,c);
49         gamma       = zeros (t,c);
50         rho         = zeros (t,c);
51         b           = zeros (t,c);
52         theta_im1_b = zeros (k,k);
53         kappa_b     = zeros (k,k);
54         kappa_f     = zeros (k,k);
55
56         for m = 1:1:(M-1)
57             delta(m,1)  = 0;
58             rho(m,1)    = 0;
59             gamma(m,1)  = 1;
60             b(m,1)      = 0;
61             zeta_f(m,1) = eta^(-1) * lambda^(-2);
62             zeta_b(m,1) = eta^(-1) * lambda^(-m-2-1);
63         end
64
65     end
66
67     if (i>=2)
68
69         % this is the initialization at every time step i
70         gamma(1,2)   = 1;
71         b(1,2)       = p_im1;
72         f(1,2)       = p_im1;
73         r(1,2)       = omega_hat_i;
74
75         % the following 'for' calculates the variables at time step i.
76         % the colomn 1 has the value of the variable at time i-1
77         % the column 2 has the value of the variable at time i
78         % the rows represent the order of the variables (row 0 is order
               1, and so on)
79
80         for m = 1:1:(M-1)
81             zeta_f(m,2)    = lambda * zeta_f(m,1) + (abs(f(m,2)^2) /
                   gamma(m,1));
```

```
82              zeta_b(m,2)     = lambda * zeta_b(m,1) + (abs(b(m,2)^2) /
                    gamma(m,2));
83              delta(m,2)      = lambda * delta(m,1) + ((conj(f(m,2))' * b(m
                    ,1)) / gamma(m,1));
84              rho(m,2)        = lambda * rho(m,1) + ((conj(r(m,2))' * b(m,2)
                    ) / gamma(m,2));
85              kappa_b(m,k)    = delta(m,2) / zeta_f(m,2);
86              kappa_f(m,k)    = conj(delta(m,2))' / zeta_b(m,1);
87              kappa(m,1)      = conj(rho(m,2))' / zeta_b(m,2);
88              gamma(m+1,2)    = gamma(m,2) - (abs(b(m,2))^2 / zeta_b(m,2));
89              b(m+1,2)        = b(m,1) - (kappa_b(m,k) * f(m,2));
90              f(m+1,2)        = f(m,2) - (kappa_f(m,k) * b(m,1));
91              r(m+1,2)        = r(m,2) - (kappa(m,1) * b(m,2));
92
93              % here we put in the first column the variable value at time
                    i-1
94              zeta_f(m,1) = zeta_f(m,2);
95              zeta_b(m,1) = zeta_b(m,2);
96              delta(m,1)  = delta(m,2);
97              rho(m,1)    = rho(m,2);
98              gamma(m,1)  = gamma(m,2);
99              b(m,1)      = b(m,2);
100
101
102             for p = 1:(k-1)
103                 kappa_b(m,p) = kappa_b(m,p+1);
104                 kappa_f(m,p) = kappa_f(m,p+1);
105             end
106
107         end
108
109         for p = 2:1:M
110             vett_r(p-1) = r(p,2);
111         end
112
113         x = min(abs(vett_r));
114
115
116         for t = 2:1:M
117             if x == abs(vett_r(t-1))
118                 j = t-1; % j is the order chosen by the lattice filter at
                        time i
119                 break
120             end
121         end
122
123     end
124
125
126     % now, we have to calculate the parameters vector
127
128     if j>=2
129         for m = 1:(j-1)
```

```
130             if m==1
131                 theta_i_b(m,m+1) = kappa_b(m,k);
132                 theta_i_f(m,m+1) = kappa_f(m,k);
133             else
134                 theta_i_b(1:m,m+1) = [0 theta_im1_b(1:m-1,m)']' - kappa_b
                        (m,k) * [-1 theta_i_f(1:m-1,m)']';
135                 theta_i_f(1:m,m+1) = [theta_i_f(1:m-1,m)' 0]' - kappa_f(m
                        ,k) * [theta_im1_b(1:m-1,m)' -1]';
136             end
137         end
138     end
139
140
141     theta_j = zeros(k,1);
142
143     if j==1
144         theta_j(1,1) = kappa(1,1);
145     else
146         theta_j(1,1) = kappa(1,1);
147         for p = 2:j
148             theta_j(1:p,1) = [theta_j(1:(p-1),1)' 0]' + kappa(p,1) * [-
                    theta_i_b(1:(p-1),p)' 1]';
149         end
150     end
151
152     theta_im1_b(:,:) = theta_i_b(:,:);
153
154 end  % end of the function
```

Bibliography

[1] K. Hinnen, *Data driven optimal control for adaptive optics*. January 2007.

[2] R. Tyson, *Principles of Adaptive Optics*. CRC PRESS, 3rd ed., 2011.

[3] M. Sadeghi, "Evaluation of recursive algorithms for nonstationary disturbance prediction in an adaptive optics experimental setup," August 2011.

[4] S. Monirabbasi and S. Gibson, "Adaptive control in an adaptive optics experiment," *J. Opt. Soc. Am. A*, vol. 27, pp. A84–A96, Nov 2010.

[5] Y.-T. Liu and J. S. Gibson, "Adaptive control in an adaptive optics for directed energy system," in *Optical Engineering*, april 2007.

[6] J. S. Gibson, J. Tesch, S. Gordoyev, and E. Jumper, "Identification, prediction and control of aero optical wavefronts in laser beam propagation," in *41st AIAA Plasmadynamics and Lasers Conference*, june 2011.

[7] J. S. Gibson and J. Tesch, "Optimal and adaptive correction of aero-optical wavefronts in an adaptive optics experiment," in *Unconventional Imaging and Wavefront Sensing VII (San Diego, CA), SPIE*, 2011.

[8] B.-S. Kim, S. Gibson, and T.-C. Tsao, "Adaptive control of a tilt mirror for laser beam steering," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 4, pp. 3417 –3421 vol.4, 30 2004-july 2 2004.

[9] N. O. P. Arancibia, N. Y. Chen, J. S. Gibson, and T.-C. Tsao, "Variable-order adaptive control of a microelectromechanical steering mirror for suppression of laser beam jitter," *Optical Engineering*, vol. 45, no. 10, p. 104206, 2006.

[10] P. Orzechowski, N. Chen, J. Gibson, and T.-C. Tsao, "Optimal suppression of laser beam jitter by high-order rls adaptive control," *Control Systems Technology, IEEE Transactions on*, vol. 16, pp. 255 –267, march 2008.

[11] Y.-T. Liu, N. Chen, and S. Gibson, "Adaptive filtering and control for wavefront reconstruction and jitter control in adaptive optics," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 2608 – 2612 vol. 4, june 2005.

[12] J. Gibson, C.-C. Chang, and N. Chen, "Adaptive optics with a new modal decomposition of actuator and sensor spaces," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 6, pp. 4619 –4625 vol.6, 2001.

[13] E. W. Bernard Wildrow, *Adaptive Inverse Control - a signal processing approach.* IEEE PRESS, 3rd ed., 2008.

[14] A. H. Sayed, *Fundamentals of Adaptive Filtering.* University of California, Los Angeles: IEEE PRESS, 1st, ed., 2003.

[15] P. S.R.Diniz, *Adaptive Filtering: Algorithms and Practical Implementation.* July 2008.

[16] H. Lev-Ari, K.-F. Chiang, and T. Kailath, "Constrained-input/constrained-output stability for adaptive rls lattice filters," *Circuits and Systems, IEEE Transactions on*, vol. 38, pp. 1478 –1483, dec 1991.

[17] A. Beghi, A. Cenedese, and A. Masiero, "System theoretic tools in adaptive optics," in *Control and Automation, 2009. ICCA 2009. IEEE International Conference on*, pp. 1049 –1054, dec. 2009.

[18] R. Conan, *Modélisation des effets de l'échelle externe de cohérence spatiale du front d'onde pour l'observation à haute résolution angulaire en astronomie.* October 2000.

[19] P. Van Overschee and B. De Moor, "Subspace algorithms for the stochastic identification problem," in *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, pp. 1321 –1326 vol.2, dec 1991.

[20] T. Katayama, *Subspace Methods for System Identification.* Springer, 2005.

[21] R. H. Hudgin, "Wave-front reconstruction for compensated imaging," *J. Opt. Soc. Am.*, vol. 67, pp. 375–378, Mar 1977.

Glossary

List of Acronyms

AO	Adaptive Optics
AR	Auto Regressive
DM	Deformable Mirror
IMC	Internal Model Control
LTI	Linear Time Invariant
LQG	Linear Quadratic Gaussian
LS	Least Square
MIMO	Multi input Multi Output
PI	Proportional Integrative
PID	Proportional Integrative Derivative
RLS	Recursive Least Square
RMS	Root Mean Square
SISO	Single Input Single Output
TF	Transfer Function
WFS	Wavefront Sensor
SVD	Singular Value Decomposition