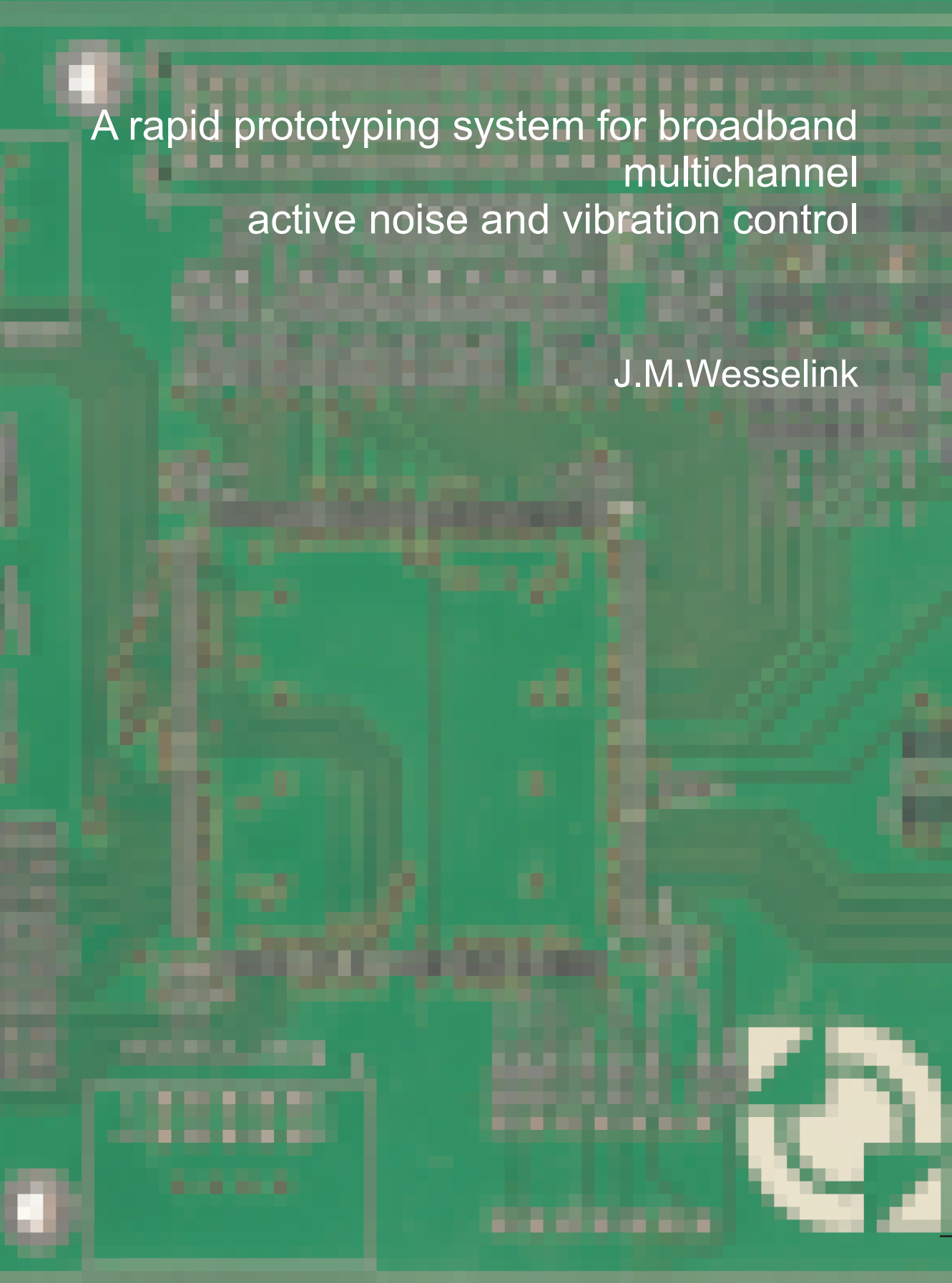
A rapid prototyping system for broadband multichannel active noise and vibration control

J.M.Wesselink



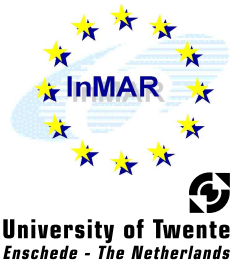
A rapid prototyping system for broadband multichannel active noise and vibration control

Johan M. Wesselink

Thesis committee:

chair and secretary:	
Prof.dr.ir. A.J. Mouthaan	University of Twente
promoter:	
Prof.dr.ir. C.H. Slump	University of Twente
assistant promoter:	
Dr.ir. A.P. Berkhoff	TNO Science and Industry / University of Twente
referee:	
Dr.-ing. T. Bein	Fraunhofer Institut für Betriebsfestigkeit und Systemzuverlässigkeit LBF
members:	
Prof.dr.ir. J. van Amerongen	University of Twente
Prof.dr. P.J.M. Havinga	University of Twente
Prof.dr.ir M. Verhaegen	TU Delft
Prof.dr.ir. A. de Boer	University of Twente

The research presented in this thesis was funded by the European commission and was part of the InMAR project.



Signals & Systems group,
EEMCS Faculty, University of Twente
P.O. Box 217, 7500 AE Enschede, the Netherlands

Printed by Gildeprint, Enschede, The Netherlands
Typesetting in LaTeX2e

ISBN 978-90-365-2936-5
DOI 10.3990/1.9789036529365

A RAPID PROTOTYPING SYSTEM FOR BROADBAND MULTICHANNEL ACTIVE NOISE AND VIBRATION CONTROL

DISSERTATION

To obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof.dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended on November 26th, 2009 at 15:00 hrs.

by

Johan Marius Wesselink
born on the 9th of March 1971
Rijssen, The Netherlands

This dissertation has been approved by:

the promoter:	Prof.dr.ir. C.H. Slump
the assistant promoter:	Dr.ir. A.P. Berkhoff

Opgedragen aan mijn vader , omdat die vroeger altijd dacht dat ik groenteboer zou worden.

Bedankt voor alles, zonder jou was dit nooit gelukt.

Wersse Wesselink, 7 Juni 1934 tot 11 September 2009

Samenvatting

In onze moderne samenleving is geluidsoverlast een steeds groter wordend probleem. Dit wordt onder andere veroorzaakt door industrialisatie, een toename in lucht-, weg- en spoorverkeer, en het toenemend gebruik van apparaten. De gevolgen hiervan zijn, naast mogelijke gehoorschade, dat mensen slecht slapen, zich slecht kunnen concentreren en last hebben van nervositeit en een verhoogde bloeddruk. Hierdoor is er een vraag ontstaan naar methoden voor actieve en/of passieve lawaaibestrijding. In dit proefschrift wordt een systeem beschreven voor het ontwikkelen van prototypes voor actieve lawaaibestrijding.

Het in dit proefschrift gepresenteerde ontwikkelsysteem is geschikt voor het realiseren van controllers voor actieve lawaai- en trillingsreductie. Dit systeem bestaat uit een embedded PC en een interface kaart met 16 analoge ingangen en 16 analoge uitgangen. De algoritmes zijn ontwikkeld met een rapid prototyping development omgeving welke is gebaseerd op RealTime Linux (RTLinux), Matlab/Simulink en de realtime workshop (RTW). Het gekozen adaptieve control algoritme is model gebaseerd en gebruikt het regularized modified filtered error least mean square (RMFeLMS) principe. Dit algoritme combineert goede convergentie eigenschappen met een relatieve lage complexiteit. Verder is dit algoritme goed geschikt voor systemen met meerdere in- en uit-gangen, omdat het interne model wordt gerepresenteerd door een state-space realisatie. Een eigenschap van de state space realisatie is dat de complexiteit dominant afhankelijk is van de orde van het systeem. De complexiteit van het state space model kan verder worden gereduceerd door het gebruik van de output normalized vorm. Het gerealiseerde systeem is geschikt voor meerdere applicaties. Hierbij kunnen zowel systemen met slechts 1 in- en uitgang als ook systemen met meerdere in- en uitgangen worden gerealiseerd. De compactheid van het systeem maakt het geschikt voor mobiele applicaties, bijvoorbeeld in een auto.

De werking van het systeem is geverifieerd met behulp van een actief paneel (smart panel). Een dergelijk paneel wordt gerealiseerd als een sandwich constructie van drie lagen. De twee buitenste lagen bestaan uit een printed circuit board (PCB) welke ook de sensoren en de actuatoren bevatten. De middelste laag bestaat of uit een schuimlaag of uit een honingraat structuur. Dit paneel is daarna op een perspex box gemonteerd. In de bodem van deze box was een luidspreker gemonteerd welke werd gebruikt als de primaire bron. Een sub-space identificatie

methode werd gebruikt om de overdracht van de primaire en secundaire paden te schatten. Deze modellen zijn nodig voor het ontwerp van de controller en voor de schatting van de maximale reductie van het foutsignaal. Verder is het mogelijk om een simulatie uit te voeren met deze modellen. Het gerealiseerde regelsysteem was een configuratie met 5 in- en uitgangen die zowel in feedforward als ook in feedback is getest.

De convergentie snelheid van het RMFeLMS algoritme wordt beperkt indien de referentiesignalen gekleurd zijn. Dit wordt veroorzaakt door het gebruik van het LMS algoritme dat minder goed presteert indien de referentie signalen gekleurd zijn. Een algoritme dat minder last heeft van deze beperking is het affine projection (AP) algoritme. De complexiteit van dit algoritme is echter hoger. Dit kan worden beperkt door het gebruik van de snelle versie van het AP algoritme dat bekend staat als het fast affine projection (FAP) algoritme. Het FAP algoritme heeft een aanmerkelijk lagere complexiteit dan het AP algoritme. De extensie maakt het mogelijk om het RMFeLMS algoritme uit te breiden met de FAP methode wat resulteerde in het RMFeFAP algoritme. Een voordeel van dit algoritme is dat het beter presteert indien de referentie signalen gekleurd zijn. Dit is bijvoorbeeld het geval in een ventilatie kanaal indien het referentie signaal wordt opgenomen met een microfoon. Om de resultaten van dit nieuwe algoritme te verifiren was het noodzakelijk om de primaire bron te vervangen door een configuratie die het referentie signaal kleurt. Het nieuwe algoritme werd zowel met behulp van simulatie als ook met realtime experimenten getest en geverifieerd.

De RMFeLMS en RMFeFAP algoritmes zijn nog steeds model gebaseerde control algoritmes die beïnvloed worden door verschillen tussen het model en de werkelijkheid. Deze verschillen zijn te wijten aan de resonanties (polen) en anti-resonanties (nulpunten) van het systeem. De invloed van deze modelfout werd verminderd door het aanbrengen van locale feedback loops welke gebruik maken van signaal paren die dual and collocated zijn. Dit resulteerde in een systeem met minder extreme resonantie en anti resonantie pieken, waardoor de controller minder gevoelig wordt voor modelfouten. Dit principe kan worden gecombineerd met een model gebaseerde controller (HAC). Een dergelijk systeem staat bekend als een low-authority/high authority (LAC/HAC) controller. De in dit proefschrift gepresenteerde methode maakt gebruik van een LAC controller die is gerealiseerd als een snelle digitale controller. Het voordeel van deze realisatie is dat die flexibel en herconfigureerbaar is. Bijkomend voordeel van de LAC/HAC structuur is een lagere gemiddelde variantie (MSE) en een betere robuustheid. Dit is waargenomen en aangetoond door middel van experimenten. De robuustheid is geverifieerd door extra gewicht aan het paneel toe te voegen, wat resulteerde in een verschuiving van de resonantie en anti-resonantie punten. Uit experimenten is gebleken dat een LAC/HAC structuur minder gevoelig is voor modelfouten dan een systeem met alleen een HAC controller.

Summary

In recent years the need for active and passive noise reduction methods has increased. This is due to an increase in the ambient noise caused by industrialization and the extended use of power tools. The effects of noise on a person can be quite severe and can cause illness and in severe cased lead to a loss of hearing. This results in a need for a method or tool to develop active and/or passive noise control systems. In this thesis a method was presented that can be used to rapidly test and evaluate active noise control systems.

The development system presented in this thesis consists of a highly integrated controller which can be used for different active noise and vibration control (ANVC) applications. The system consists of an embedded PC and an interfacing card that provides up to 16 analog input and output channels. The algorithms were developed and evaluated using a rapid prototyping environment based on RealTime Linux, Matlab/Simulink and the Realtime Workshop (RTW). The controller is model based and uses the RMFeLMS algorithm. This algorithm combines good convergence properties with a relatively low computational complexity. It is especially well suited for multiple input and multiple output systems due to the fact that the model is realized as a state space description. The complexity of this algorithm is lowered further by using an output normalized state space realization. The overall development system is suited for different applications. It can be used for single channel (SISO) as well as for multiple channel (MIMO) systems and experimental setups. The realized system is relatively compact and can be used in mobile applications, such as in cars.

The working of the system was verified by means of an active panel. This panel consists of a sandwich construction of three layers. The two outer layers consist of printed circuit boards which also contain the sensors and actuators. The inner layer is either a foam based material or it is a material that has a honeycomb structure. The panel was mounted in a perspex box. The bottom of the box contained a loudspeaker that was used to generate the primary noise signal. A sub-space identification method was used to estimate the primary and secondary transfer functions. These models where then used to simulate the overall system. The results from these simulations were compared to practical experiments on the same setup. The experiments and simulations were performed using a 5 channel controller which was configured in a feedforward as well as feedback

configuration. This test showed the applicability of the development system for ANVC applications.

The RMFeLMS algorithms still suffers from suboptimal convergence properties when the reference signals are colored. In the RMFeLMS algorithm this is caused by the LMS adaptation mechanism which does not behave well if the inputs are colored. An adaptation algorithm that suffers less from this limitation is the affine projection algorithm. The complexity of this algorithm is higher than that of the LMS algorithm. However, it is possible to reduce this complexity by using a fast derivative, the fast affine projection (FAP) algorithm. Therefore the RMFeLMS was extended to include the FAP adaptation algorithm resulting in the RMFeFAP algorithm. This new algorithm is suited for applications in which the reference signals are colored. This is the case in for instance a duct-like structure using a microphone as a reference signal. In the test setup a duct-like structure was used to replace the primary source loudspeaker. The results and performance of this new algorithm were verified by means of experiments and simulations.

The RMFeLMS and RMFeFAP algorithm are model-based control algorithms that still suffer from potential model mismatch. This model mismatch can be traced back to the resonances (poles) and anti-resonances (zeros). Furthermore, the model is estimated off-line. The influence of model mismatch was reduced by adding a feedback controller that used dual and collocated signal pairs. This resulted in a system with less extreme resonance and anti-resonance peeks, making the overall system less sensitive to model mismatch. These low-authority control loops were combined with the high-authority multichannel adaptive controller. This resulted in a low-authority control / high-authority control (HAC/LAC) strategy. The method presented in this thesis is based on a low-authority controller that is implemented as a high-speed digital control loop, resulting in a flexible and reconfigurable system. The advantage of such a structure is that it improves the overall mean square error (MSE) and robustness of the system. This was verified by means of experiments. The robustness was tested by adding extra weight to the panel resulting in shifted resonance and anti resonance peaks. It was shown that the HAC/LAC structure behaves better under model mismatch than the system without low-authority control.

Contents

Nomenclature

Abbreviations

AD	Analog to Digital
ADDA	Analog to Digital and Digital to Analog
ANC	Active Noise Control
ANVC	Active Noise and Vibration Control
AP	Affine projection
AR	Autoregressive
ARMA	Autoregressive Moving Average
ASAC	Active Structural Acoustic Control
AVC	Active Vibration Control
DA	Digital to Analog
FAP	Fast Affine projection
FeLMS	Filtered-error LMS
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FRLS	Fast Recursive Least Squares
FxLMS	Filtered-reference LMS
HAC	High-Authority Control
HFeLMS	Hybrid filtered error LMS
IIR	Infinite Impulse Response
IMC	Internal Model Control
InMAR	Intelligent Material for Active Noise Reduction
LAC	Low-Authority Control
LMS	Least Mean Square
LQR	Linear Quadratic Regulator

MA	Moving Average
MIMO	Multiple-input Multiple-output system
OS	Operating System
PCB	Printed Circuit Board
PLL	phase locked loop
PLMS	preconditioned filtered error LMS
PPF	Positive Position Feedback
RLS	Recursive Least Squares
RMFe	Regularized Modified Filtered Error
RMFeAP	Regularized Modified Filtered-Error AP
RMFeFAP	Regularized Modified Filtered-Error FAP
RMFeLMS	Regularized Modified Filtered-Error LMS
SISO	Single-input single-output system
SRT	Simulink Realtime Target

Mathematical Symbols

α	Step size of the LMS algorithm
β	RMFe regularization level
γ	Leakage of the LMS algorithm
d(n)	Vector of L disturbance input signals
e(n)	Vector of L error input signals
u(n)	Vector of M control output signals
x(n)	Vector of K reference input signals
y(n)	Vector of L process output signals

Chapter 1

Introduction

In this chapter a brief introduction to the different methods for noise and vibration reduction is given. Noise and vibration reduction can be realized using active as well as passive methods. In this thesis the focus is on active noise control. A short survey of the state of the art is given. Finally, the contributions presented in this thesis are summarized.

1.1 Introduction

1.1.1 Influence of noise on its environment

The influence of noise on our modern day society has been studied extensively. The main cause of an increase in noise level is due to the industrialization and the extended use of power tools. This 'noise pollution' can lead to a feeling of unpleasantness and in severe cases can lead to hearing loss and illness (See Ref. [1]). The development of suitable solutions can be driven by legislation or by the desire to increase comfort.

1.1.2 Applications of active noise and vibration control

In this section some examples of Active Noise Control (ANC), Active Vibration Control (AVC) and Active Noise and Vibration Control (ANVC) in everyday life are given. It also contains an outlook on ongoing research in the field of active noise control. Some commercial applications of ANC, AVC or ANVC are:

> Reduction of noise inside a car using the already installed car audio system. This system is installed in cars produced by Honda and Toyota. Using the on-board audio system reduces the overall cost, making the system more cost effective. This system is described in Ref. [2].

Reduction of the propeller-induced cabin noise in propeller aircraft, such as aircrafts made by Saab (See Ref. [3]), Bombardier, De Havilland. In a propeller aircraft the noise within the passenger cabin dominantly consist of tonal components. It was very early recognized that this could be solved by reducing the fundamental and some harmonics, see for instance Refs. [4–6].

A headphone that cancels noise from external sources. Such a headphone uses a simple analog feedback controller. This principle is adopted by several vendors such as for instance, for domestic applications, Bose, Sennheiser, Sony and other companies, and for industrial applications, David Clark, Flightcom, Lightspeed and other companies. A more sophisticated design that uses an adaptive controller is presented in Ref. [7].

Reduction of the noise and vibration from the engine by means of an active engine mount. The active mount reduces the vibration that is transmitted from the engine to the body of the car. An implementation of this system is described in Ref. [8].

Reduction of the noise generated in a ventilation duct due to a fan, using an active noise control system. An example of such a system can be found in Ref. [9].

The challenge is to make active noise and vibration control cost effective. Usually, this is only feasible if the solution can be integrated into an already existing system such as in a car audio system.
Active noise and vibration control is an active research field. Some examples of areas that are under investigation are.

Actively controlled panels, also known in literature as smart-panels. A control system is used to reduce the sound transmitted through the panel (See Refs. [10–12]). Possible applications can be found in buildings, trains, cars, aircrafts, etc.

Reduction of noise transmitted through a double-glazed window (See Refs. [13–15]).

This list not exhaustive, but it gives an impression of some active research areas. The European commission tried to promote this research by providing a research grant for the project Intelligent Material for Active Noise Reduction (InMAR) within the 6th Framework Programme. The work as described in this thesis was performed within this project. The goal was to research the applicability of intelligent materials for active noise reduction.

1.1.3 Passive noise and vibration reduction

Passive noise reduction can be realized by adding stiffness, mass or damping or by isolating parts of the structure, depending on the application at hand. One

approach is to decouple the structure from the vibrating environment using a mass-spring system, where the resonance frequency is sufficiently below the excitation frequency. In principle, it is possible to use these passive methods for any frequency. However, for low frequencies the weight of the structure can increase significantly. This makes it less suitable for certain application areas. In addition, the system may become rather undamped at the mass-spring resonance frequency, which can be reduced by active methods. The focus of this thesis is on active noise and vibration control, although it is often used in combination with passive means.

1.1.4 Active noise and vibration reduction

There is still a need for lightweight structures that either have good vibration reduction properties and/or good sound isolating properties. However, passive methods usually increase the weight and size of the structure and therefore maybe unsuitable. Active noise and vibration control may solve this problem. Active noise control can be realized by canceling the acoustic waves in the air, using loudspeakers and microphones. Active vibration control requires a sensor and an actuator that are bound to the structure.

In active noise and vibration control a controller is required that generates the control signals that reduces the vibration of the structure (AVC) or the sound pressure in the air (ANC). Such a system consists of a sensor, an actuator and a control system.

1.1.5 Short history of active noise control

The ventilation duct is a thoroughly studied object in the field of active noise control (See Ref. [9]). In this application it is often possible to measure a reference signal. A loudspeaker near the end of the duct generates the signal that is needed to cancel the sound at the end of the duct. An error microphone is needed in the case of adjustments of the control action (feedback or feedforward adaptive control). In most applications this microphone is placed near to or directly at the exhaust of the duct. It is necessary to have sufficient spatial separation between the reference microphone, the loudspeaker and the error microphone. Causality of the controller and near-field effects determine the minimum spatial separation.

The complexity of an active noise control system is governed by a three dimensional problem, which quickly becomes unsolvable for higher frequencies. If the noise is transmitted through the boundaries of the three dimensional acoustic space then a potentially more efficient control strategy is to control the sound radiation directly at the boundaries, which is a two dimensional configuration. This can be realized using acoustic actuators or structural actuators. The control of sound radiation from a boundary structure by modifying the vibration in this structure, such as a panel, resulted in a new research field which was termed Active Structural Acoustic Control (ASAC) [16]. An early application of this was

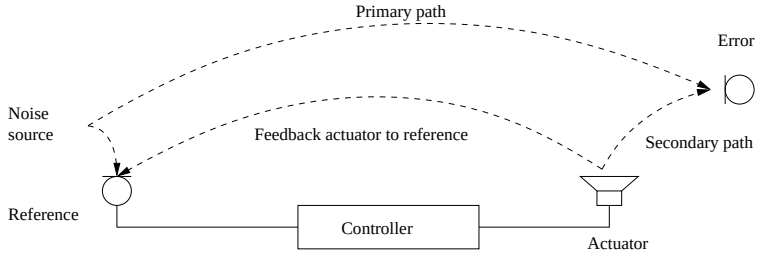
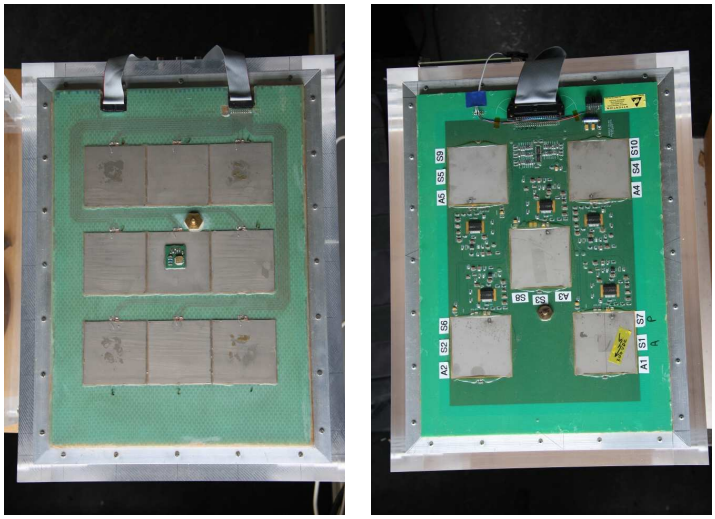


Figure 1.1: *Illustration of a simple active noise control problem using a feedforward approach.*

the actively controlled panel (Smart-panel), using piezoelectric patch actuators. First publications date back to the ending of the eighties/beginning of the nineties (See for instance Ref. [17]). In the middle of the nineties the term 'smart-panel' first emerged. Several definitions for a smart-panel can be found. Ideally a panel which only needs a power supply is meant. However, other authors are less strict and also include panels that are simple aluminum plates that have actuators and sensors mounted on them. All the signal condition electronics and the controller are separated from the panel. No, integration is applied in this case. In this thesis a more compact and partly integrated panel is described.

1.2 Problem definition

The general idea of active noise control is that a pressure wave can be canceled by a wave with opposite phase and the same amplitude. Early applications dealt with tonal noise. The same general idea of active noise control can be applied to broadband noise. In this thesis the implementations for a certain class of broadband noise is presented; only linear time invariant processes are considered. An overview of a simple feedforward active noise system can be found in Fig 1.1. In this figure, the disturbance source "Noise source" is on the left and the spot in which the reduction is required is on the right. The controller should generate a signal that reduces the noise level near the "error" sensor. In Figure 1.1, a number of important paths can be identified. The primary path is the path from the noise source to the error microphone. The secondary path is the path from the actuator to the error microphone. The path from the actuator to the reference sensor is an undesired path that potentially reduces the overall performance of the system if it is assumed that the reference signal is independent from the control system output. The controller needs sufficient time to calculate an appropriate signal for the actuator. This is realized by placing the reference sensor close to the noise source. It is also necessary to have sufficient spatial separation. That means that the time needed for the sound wave to propagate to the error sensor should be



(a) (b)

Figure 1.2: *First (a) and second (b) piezoelectrically driven panel.*

longer than the time needed for the controller to produce the output signal.

1.2.1 The smart-panel

In the InMAR project, the applicability of panels that are driven by piezoelectric patches was researched. In this thesis, the test object therefore is an actively controlled panel also known as a smart-panel, which was used to evaluate a rapid prototyping system. As stated before several realizations of a smart-panel are possible. The smart-panel used for the research presented in this thesis is a hybrid panel. This panel is a composition of several layers. The outer layers are made of epoxy-based printed circuit board (PCB). The inner layer is either a honeycomb structure or a foam based material. In the research performed for this thesis both materials were used. Both panels are lightweight and are not good at reducing the sound that is transmitted through them passively. This is especially true for lower frequencies.

Two types of panels were designed and tested. The first one is a simple panel that initially only contained piezoelectric patches; a picture of it can be found in Figure 1.2(a). These patches can be used as sensors as well as actuators. One side of the panel contains the actuators while the other side contains the sensors. In the setup used, the actuator and sensor are collocated, meaning that they are aligned but on the opposite side of the panel. In a later experiment, in which low-authority control [18] was used, external acceleration sensors were added. The second panel is a highly integrated panel; a picture of it can be found in Figure

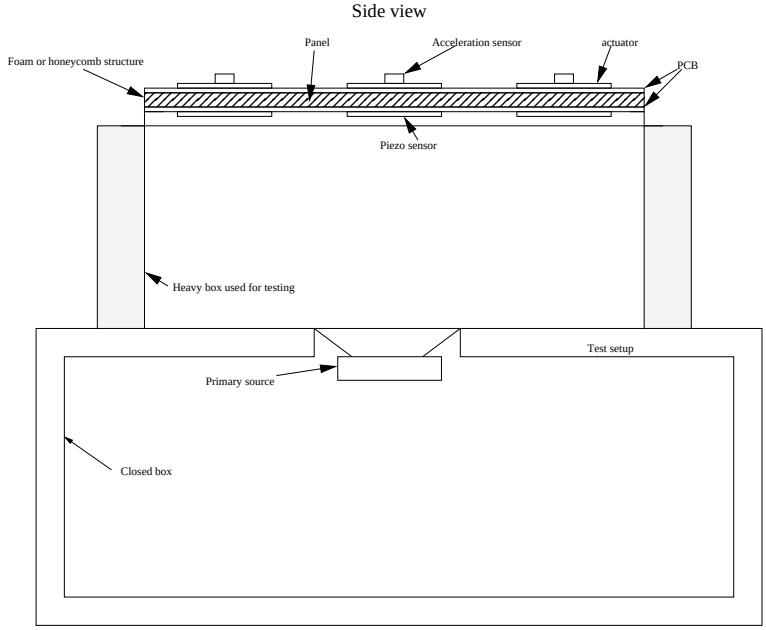


Figure 1.3: *Setup used for the Smart-panel experiments.*

1.2(b). In this panel all signal conditioning and power electronics was mounted on the PCB's that were used as the outer layers. However, no control electronics was mounted on the PCB. The controller is still a separate box. An overview of the panel can be found in Figure 1.3. The test setup consists of a perspex box. The box is made of 5 cm thick perspex. On the bottom of the box is a primary source loudspeaker. This loudspeaker is mounted inside the enclosed box. The test setup can be used for electronic feedforward as well as for feedback experiments.

1.2.2 The control architecture

In active noise and vibration control, two major control strategies exist. The first type is the feedforward controller, in which reference signals are measured to generate the required control signals. The second type is a feedback controller, in which the error signal are measured and directly used to generate the control signals. The feedback controller is useful in scenarios where a suitable reference signal can not be obtained. The design of a feedforward controllers depends on the statistical properties of the source. For statistical stationary sources an optimal Wiener filter is sufficient. In most systems the statistics may vary slowly over time, for instance the number of revolutions of an engine may change. This results in a controller that needs the following properties. It must be able to track statistical

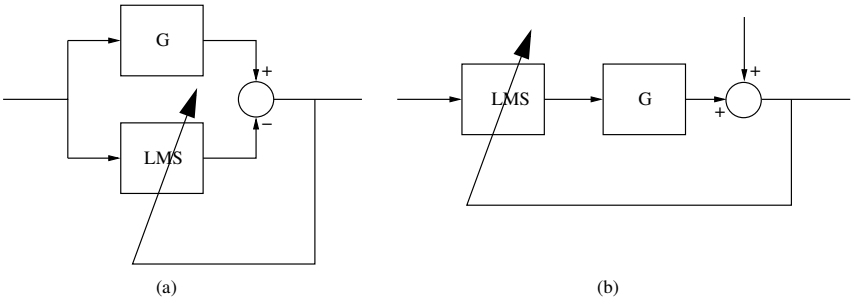


Figure 1.4: *LMS for system identification (a) or for active noise control (b)*

changes and it must be robust for parameter uncertainty. Adaptive filters have been suggested for this purpose [19]. The most commonly used adaptive feedforward controllers are based on the Least Mean Square method (LMS). In most textbooks (See [20]) the filtered reference (FxLMS) and the filtered error (FeLMS) algorithms are used. These algorithms are model-based controllers and therefore require a model of the plant. This model can be derived using an identification method such an LMS-filter or a sub-space method [21]. It is possible to translate a feedforward controller into a feedback controller using internal model control (IMC) (See the textbook by Elliott [20]).

1.3 Literature overview

The literature overview is divided into four sections. The first section gives a short introduction on the history of the adaptive filters. The focus is on the FeLMS and FxLMS algorithms and most of its derivatives. In the second section a quick introduction into higher-order substitutes for the least mean square algorithm is given. In the third part, different control architectures are described. Finally, in the fourth part, a short introduction into the history of the smart-panel is given.

1.3.1 Adaptive algorithms

In 1959, B. Widrow published an article which in later years became quite famous. In this article the theory for the so-called least mean square (LMS) algorithm was formulated. The LMS algorithm has been extensively used in for instance echo cancellation for telephone lines. In this case a filter is estimated that is parallel to the adaptive filter. This is shown in Figure 1.4a. However, for active noise cancellation a modification is required. In this case the output of the adaptive filter is concatenated to the secondary transfer function of the plant G. This is illustrated in Figure 1.4b. The LMS filter does not converge in such a setup, if there is no correction for the secondary path G. In 1980, D.R Morgan [22] published a paper

that proposed a solution to this problem. The algorithm presented in this paper was not with active noise control in mind, but it is an early reference to FxLMS. In most active noise cancellation problems one single actuator-sensor pair is not sufficient to get good results. The ANC example implemented in the propeller aircrafts requires several loudspeakers and error microphones. This means that the FxLMS algorithm must be expanded for MIMO systems. In 1987, Elliott, Stothers and Nelson [23] published a paper that addressed this problem. A limitation of the FxLMS algorithm is its computational complexity. It scales with number of reference sensors, error sensors and the number of actuators and the adaptive filter length. In 1996, E. Wan [24] proposed an optimization for this problem. He called it the adjoint LMS algorithm. The basic idea is to perform the filtering with the plant model in the adaptation loop. In this case the complexity scales with the number of error sensors, the number of actuators and the filter length. This makes the adjoint LMS algorithm more suitable for systems with a high number of reference signals than the FxLMS algorithm. In the literature the adjoint LMS algorithms is also called filtered-error LMS algorithm (FeLMS). The FeLMS algorithm uses a time-reversed transpose plant model in the error-sensor loop. This makes the system non-causal. By prefixing the plant model with a delay the overall system is made causal. This also requires an extra delay in the reference signals needed for the update rule. In practice this delay reduces the maximum step size of the FeLMS algorithm, reducing the convergence speed. A method that tries to reduce the delay in the adaptation loop is the hybrid-filtered error LMS algorithm (HFeLMS) (See [25]). The convergence speed of the FeLMS algorithm is also reduced due to non-white reference signals, cross-correlation between reference signal, frequency dependency of the secondary path and cross-correlation between the transfer functions of the secondary path. In the textbook by S. Elliott [20] an algorithm was introduced that reduces these influences. This algorithm was called the preconditioned filtered-error algorithm (PLMS). This was realized by whitening the reference signals and removing the cross-correlation and frequency dependence of the secondary path. The reference signals are made white by the inverse filter of the coloring process. The influence on the secondary path can be reduced by filtering the output of the controller with the inverse of the minimum-phase plant function, which makes it necessary to filter the adaptation loop with the time-reversed transpose all-pass part of the plant function. A single channel minimum-phase/all-pass decomposition was presented in [26]. In this paper also a whitening filter was tested, however each reference input was filtered separately, which still results in correlated, but white reference signals. A method for decorrelating the reference signals is given in Ref. [27]. Another method that also introduced a pre-whitening filter and a decomposition of the secondary path, but now in the frequency domain is presented in Ref. [28]. In the algorithms presented so far the convergence speed is reduced by the inherent delay in the adaptation loop. In 2007, an algorithm was presented that removes the negative influence of this delay. This algorithm basically uses the same minimum-phase/all-pass decomposition as the PLMS algorithm, however it uses a numerical method which

is called inner-outer factorization. A delay-compensation technique and double control filters are used to further reduce the inherent delay in the adaptation loop. This algorithm still suffers from poor stability if the plant has zeros. A solution for this was found in a regularization technique that preserves the factorization properties, but limits the output of the inverse plant model. The combination of the delay compensation technique and the regularization method was called the regularized modified filtered error algorithm (RMFeLMS) [29].

A limitation of the algorithms presented so far is the that they require an off-line identification of the secondary plant. If the plant changes due to parametric uncertainty this results in plant mismatch. This is especially problematic for lightly damped resonances. In this case a method can be used that estimates the plant on-line, resulting in full adaptive control. The plant must be estimated parallel to the secondary path. In Chapter 3 of the textbook by Elliott [20] such an algorithm is described. In this case it is necessary to inject identification noise into the secondary path, which is uncorrelated to the system noise. The estimated plant is then copied into the secondary plant model. This architecture can be used for almost all adaptive control schemes. The necessity to inject noise into the system is one of the major disadvantages of on-line identification [30]. The injected noise is audible, and, if the noise-level is low it results in slow convergence. This slow convergence, caused by low-noise levels, results in bad tracking behavior for rapidly changing plants.

1.3.2 High order adaptive algorithms

The LMS algorithm is extensively studied, it is robust and has a low computational complexity. However, it suffers from degraded performance when the reference signal is colored, resulting in an autocorrelation matrix with a large eigen value spread. This in turn results in a degraded convergence rate. Higher-order methods can improve the convergence rate. The Newton iteration is a higher-order method, however it suffers from the problem that the autocorrelation matrix needs to be estimated. Calculating and estimating this matrix and then taking the inverse can be challenging. Another higher-order method is known as the recursive least squares (RLS) algorithm. The robustness and the computational complexity are the two major limitations of the RLS algorithm. In this algorithm, it is again necessary to calculate the inverse autocorrelation matrix. However, in this algorithm the inverse autocorrelation matrix is calculated iteratively using the matrix inversion lemma. The RLS algorithm still has complexity that has an order of $O(l^2)$ in which l is the filter length. In the past, symmetry and redundancy in the update rule was used to implement a fast version of the RLS algorithm called FRLS. The complexity of this algorithm is again $O(l)$. A method in which the complexity can be adjusted is the so called affine projection (AP) algorithm (See [31]). The order of this algorithm can be adjusted to the need of the problem at hand. This order directly influences the size of the autocorrelation matrix. The Fast Affine Projection (FAP) algorithms were introduced in Refs. [32, 33]. The

FAP and FRLS algorithms both use fast transversal filters to find an estimate for the inverse autocorrelation matrix. However, this method suffers from poor numerical stability. Many authors therefore use another method for finding the inverse autocorrelation matrix. In the paper by Bouchard [34], the iterative Gauss-Seidel algorithm is used to approximate the inverse autocorrelation matrix. This paper also introduces a multi-channel filtered-reference algorithm using FAP and AP. Heping [35] proposed to use the conjugate gradient method for solving the inverse matrix. In a paper by Oh [36], it is assumed that the filter length is considerably longer than the affine projection order and therefore the matrix can be considered Toeplitz. Heping [37] gives an overview of the most widely used algorithms. In this paper the complexity of these algorithms is analyzed and a method is introduced that used the LDL^T decomposition to find the inverse autocorrelation matrix. In a paper by Douglas [38] a fast approximate implementations of the FAP-algorithm is given. In 1998, Rupp [39] showed that the AP/FAP algorithm has the intrinsic property that reference signals that are colored by an autoregressive process are de-colored. In the same paper, a modification was presented that also works for reference signals that are colored by a moving average process or a moving-average autoregressive process.

1.3.3 Control architectures

The feedback and feedforward control architecture can be realized as a simple single-input single-output (SISO) system. Such a simple controller, however may result in poor performance in the case of a complex dynamic system such an actively controlled panel (Smart-panel) (See ref. [17]). In this case an architecture is required that uses several actuators and sensors to realize a good performing control system. Such a system can be realized as a multiple-input multi-output (MIMO) system or as multiple independent SISO controllers or as a distributed controller. In the case of a MIMO based system, which is known as a centralized control architecture, the influence of different sensor-actuator pairs on each other is taken into account. However, a MIMO based architecture may suffer from two major constraints. Firstly, the computational complexity can rise too high when the number of channels increases, making the system unfeasible. Secondly, stability and robustness can become problematic for a complex model-based controller, due to the inherent model mismatch, which is caused by non-linearity and parametric uncertainty. Furthermore, if one part of the system fails, it may lead to failure of the complete system. There are many control design methods that take the parametric uncertainty into account and try to design a robust controller (See ref. [40]). In the case of multiple SISO-based controllers, which is known as decentralized control, the loops may influence each other and destabilize the overall system. However, this can be circumvented if a collocated and dual sensor-actuator pair is used in a feedback configuration, reducing the overall energy in the system (See ref. [18]). A SISO system only requires moderate computational complexity.

The decentralized algorithm operates as a local control strategy on a sensor-actuator pair. It is possible to use this architecture for feedforward as well as feedback control, however in this section the focus lies on feedback control. In the ideal case a sensor-actuator pair is collocated and dual. A collocated dual pair extracts energy from the system when negative feedback is used, reducing the overall stored energy in the system [18]. A consequence of this is that the pair is stable for an arbitrarily large gain. The reason for this is that a collocated and dual pair always has a phase shift between -90 and +90 degrees making it possible for the loop gain to be arbitrarily large. Then, the transfer function of the open loop will never enclose the -1 point in the Nyquist diagram: the locus is in the right side of the Nyquist plot. A restriction of this approach is that the system is not damped for very large gains. Instead, the response becomes pinned at the location of the sensor actuator pair (See Ref. [10, 18]) introducing new modes undamped modes. But the idea can still be used to introduce damping into the structure if the gain is not too large. An example of a collocated dual pair is a force actuator combined with a velocity sensor. The velocity signal can be derived by a simple integration of an acceleration signal. A further consequence of this approach is that it is possible to use multiple independent feedback loops. The fact that the overall stored energy in the system is reduced ensures stability for such multiple SISO systems. This is explained in the textbook by Preumont [18].

The centralized approach uses several sensors and actuators, where the number of sensors can be different from the number of actuators. A model based controller is used to control the overall system. It is possible to design such a controller using classical and modern design methods (See ref. [41]). As stated before, IMC can also be used to translate a feedforward controller into a feedback controller. This translates the feedback controller into a predictive feedforward filter. Such a filter behaves as a whitening filter if the delay is one sample. In the textbook by Elliott (Ref [20]) it is shown that a predictive feedforward filter only works well for colored input signals. A feedback controller obeys the Bode sensitivity integral. In the case of a non-minimum phase plant, if a peak at a frequency is reduced it must lead to increase at other frequencies (See [20]), which is known as the water-bed effect.

In a centralized or decentralized structure, an interesting question arises what happens if both methods were to be combined. Preumont (Ref. [18]) describes a high-authority and low-authority (HAC/LAC) control architecture. In this approach an architecture is used in which the LAC architecture consists of a collocated sensor-actuator pair and simple decentralized analog feedback controller. The HAC architecture is designed using a model-based controller which uses for instance a linear quadratic regulator (LQR). The use of a HAC/LAC architecture yields three major advantages [18]. Firstly, the active damping extends outside the bandwidth of the HAC control loop, which reduces the settling times outside the control bandwidth. Secondly, it is easier to gain-stabilize the modes outside the bandwidth of the outer loop. And thirdly the large damping of the modes inside the controller bandwidth makes them more robust to parametric uncer-

tainty. In the paper by Herold, Mayer and Hanselka a method using piezoelectric sensors and actuators and positive position feedback (PPF) was described (See [42]). The principle of PPF is described in the book by Preumont. In this configuration a second-order filter is used as the control filter which is combined with positive feedback. The control filter is then tuned to reduce one of the desired resonance peaks. This is realized by setting the resonance frequency equal to the filter frequency. A network of multiple parallel second-order filters is required when multiple resonance peaks need to be reduced.

1.3.4 The smart-panel

In 1991, Wang, Fuller and Dimitriadis published a paper (Ref [17]) that first introduced a panel in which a piezoelectric patch actuator was used to reduce the sound transmitted through the panel. In this paper, the location of the actuator(s) and the type of actuator was studied. It was concluded: 1) that a single actuator is not sufficient, 2) point force actuators yield slightly better results than a piezoelectric patch actuators, 3) if the number of actuators increases the reduction also increases. From this, it can be concluded that the panel must have as many actuators as feasible which should behave as point force actuators. The piezoelectric patch actuator is a suitable choice for an actively controlled panel due to its small form factor. A piezoelectric patch actuator does not exert force, which reduces the robustness of the control-system. The influence of the sensor was also studied extensively in many papers (for example the paper by Sors and Elliott from 1999 [10]. In this paper, a setup with 5 error sensors and 5 actuators in a collocated setup was proposed. An experiment using a single channel feedback controller with IMC was carried out. It appeared to be not possible to investigate the 5 channel setup at that time due to computational limitations. In a simulation study, different actuator and sensor combinations were compared. The sensor/actuator combinations compared included the point force actuator combined with an acceleration sensor and the piezoelectric sensor combined with a piezoelectric actuator.

Another research area concentrates on the radiation modes of a vibrating plate. Normally the error signal is measured at the sensor position. But minimizing this error signal does not guarantee a reduction of the noise in the far field. A method of circumventing this problem is to incorporate the radiation mode approach into the error signal. The idea is to weight the error signal with a function that approximates the radiation behavior of the panel, taking into account the frequency dependence of the radiation modes. Early papers are by Borgiotti [43] and Elliott and Johnson [44].

A model based MIMO controller needed for a 5 channel setup can be become quite complex. To overcome this problem many solutions have been studied including decentralized MIMO systems. This is possible by means of a very carefully designed robust SISO controller or by using a dual and collocated sensor actuator pair. One such a setup consists of a collocated acceleration sensor and inertial actuator pair. In this setup the inertial actuator generates a point force and the

acceleration sensor measures velocity which is obtained after integration. The problem with this setup is that an inertial actuator is relatively heavy, making such a setup unpractical [45]. It is, however, possible to use a piezoelectric actuator instead. In the publication by Gardonio, Bianchi and Elliott (Ref. [46–48]) this approach was demonstrated and measurement and extensive simulations of such a setup were presented. The use of a piezoelectric actuator limits the gain in the setup. For small gains, the controller only adds damping to the structure. For large-gain feedback, the plate is fixed at the sensor actuator position, which makes the panel transparent in an acoustical sense for other frequencies. This was also concluded in the paper by Sors and Elliott (Ref. [10]).

1.4 The research work

1.4.1 The research question

The main focus of this thesis is on active noise and vibration control. The basic research question is: "**To design a multi-channel active noise and vibration control system. It should be suited for different applications including mobile demonstrators such as in cars, it should provide a rapid prototyping environment and it should provide a possibility for stand-alone operation.**" This ambitious goal can be broken down in several parts.

What kind of platform is needed to quickly evaluate different algorithms using a rapid prototyping environment?

What kind of control algorithm is needed? FxLMS, FeLMS, RMFeLMS, robust control such as H_∞ and so on.

How should the architecture of the controller look like? Should the controller be centralized or decentralized. Is it necessary to use a HAC/LAC architecture?

1.4.2 The contributions

In this thesis the algorithm introduced by Berkhoff and Nijsse [49] plays a central role. The algorithm was implemented and extensively tested and verified by means of simulation and measurements on a test setup. A method that predicts the expected performance was derived. A method that improves the convergence speed for highly colored signals was added. Finally, the system was expanded with a HAC/LAC method. The steps were as follows:

An appropriate algorithm was selected. The RMFeLMS algorithm was found to be the suitable algorithm in view of the application considered. It was implemented on a realtime platform and its performance was evaluated. The result of these measurements were compared to simulations that used the secondary and primary path models.

A method was derived that is able to predict the theoretical performance of the RMFeLMS algorithm. It uses a finite impulse response (FIR) realization of the Wiener filter estimation that takes into account the regularization of the RMFe structure.

The RMFe algorithm was expanded with an affine projection method. This improves the convergence rate for strongly colored reference signals, as present in systems with many resonance modes, such as in duct-like structures.

A HAC/LAC architecture was implemented. In this case the LAC architecture was realized using a high-speed digital controller. The controller was realized using 3 concatenated infinite impulse response (IIR) 2^{nd} order filters. It is shown in this thesis that this improves the robustness and also the overall system performance.

From a practical standpoint several goals were realized. A rapid prototyping system was developed which has been used in different experiments and applications such as described in Refs. : [50–53]. The basic code generation system as provided by TNO was extensively modified. The list of extensions is as follows:

A system was designed and implemented on which it is possible to quickly evaluate different algorithms. This system consists of a realtime operating system (OS), a rapid prototyping interface using Matlab/Simulink and an interface board that performs the analog to digital (AD) and digital to analog (DA) conversion (ADDA). The rapid prototyping software is a commercially available product from the MathWorks. The ADDA card and the necessary interfacing were designed during the research period. The board uses a field programmable gate array (FPGA) to implement all interface logic.

All the logic necessary to perform local high-speed signal processing on the FPGA was implemented. The sample rate of the FPGA is approximately 100 kHz. It is possible to select a number of fixed sample rates, where the maximum sample rate is 16 kHz and the minimum sample rate is 250 Hz, with a typical sample rate of 2 kHz. This was realized by adding an interpolator and decimator circuit to the FPGA. A high speed local feedback path, consisting of 3 IIR 2^{th} order-filters, was added. Furthermore, it is also possible to use an IIR filter at the output of the interpolator which can be used for frequency response correction, of e.g. a loudspeaker.

A special digital IO unit was added to the FPGA logic. This unit has flank detection logic and is able to measure period times with a resolution of 30 ns. This makes it possible to realize a tonal control system using a software-based phase locked loop (PLL) and a period counter.

1.5 Thesis outline

The system was tested extensively on the actively controlled panel (smart-panel). The goal was to realize and perform measurements on a practical Smart-panel and compare them with simulations. Using different setups, this panel and the algorithm were tested and verified. The algorithm was then extended with a method to improve the convergence speed for strongly colored reference signals. The HAC/LAC method was also used to improve performance and robustness. A major goal has been to use part of this research in real applications. For this reason two chapters are dedicated to the controller and the software necessary to realize such a system. The chapters are broken down as follows:

Chapter 2 introduces two commonly used adaptive algorithms for active noise control. In this chapter the advantages and disadvantages are summarized.

Chapter 3 presents results of implementation of the regularized modified filtered error algorithm (RMFe) [29]. It improves on the algorithms which are described in Chapter 2. This algorithm uses a state-space based model of the plant reducing the computational load in the case of MIMO systems. A major advantage of this algorithm is that it scales better with respect to an increased sample rate in the case of MIMO-systems when compared to the algorithms presented in Chapter 2, which use a FIR-based model of the plant. The RMFe algorithm works for feedforward as well as for feedback control, when the latter is combined with IMC. In this chapter, an extension is introduced that improves the convergence for highly colored reference signals in the case of feedforward control. Finally, measurement results are presented and evaluated.

Chapter 4 introduces a system in which a LAC and HAC controller are combined to get a hybrid control system. In this control system, a feedforward and a feedback method are used to get a more robust and overall better performing system. A special high sample rate local feedback loop was used to realize this system.

Chapter 5 discusses hardware architectures optimized for active noise control. In this chapter, also the impact of latency is addressed. The hardware architecture as designed and implemented during this project is described. This architecture is capable of handling up to 16 inputs and 16 outputs. The design trade-offs and limitations are explained.

Chapter 6 discusses the realtime platform used to actually implement the RMFe algorithm. This chapter addresses the realtime software needed to interface with the hardware platform.

In Chapter 7 the conclusions are given.

Chapter 2

Algorithms for active noise and vibration control

2.1 Introduction

In active noise control several different control strategies are possible. In most applications an adaptive controller is required, due to the non-stationary statistics. Adaptive controllers are commonly based on the least mean square (LMS) principle, due to its low computational complexity. But it is also possible to use higher order algorithms such as the recursive least square (RLS) and the affine projection (AP) methods (See refs. [54, 55]). Other methods are for instance based on a Kalman filter in combination with a state estimator (See [40]). In the first section the optimal Wiener filter is derived. The optimal Wiener filter is used as a reference to compare other solutions. A new control structure should have similar performance characteristics as this theoretical optimum. In the next two sections the filtered reference and filtered error algorithms are described. The last two sections describe internal model control (IMC), which can be used to reduce the influence of feedback from the control output to the reference input. It is also possible to use this technique to translate a feedforward controller into a feedback controller.

2.2 The optimal Wiener filter solution

The Wiener filter is used to derive an estimate of the theoretical optimal performance. The basic structure for a Wiener filter can be found in Figure 2.1. The method used to derive the Wiener filter is similar to that used in the book of Elliott [20]. This method makes it possible to derive the filtered reference least mean square algorithm (FxLMS) as an extension. The sample moment will be denoted as

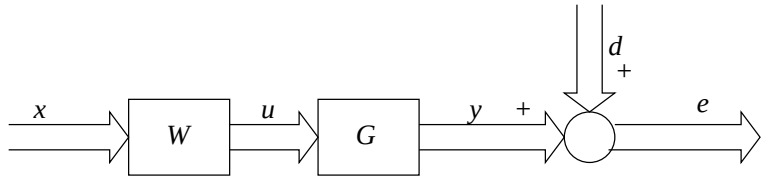


Figure 2.1: *Wiener filter with the secondary plant included.*

n. The signals are defined as $x(n) = [x_1(n) \dots x_K(n)]^T$, $d(n) = [d_1(n) \dots d_L(n)]^T$, $e(n) = [e_1(n) \dots e_L]^T$ and finally $u(n) = [u_1(n) \dots u_M(n)]^T$. The expression will be derived using convolution. Define the error as

$$e_l(n) = d_l(n) + \sum_{m=1}^{M} \sum_{j=0}^{J-1} g_{lmj} u_m(n-j), \tag{2.1}$$

in which g_{lmj} is J-th impulse response coefficient from the the M-th actuator to the L-th error sensor. The input to the plant $u(n)$ can be defined as:

$$u_m(n) = \sum_{k=1}^{K} \sum_{i=0}^{I-1} w_{mki} x_k(n-i), \tag{2.2}$$

which results in the following overall equation

$$e_l(n) = d_l(n) + \sum_{m=1}^{M} \sum_{j=0}^{J-1} \sum_{k=1}^{K} \sum_{i=0}^{I-1} g_{lmj} w_{mki} x_k(n-i-j). \tag{2.3}$$

Define the filtered reference signal r_{lmk} as

$$r_{lmk}(n) = \sum_{j=0}^{J-1} g_{lmj} x_k(n-j), \tag{2.4}$$

which in turn simplifies equation (2.3) to

$$e_l(n) = d_l(n) + \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{i=0}^{I-1} w_{mki} r_{lmk}(n-i). \tag{2.5}$$

The triple sum can be rewritten as a sum of a vector products

$$e_l(n) = d_l(n) + \sum_{i=0}^{I-1} \underline{w}_i^T \underline{r}_l(n-i). \tag{2.6}$$

where

$$\underline{w}_i = [\ w_{11i} \quad w_{12i} \ldots w_{1Ki} \quad w_{21i} \ldots w_{MKi}]^T\ , \tag{2.7}$$

and

$$\underline{r}_l(n) = [\ r_{l11}(n) \quad r_{l12}(n) \ldots r_{l1K}(n) \quad r_{l21}(n) \ldots r_{lMK}(n)\ ]^T. \tag{2.8}$$

The error criteria can now be written as

$$e(n) = d(n) + R(n)w, \tag{2.9}$$

with

$$R(n) = \begin{bmatrix} \underline{r}_1^T(n) & \ldots & \underline{r}_1^T(n-I+1) \\ \vdots & & \\ & \ddots & \vdots \\ \underline{r}_L^T(n) & \ldots & \underline{r}_L^T(n-I+1) \end{bmatrix}, \tag{2.10}$$

and

$$w = [\ \underline{w}_0^T \quad \ldots \quad \underline{w}_{I-1}^T\ ]^T. \tag{2.11}$$

Define the quadratic cost criteria as

$$J = E\left[e^T(n)e(n)\right] = tr.E\left[e(n)e^T(n)\right] = E\left[tr.\left\{e(n)e^T(n)\right\}\right], \tag{2.12}$$

with the property of the expectation operator that $E(x+y) = E(x) + E(y)$. Minimizing Eq. (2.12) by taking the derivative and setting it to zero will result in the solution for the optimal Wiener filter. The cost criteria can be written as

$$J = E\left[tr.\left\{(d(n) + R(n)w)(d(n) + R(n)w)^T\right\}\right], \tag{2.13}$$

which is equal to

$$\begin{aligned} J \quad = \quad & E\left[tr.\left\{d(n)d^T(n)\right\}\right] + E\left[tr.\left\{d(n)w^TR^T(n)\right\}\right] + \\ & E\left[tr.\left\{R(n)wd^T(n)\right\}\right] + E\left[tr.\left\{R(n)ww^TR^T(n)\right\}\right]. \end{aligned} \tag{2.14}$$

Eq. (2.14) can be simplified, using the properties of the trace operator. Set $A = d(n)w^T$ and $B = R$ and notice that both A and B are of the dimension $L \times MKI$. It can be shown that the trace operator has the property that $tr.\{AB^T\} = tr.\{B^TA\} = tr.\{A^TB\} = tr.\{BA^T\}$, when A and B have the same dimension. These properties can be used to show that $tr.\left\{d(n)w^TR^T(n)\right\} = tr.\left\{R^T(n)d(n)w^T\right\}$ and $tr.\left\{R(n)wd^T(n)\right\} = tr.\left\{R^T(n)d(n)w^T\right\}$. Using the same trace properties it can be shown that $tr.\left\{R(n)ww^TR^T(n)\right\} = tr.\left\{w^TR^T(n)R(n)w\right\}$. This will result in the reduced form of Eq. (2.14)

$$\begin{aligned} J \quad = \quad & tr.\left\{E\left[d(n)d^T(n)\right]\right\} + 2tr.\left\{E\left[R^T(n)d(n)\right]w^T\right\} + \\ & tr.\left\{w^TE\left[R^T(n)R(n)\right]w\right\}. \end{aligned} \tag{2.15}$$

The trace operator has some nice properties with respect to taking the derivate. The following equations can be shown to be valid (See [20])

$$\frac{\partial tr.\left(BA^T\right)}{\partial A} = B, \tag{2.16}$$

and

$$\frac{\partial tr.\left(A^TBA\right)}{\partial A} = (B + B^T)A. \tag{2.17}$$

The cost function J in Eq. (2.15) has a unique global solution, when $E\left[R^T(n)R(n)\right]$ is positive definite. This is the case when the inputs are sufficiently persistently excited. This unique global solution can be found by taking the derivative with respect to w and setting it to zero $\frac{\partial J}{\partial w} = 0$. The derivative can be found by setting $A = w$ and $B = E[R^T(n)d(n)]$ in Eq. (2.16) and $A = w$ and $B = E[R^T(n)R(n)]$ in Eq. (2.17). Also notice that $E[R^T(n)R(n)]$ is symmetric so that $B = B^T$, which results in

$$\frac{\partial J}{\partial w} = 2E\left[R^T(n)R(n)\right]w + 2E\left[R^T(n)d(n)\right] = 0, \tag{2.18}$$

leading to the optimum solution,

$$w_{opt} = -E\left[R^T(n)R(n)\right]^{-1} E\left[R^T(n)d(n)\right]. \tag{2.19}$$

The optimal Wiener filter can be found by using equation (2.19). However, the inverse matrix is very large ($MKI \times MKI$). This matrix is in the block Toeplitz structure and can be solved quite efficiently (Ref. [56]).

2.3 The filtered reference algorithm

In active noise control an commonly used algorithm is the filtered reference least mean square algorithm (FxLMS). The derivation of the multiple-input multiple-output (MIMO) FxLMS algorithm is analog to the derivation of a single-input single-output (SISO) FxLMS algorithm. The error for a system in which the filter coefficients are modified in each time step can be expressed as

$$e(n) = d(n) + R(n)w(n). \tag{2.20}$$

The expectation operator in Equation (2.15) will be dropped and it is assumed that the filters coefficients are dependent on time. This results in taking the instantaneous value as a rough estimate for the expectation value resulting in

$$\begin{aligned} J \;=\; & tr.\left\{e(n)e^T(n)\right\} = tr.\left\{w^T(n)R^T(n)R(n)w(n)\right\} + \\ & 2tr.\left\{R^T(n)d(n)w^T\right\} + tr.\left\{d^T(n)d(n)\right\}. \end{aligned} \tag{2.21}$$

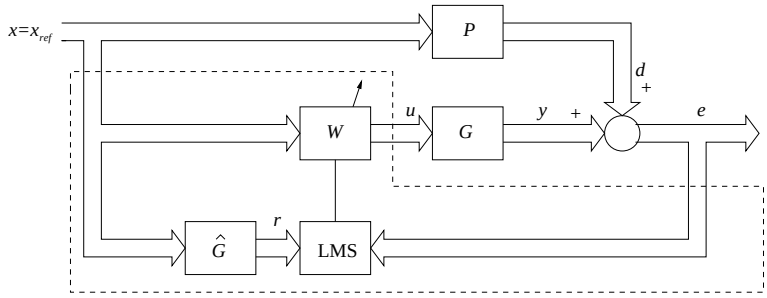


Figure 2.2: *Multiple input and multiple output filtered reference block diagram.*

The idea is to find an iterative solution that minimizes the cost function. To realize this the derivative of the cost criteria with respect to the filter coefficients $w(n)$ will be taken. The trace operator properties of Eqs. (2.16) and (2.17) will be used ton find the derivative of the cost criteria. Which results in

$$\frac{\partial tr.\left\{e(n)e^T(n)\right\}}{\partial w} = 2R^T(n)R(n)w + 2R^T(n)d(n), \tag{2.22}$$

in which Eq. (2.20) is substituted, resulting in

$$\frac{\partial tr.\left\{e(n)e^T(n)\right\}}{\partial w} = 2R^T(n)e(n). \tag{2.23}$$

The stochastic gradient algorithm is used to find an iterative solution for $w(n)$, which result in

$$w(n+1) = w(n) - \alpha\frac{\partial tr.\left\{e(n)e^T(n)\right\}}{\partial w(n)}, \tag{2.24}$$

resulting in

$$w(n+1) = w(n) - \alpha R^T(n)e(n). \tag{2.25}$$

In a real application the plant needs to estimated $\hat{R}(n)$. A disadvantage of this algorithm is that there is no form of normalization. The easiest way to do this by making sure that the filter coefficients can not become arbitrary large. This is realized by expanding the cost criteria with a weighting of the filter coefficients resulting in

$$J = tr.\left\{e(n)e^T(n)\right\} + \gamma tr.\left\{w(n)w^T(n)\right\}, \tag{2.26}$$

it can be shown that this results in the following update rule

$$w(n+1) = (1-\alpha\gamma)w(n) + \alpha\hat{R}^T(n)e(n). \tag{2.27}$$

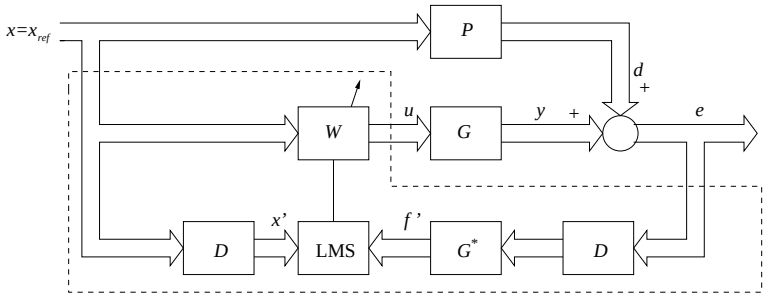


Figure 2.3: *The filtered error lms algorithm.*

2.4 The filtered error algorithm

The filtered error algorithm can be derived when the optimal Wiener filter solution will be expanded into a sums of matrix impulse responses. This derivation is well documented in literature and will not be repeated here for brevity. This derivation starts with the following error

$$e(n) = d(n) + \sum_{j=0}^{J-1} \sum_{i=0}^{I-1} G_j W_i x(n-i-j) \tag{2.28}$$

in which G_j is the $L \times M$ matrix of the j-th coefficient of the plant impulse response and W_i is the matrix $M \times K$ of i-th coefficient of the impulse response of the filter. In chapter 5 of the textbook by Elliott [20] it is shown that the derivate for the cost criteria is equal to

$$\frac{\partial J}{\partial W_i} = 2 \sum_{j=0}^{J-1} G_j^T R_{xe}(i+j), \tag{2.29}$$

with

$$R_{xe}(m) = E\left[e(n+m)x^T(n)\right]. \tag{2.30}$$

Start by substituting Eq. (2.30) into Eq. (2.29) resulting in the following equation

$$\frac{\partial J}{\partial W_i} = 2E\left[\sum_{j=0}^{J-1} G_j^T e(n+j)x^T(n-i)\right]. \tag{2.31}$$

The vector of M filtered error signals is written as

$$f(n) = \sum_{j=0}^{J-1} G_j^T e(n+j), \tag{2.32}$$

which results in

$$\frac{\partial J}{\partial W_i} = 2E\left[f(n)x^T(n-i)\right]. \tag{2.33}$$

This final equations can be used to derive the filtered error algorithm. Take the instantaneous value of the derivative and implement it as a simple update rule. This result in

$$W_i(n+1) = W_i(n) - \alpha f(n)x^T(n-i), \tag{2.34}$$

which is known as the adjoint LMS-algorithm. This update rule, however is not causal due to the fact that the filtered signal $f(n)$ requires a time advanced error signal. To make the update rule causal a shift over J-1 samples is needed for the error signal $e(n)$ as well as the reference signal $x(n)$. The update rule needs to be written as

$$W_i(n+1) = W_i(n) - \alpha f(n-J+1)x^T(n-i-J+1), \tag{2.35}$$

with the filtered error represented as

$$f(n-(J-1)) = \sum_{j'=0}^{J-1} G^T_{(J-1)-j'} e(n-j'), \tag{2.36}$$

with $j = (J-1)-j'$. The computational complexity of the filtered error algorithm is better than that of the filtered reference algorithm in the case of many reference signals. The the converge rate of the FeLMS algorithm, however is less than that of the FxLMS algorithm. The main cause of this can be found in the extra delay needed to make the algorithm causal. This delay makes it necessary to increase the step size to prevent instability. In Figure 2.3 an overview of the filtered error algorithm is given.

2.5 Internal model control

In some applications feedback exists from the output of the controller to the reference inputs. This feedback can lead to a reduction in performance of the controller. A method that reduces the impact of this feedback path is called internal model control (IMC). In this case the influence of the undesired-feedback path is subtracted from the reference signals using a model of this path [20]. This method requires an estimated model that described the path from the actuator to the reference sensor. In Fig 2.4 it can be seen how to implement the internal model control system. Internal model control can also be used to translate a feedforward controller into a feedback controller

2.6 Feedback using IMC

The feedforward controller is the preferable method to implement an active noise and vibration system. However, it is not always possible to measure a reference

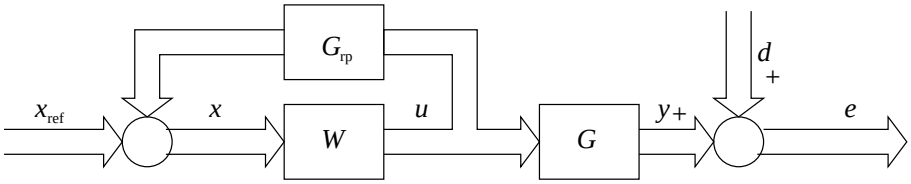


Figure 2.4: *Internal model control to reduce the feedback from the actuator to the reference sensor*

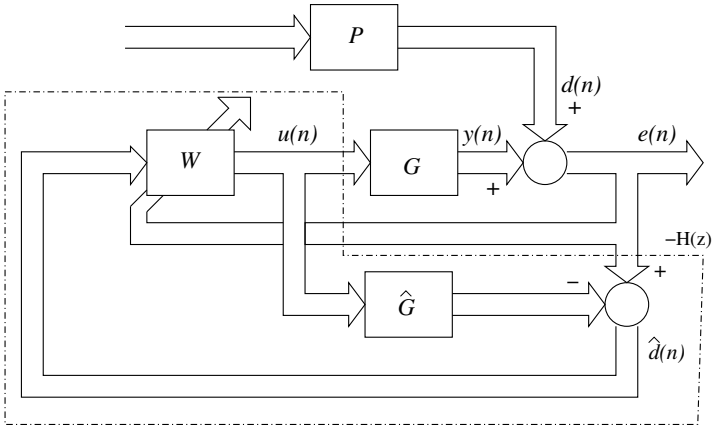


Figure 2.5: *Internal model control and feedback combined.*

signal. In this case a feedback controller is required. A feedback controller can be designed using classical or modern control strategies such as instance H_∞, LQR combined with a Kalman filter and so on. These are fixed controllers. However, an adaptive controller is preferred, due to slowly-varying statistical properties. The feedforward controllers presented in this Chapter are adaptive controller. A feedforward controller can be translated into a feedback controller using internal model control. In this case the control signal is filtered using an estimate of the secondary plant and then subtracting from the measured error signals resulting in a estimated disturbance signal. The schematic view for IMC can be found in Fig. 2.5. From Figure 2.5 a simple feedback controller $H(z)$ can be derived

$$H(z) = \frac{-W(z)}{1 + \hat{G}W(z)}, \tag{2.37}$$

resulting in the overall transfer from the disturbance to the error as

$$S(z) = \frac{E(z)}{D(z)} = \frac{1 + \hat{G}(z)W(z)}{1 - [G(z) - \hat{G}(z)]W(z)}. \tag{2.38}$$

If the model matches the real plant $\hat{G}(z) = G(z)$ then it can be written as

$$S(z) = 1 + G(z)W(z), \tag{2.39}$$

from which it can be concluded that the feedback controller becomes an optimal prediction filter. The overall filter works as a whitening filter, it de-colors the noise signal, making the spectrum flat.

Chapter 3

The regularized modified filtered error algorithm

3.1 Introduction

Many algorithms used for broadband active noise control are based on the adaptive Least-Mean-Square (LMS) algorithm [57].[1] The low complexity and the relatively good robustness properties are the major advantages of the LMS algorithm. However, the speed of convergence of the algorithm may be substantially less than desired. There are several possible reasons for a reduced convergence speed. The frequency dependence of the secondary path and the shape of the spectrum of the reference signal may lead to reduced convergence rates. These two influences have been compensated for in the preconditioned filtered-error algorithm [20]. Another algorithm that has similar performance figures, is the hybrid filtered error LMS algorithm [25]. These algorithms have an additional advantage that they are more efficient than the filtered-reference algorithm for the case of multiple reference signals. In the hybrid filtered-error algorithm the error filter has been modified to reduce the influence of the delay in the adaptation loop, which in turn improves the convergence properties. In the preconditioned filtered-error LMS (PLMS) algorithm it was proposed to filter the reference signals [54] with the inverse of the minimum-phase process that colors the reference signals and to use a minimum-phase/all-pass decomposition for the secondary path. A method using a whitening of multiple reference signals and single-channel minimum-phase/all-pass decomposition was presented in Ref. [26]. Another type of factorization based on singular value decomposition in the frequency domain suitable for multichannel systems was presented in Ref. [28]. A disadvantage of the filtered-error algorithm is that the maximum convergence speed is reduced due to the inherent delay in

[1] The work in this chapter was previously published in The Journal of the Acoustical Society of America (See [58])

the adaptation path. The negative influence of this delay can be eliminated by using an inner-outer factorization of the transfer path between the actuators and the error sensors, combined with a delay compensation technique using double control filters [29]. The latter algorithm can be combined with a regularization technique that preserves the factorization properties [29]. Also in the latter algorithm, the so-called Regularized Modified Filtered-Error (RMFe) algorithm, a requirement for good convergence is that the reference signals are white. In the Affine Projection (AP) algorithm and the Fast Affine Projection (FAP) algorithms [31–33], the decolorizing mechanism is included in the adaptation loop. The latter algorithms are able to decolorize auto-regressive processes [39]. In the latter reference, Rupp also described algorithms that are able to decolorize moving average processes. A multi-channel modified filtered reference implementation using AP or FAP was introduced by Bouchard [34]. In the AP and the FAP algorithm it is necessary to calculate the inverse of an estimated auto-correlation matrix. In the algorithm introduced by Bouchard this inversion is performed by using an iterative Gauss-Seidel algorithm. In another publication it was proposed to use the conjugate gradient method to find the inverse [35]. Other methods were based on the assumption that the governing matrix is Toeplitz [36], which holds when the adaptive filter is considerably longer than the affine projection order. In a paper by Heping an overview of methods for finding the inverse matrix is presented [37]. Heping analyzes the complexity for the most widely used methods, and also gives a method for computing the inverse with an LDL^T decomposition. Fast approximate implementations of the FAP-algorithm are given by Douglas [38].

The major contributions of the present paper are as follows. Firstly, results are given of a real-time implementation of the RMFe algorithm of Ref. [29]. Real-time results will be given for feedback and feedforward implementations. The steady-state reductions of the algorithms obtained in real-time are compared with the optimal, causal Wiener filter solutions. Secondly, methods are described to improve the convergence speed of the RMFeLMS algorithm for colored reference signals by an extension with a multichannel FAP algorithm, using the possibility for delay-less adaptation as provided by the RMFe algorithm. The paper is organized as follows. Section 2 will give an introduction to the RMFeLMS algorithm. Section 3 gives a description of the modifications that are needed to extend the RMFeLMS algorithm with an AP algorithm and its fast derivatives. Real-time experimental results and the optimal Wiener solutions will be presented in Section 4. The results will be presented for a feedback architecture as well as for a feedforward architecture. Section 5 will contain the conclusions.

3.2 The regularized modified filtered-error algorithm

In this section the equations for the regularized modified filtered-error algorithm will be derived. The starting point is the filtered-error algorithm (see Fig. 3.1).

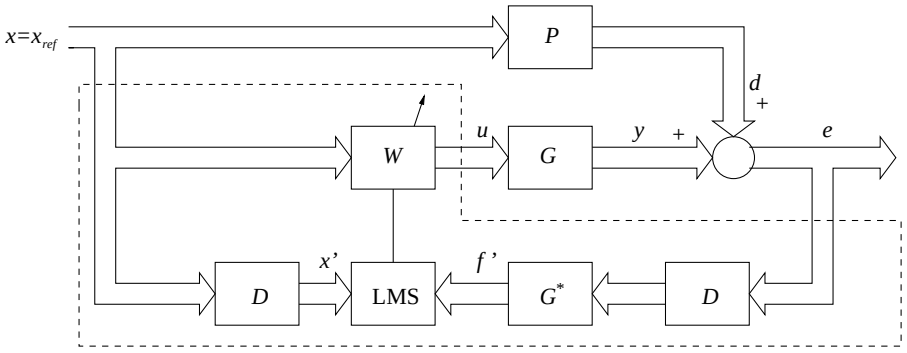


Figure 3.1: *Filtered-error adaptive control scheme.*

The controller can be found within the dashed line.

The sample moment is denoted by n and the unit-delay forward-shift operator as q. The signal vectors are defined as $x(n) = [x_1(n) \dots x_K(n)]^T$, $d(n) = [d_1(n) \dots d_L(n)]^T$, $e(n) = [e_1 \dots e_L(n)]^T$, and $u(n) = [u_1 \dots u_M(n)]^T$, which are the reference signals, the disturbance signals, the error signals, and the actuator driving signals, respectively. The assumption is that noise which is uncorrelated with the reference signal is negligible. The disturbance signals $d(n)$ result from an assumed $L \times K$-dimensional primary path $P(q)$ driven by the reference signals. The goal of the algorithm is to minimize the signals $e(n) = d(n) + y(n)$, where $y(n)$ are the contributions from an $L \times M$-dimensional transfer path $G(q)$, the secondary path, which is driven by actuator driving signals $u(n)$. The actuator driving signals are obtained by feeding the reference signals through an $M \times K$-dimensional control filter $W(q)$. The controller $W(q)$ is assumed to be an $M \times K$ dimensional matrix with finite impulse response (FIR) filters. The i-th filter coefficient of this control filter is denoted as the $M \times K$ matrix W_i, where $i = 0..I-1$, i.e., $W(q) = \sum_{i=0}^{I-1} q^{-i} W_i$. The adaptation in this algorithm is obtained with an LMS rule:

$$W_i(n+1) = W_i(n) - \alpha f'(n) x'^T(n-i), \tag{3.1}$$

where ^{T} denotes matrix transpose and where $x'(n)$ is a delayed version of the reference signal such that

$$x'(n) = D_K(q)x(n), \tag{3.2}$$

in which D_K is a $K \times K$-dimensional delay operator resulting in a delay of $J-1$ samples:

$$D_K(q) = q^{-J+1} I_K, \tag{3.3}$$

and in which $f'(n)$ is a filtered and delayed version of the error signal, such that

$$f'(n) = G^*(q) D_L(q) e(n). \tag{3.4}$$

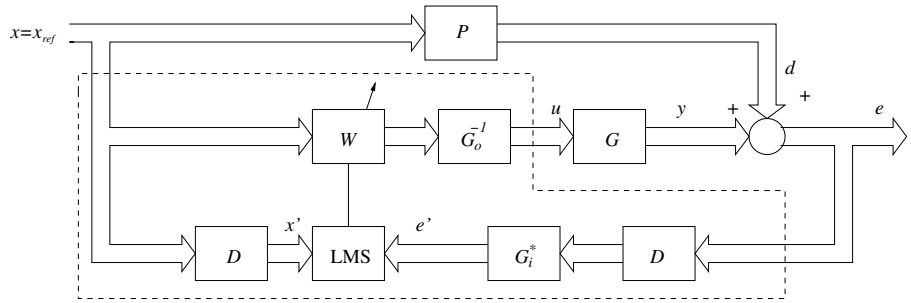


Figure 3.2: *Preconditioned filtered-error adaptive control scheme.*

In Eq. (3.4), the filtering is done with the adjoint $G^*(q)$, which is the transposed time-reverse of the secondary path $G(q)$, i.e., $G^*(q) = G^T(q^{-1})$. The adjoint $G^*(q)$ is anti-causal and has the dimensions $M \times L$. To make the adjoint predominantly causal it needs an additional delay operator D_L. The resulting matrix of transfer functions can be implemented as an $M \times L$ matrix of impulse responses with order $J - 1$, resulting in J coefficients for each impulse response. The adaptation rate [54] is controlled by α.

The frequency dependence of the secondary path $G(q)$ and the interaction of the different channels will reduce the convergence rate. The convergence can be improved by inserting the inverse of the secondary path in between the control filter $W(q)$ and the secondary plant $G(q)$ (Ref.[54]). To ensure stability, only the minimum-phase part of the plant is to be inverted. The secondary path can be written as

$$G(q) = G_i(q)G_o(q), \tag{3.5}$$

where

$$G^*(q)G(q) = G_o^*(q)G_o(q), \tag{3.6}$$

since

$$G_i^*(q)G_i(q) = I_M. \tag{3.7}$$

Assuming $L \geq M$, the transfer function $G_i(q)$ has dimensions $L \times M$ and the transfer function $G_o(q)$ has dimensions $M \times M$. The extraction of the minimum-phase part and the all-pass part is performed with a so-called inner-outer factorization [59, 60]. The control scheme in which the inverse $G_o^{-1}(q)$ is used can be found in Fig. 3.2. The update rule for this approach is

$$W_i(n+1) = W_i(n) - \alpha e'(n){x'}^T(n-i). \tag{3.8}$$

In Fig. 3.2, the filtered error is denoted with $e'(n)$, emphasizing that the spectral characteristics of $e'(n)$ closely resemble that of the real error signal $e(n)$. It

should be noted, however, that the dimensions are different since $e(n)$ is an $L \times 1$-dimensional signal while $e'(n)$ is an $M \times 1$-dimensional signal.

A shortcoming of the algorithm in Fig. 3.2 is that the convergence rate still suffers from the delay in the adaptation path. The path that reduces the convergence in Fig. 3.2 will now be analyzed. From this analysis, another algorithm will be derived that improves the behavior of the convergence. Let us start with the error signal $e'(n)$, which can be written as

$$e'(n) = G_i^*(q)D_L(q)[d(n) + G(q)G_o^{-1}(q)W(q)x(n)]. \tag{3.9}$$

Introducing the $M \times M$ matrix $D_M(q)$ having a delay that is identical to the $L \times L$ matrix $D_L(q)$, and using D_L instead of D_M, Eq. (3.9) can be rearranged to

$$\begin{aligned} e'(n) &= G_i^*(q)D_L(q)d(n) + \\ &\quad D_M(q)G_i^*(q)G(q)G_o^{-1}W(q)x(n). \end{aligned} \tag{3.10}$$

Using Eqs. (3.5) and (3.7), $e'(n)$ can be expressed as

$$e'(n) = d'(n) + y'(n), \tag{3.11}$$

where an auxiliary disturbance signal $d'(n)$is defined as

$$d'(n) = G_i^*(q)D_L(q)d(n), \tag{3.12}$$

and where, according to Eq. (3.7), the delayed preconditioned control output $y'(n)$ is defined by

$$y'(n) = D_M(q)W(q)x(n). \tag{3.13}$$

From this equation it can be seen that the transfer function between the output of $W(q)$ and $y'(n)$ is a simple delay $D_M(q)$. Let us now define an auxiliary output $y''(n) = y'(n)$ by

$$y''(n) = W(q)D_K(q)x(n), \tag{3.14}$$

where $D_K(q)$ is $K \times K$ dimensional matrix having the same delay as $D_M(q)$. In this equation there is no delay anymore between the controller $W(q)$ and $y''(n)$. In order to realize this, the signal $e''(n) = e'(n)$ is introduced by noting that $y'(n) = y''(n)$:

$$e''(n) = d'(n) + y''(n). \tag{3.15}$$

Since $d'(n)$ is not directly available its needs to be reconstructed. This can be done using Eq. (3.11):

$$d'(n) = e'(n) - y'(n), \tag{3.16}$$

where, according to Eq. (3.13), $y'(n)$ can be obtained as the delayed version of the output of $W(q)$. Using $x'(n) = D_K(q)x(n)$, $y''(n)$ can be written as

$$y''(n) = W(q)x'(n). \tag{3.17}$$

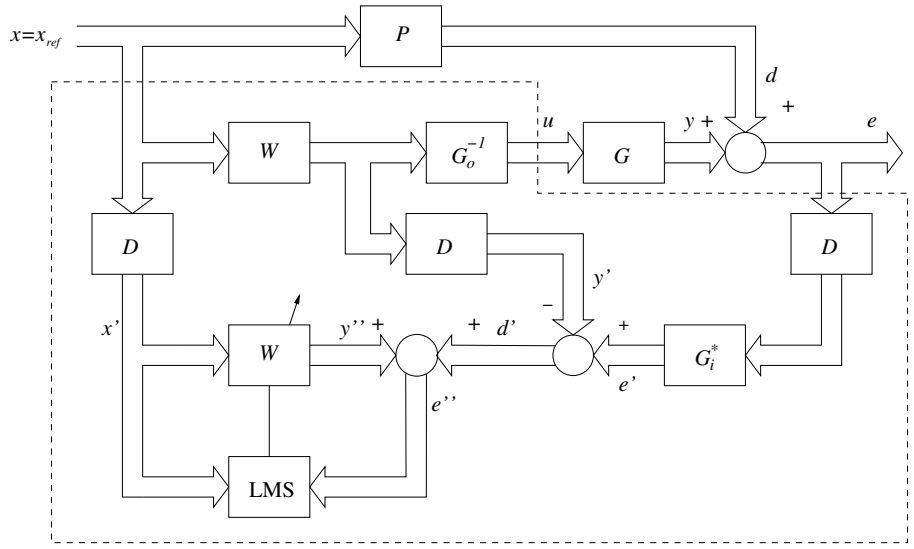


Figure 3.3: *Modified filtered-error adaptive control scheme.*

The signal $x'(n)$ is already available as the input to the LMS block in the control schemes of Figs. 3.1 and 3.2. Equation (3.15) can now be written as

$$e''(n) = d'(n) + W(q)x'(n). \tag{3.18}$$

The term $y''(n)$ can be obtained by adding a second set of control filters W operating on the delayed reference signals $x'(n)$. A block diagram based on Eq. (3.18) can be found in Fig. 3.3. The update rule for Fig. 3.3 is

$$W_i(n+1) = W_i(n) - \alpha e''(n){x'}^T(n-i). \tag{3.19}$$

In the control scheme of Fig. 3.3 (and also Fig. 3.2), the output $u(n)$ can be very large if the plant G contains zeros or near zeros. In this case the inverse of the outer factor $G_o^{-1}(q)$ can have very high gains, which leads to saturation of the control signal $u(n)$. To prevent this some kind of regularization is required [29]. The required regularization must satisfy Eqs. (3.5) to (3.7) [29]. This can be realized by using an augmented plant $\overline{G}(q)$:

$$\overline{G}(q) = \begin{bmatrix} G(q) \\ G_{\mathrm{reg}}(q) \end{bmatrix}, \tag{3.20}$$

in which the regularization is performed by the $L' \times M$-dimensional transfer function $G_{\mathrm{reg}}(q)$. For simple weighting of $u(n)$ the regularization function can be set to

$$G_{\mathrm{reg}}(q) = \sqrt{\beta} I_M. \tag{3.21}$$

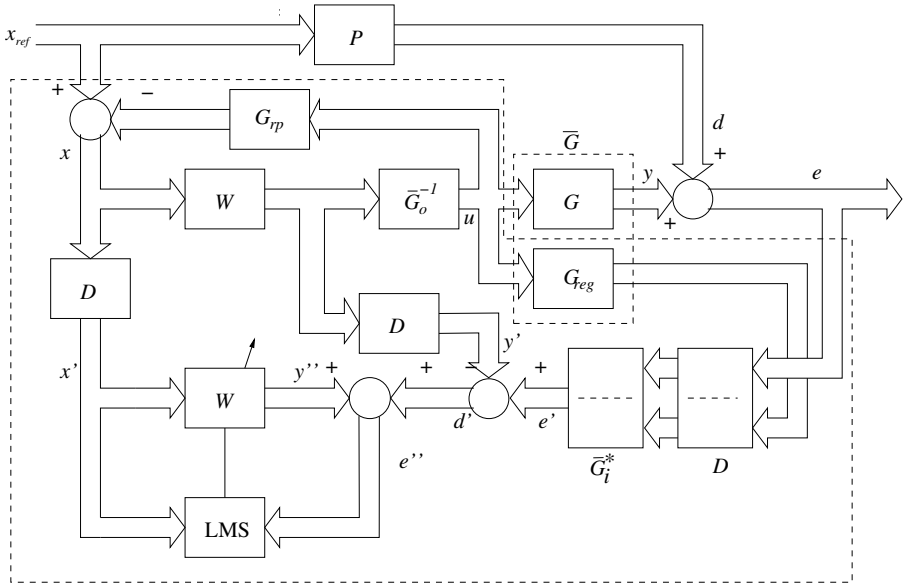


Figure 3.4: *Regularized modified filtered-error adaptive control scheme with IMC.*

The full $(L+L') \times M$ dimensional augmented plant is used in the control scheme, as well as the full $(L+L') \times M$-dimensional inner-factor $\overline{G}_i(q)$ and the $M \times M$-dimensional outer-factor inverse $\overline{G}_o^{-1}$. The modified inner-factor $\overline{G}_i$ is all-pass because $\overline{G}_i \overline{G}_o = \overline{G}$. A control scheme in which this regularization is incorporated can be found in Fig. 3.4. This method has the advantage that it is consistent with the modified filtered-error scheme, which is dependent on the all-pass property $\overline{G}_i^*(q)\overline{G}_i(q) = I_M$. The scheme in Fig. 3.4 is a generalized form meaning that different regularization schemes can be used for $G_{\text{reg}}(q)$.

3.3 Affine projection

The algorithm presented in Fig. 3.4 has good convergence properties when the reference signals are white. Such a situation also arises when the algorithm is used in a feedback architecture [29]. However, the performance degrades if the signals are strongly colored. In the original preconditioned filtered-error algorithm [54] the reference signals are pre-filtered with the transfer function that colors the noise [54], assuming a-priori knowledge of the process that colorizes the noise. The affine projection algorithm is capable of estimating the autoregressive part of the process that colors the noise [39] while adapting the controller parameters. In this section the Affine projection algorithm (AP) and the fast affine projection algorithm (FAP) will be derived, starting from the optimal Wiener filter. The

FAP algorithm will be derived especially for the regularized modified filtered-error scheme. The derivation starts by rewriting Eq. (3.18) as:

$$e''(n) = d'(n) + \mathbf{W}^T \underline{x}'(n), \tag{3.22}$$

in which the delay line $\underline{x}'$ is defined as

$$\underline{x}'(n) = \left[\begin{array}{cccc} x'^T(n) & \dots & x'^T(n-I+1) \end{array} \right]^T, \tag{3.23}$$

and the filter coefficient matrix $\mathbf{W}$ is defined as

$$\mathbf{W} = \left[\begin{array}{ccccc} \underline{\mathbf{w}}_1 & \dots & \underline{\mathbf{w}}_m & \dots & \underline{\mathbf{w}}_M \end{array} \right], \tag{3.24}$$

with

$$\underline{\mathbf{w}}_m = \left[\begin{array}{c} \mathbf{w}_{1m} \\ \vdots \\ \mathbf{w}_{im} \\ \vdots \\ \mathbf{w}_{Im} \end{array} \right] \qquad \mathbf{w}_{im} = \left[\begin{array}{c} w_{1im} \\ \vdots \\ w_{kim} \\ \vdots \\ w_{Kim} \end{array} \right]. \tag{3.25}$$

The MIMO-FAP algorithm will be derived starting at the MIMO-Wiener filter solution. Using Eq. (3.22) the cost criterium becomes

$$\begin{aligned} J &= \text{tr.}\{\text{E}[e''(n)e''^T(n)]\} = \\ &\quad \text{tr.E}\{(d'(n) + \mathbf{W}^T\underline{x}'(n))(d'(n) + \mathbf{W}^T\underline{x}'(n))^T\}. \end{aligned} \tag{3.26}$$

where tr. denotes the trace operator and E denotes the expected value. Equation (3.26) can be written as:

$$\begin{aligned} J &= \text{tr.}\{\mathbf{R}_{d'd'}\} + \text{tr.}\{\mathbf{R}_{x'd'}^T\mathbf{W}\} + \text{tr.}\{\mathbf{W}^T\mathbf{R}_{x'd'}\} \\ &\quad + \text{tr.}\{\mathbf{W}^T\mathbf{R}_{x'x'}\mathbf{W}\}, \end{aligned} \tag{3.27}$$

with

$$\begin{aligned} \mathbf{R}_{d'd'} &= \text{E}\{d'(n)d'^T(n)\}, \\ \mathbf{R}_{x'd'} &= \text{E}\{\underline{x}'(n)d'^T(n)\}, \\ \mathbf{R}_{x'x'} &= \text{E}\{\underline{x}'(n)\underline{x}'^T(n)\}. \end{aligned} \tag{3.28}$$

The gradient of J with respect to the controller coefficients is:

$$\frac{\partial J}{\partial \mathbf{W}} = 2(\mathbf{R}_{x'd'} + \mathbf{R}_{x'x'}\mathbf{W}). \tag{3.29}$$

The minimum for J is obtained by setting $\frac{\partial J}{\partial \mathbf{W}}$ to zero, resulting in the optimum MIMO-Wiener filter $\mathbf{W}_{\text{opt}}$

$$\mathbf{W} \quad = \quad \mathbf{W}_{\text{opt}} = -\mathbf{R}_{x'x'}^{-1}\mathbf{R}_{x'd'}. \tag{3.30}$$

The derivation of the AP algorithm will start from a Newton iteration [55], which is written as

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu\hat{\mathbf{R}}_{x'x'}^{-1}\left[\hat{\mathbf{R}}_{x'd'} + \hat{\mathbf{R}}_{x'x'}\mathbf{W}(n)\right], \tag{3.31}$$

where $\hat{\mathbf{R}}_{x'x'}$ and $\hat{\mathbf{R}}_{x'd'}$ are estimates of $\mathbf{R}_{x'x'}$ and $\mathbf{R}_{x'd'}$, respectively. Let K_A be the affine projection order, then the approximations for the correlation matrices are written as:

$$\hat{\mathbf{R}}_{x'x'} = \frac{1}{K_A}\sum_{i=0}^{K_A-1}\underline{x}'(n-i)\underline{x}'^T(n-i) = \frac{1}{K_A}\mathbf{X}\mathbf{X}^T, \tag{3.32}$$

and

$$\hat{\mathbf{R}}_{x'd'} = \frac{1}{K_A}\sum_{i=0}^{K_A-1}\underline{x}'(n-i)d'^T(n-i) = \frac{1}{K_A}\mathbf{X}\mathbf{D}, \tag{3.33}$$

in which $\mathbf{X}(n)$ is an $IK \times K_A$-dimensional matrix that contains the last K_A vectors of $\underline{x}'(n)$, i.e.,

$$\mathbf{X}(n) = \left[\begin{array}{ccc} \underline{x}'(n) & \ldots & \underline{x}'(n-K_A+1) \end{array}\right]. \tag{3.34}$$

and in which $\mathbf{D}(n)$ is the $K_A \times M$-dimensional matrix containing the last K_A vectors of $d'(n)$, i.e.,

$$\mathbf{D}(n) = \left[\begin{array}{c} d'^T(n) \\ \vdots \\ d'^T(n-K_A+1) \end{array}\right]. \tag{3.35}$$

For notational convenience the time index of $\mathbf{D}(n)$ and $\mathbf{X}(n)$ will be dropped. Substitution of Eqs. (3.32) and (3.33) into Eq. (3.31) results in:

$$\begin{aligned} \mathbf{W}(n+1) &= \mathbf{W}(n) - \mu[\mathbf{X}\mathbf{X}^T]^{-1}[\mathbf{X}\mathbf{X}^T\mathbf{W}(n) + \mathbf{X}\mathbf{D}] \\ &= \mathbf{W}(n) - \mu[\mathbf{X}\mathbf{X}^T]^{-1}\mathbf{X}[\mathbf{X}^T\mathbf{W}(n) + \mathbf{D}]. \end{aligned} \tag{3.36}$$

In order to prevent problems with the inversion of $\mathbf{X}\mathbf{X}^T$, a small regularization term δ will be added to the diagonal elements. Then, the overall update rule becomes

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu[\mathbf{X}\mathbf{X}^T + \delta\mathbf{I}]^{-1}\mathbf{X}[\mathbf{X}^T\mathbf{W}(n) + \mathbf{D}], \tag{3.37}$$

which, using the matrix inversion lemma

$$(\mathbf{A}+\mathbf{B}\mathbf{C}\mathbf{D})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B}]^{-1}\mathbf{D}\mathbf{A}^{-1}, \tag{3.38}$$

can be written as

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu\mathbf{X}[\delta\mathbf{I} + \mathbf{X}^T\mathbf{X}]^{-1}[\mathbf{X}^T\mathbf{W}(n) + \mathbf{D}]. \tag{3.39}$$

Due to the latter step the matrix inversion can be computed more efficiently. Equation (3.39) defines the Affine Projection (AP) algorithm.

The computational complexity of AP can be further reduced by the FAP algorithm [32, 33]. Starting from the basic AP equation it will be shown how to expand the FAP algorithm to work in combination with the scheme presented in Fig. 3.4. Equation (3.39) will be written as

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu\mathbf{X}(n)\boldsymbol{\xi}(n), \tag{3.40}$$

in which

$$\boldsymbol{\xi}(n) = [\delta\mathbf{I} + \mathbf{X}^T(n)\mathbf{X}(n)]^{-1}\mathbf{E}(n). \tag{3.41}$$

and in which a $K_A \times M$-dimensional matrix $\mathbf{E}(n)$ is defined that contains the last K_A error vectors:

$$\mathbf{E}(n) = \begin{bmatrix} e''^T(n) \\ \vdots \\ e''^T(n-K_A+1) \end{bmatrix}, \tag{3.42}$$

such that

$$\mathbf{E}(n) = \mathbf{D}(n) + \mathbf{X}^T(n)\mathbf{W}(n). \tag{3.43}$$

In the original FAP algorithm the three equations Eq. (3.43), (3.41), and (3.40) will be reduced in complexity. In the original FAP algorithm [33] it has been proposed to use a fast transversal filter to estimate the inverse autocorrelation matrix. This leads to issues with stability so it has been proposed [35] to calculate the inverse directly. In this paper only Eqs. (3.43) and (3.40) will be simplified and the matrix inversion in Eq. (3.41) will be approximated using a steepest-descent iteration. The error signal Eq. (3.43) will be calculated in such a way that the complexity will fall from $IK \times K_A$ to $IK \times 1$. The complexity of the update rule Eq. (3.40) will be reduced from $IK \times K_A \times L$ to $IK \times L$. It can be shown [33] that the error signal matrix can be written in the following form

$$\mathbf{E}(n) = \begin{bmatrix} e''^T(n) \\ (1-\mu)\overline{\mathbf{E}}(n-1) \end{bmatrix}, \tag{3.44}$$

in which $\overline{\mathbf{E}}(n)$ are the $K_A - 1$ top rows of $\mathbf{E}(n-1)$.
FAP uses the assumption that the update rule can be rewritten in a form that uses alternative update coefficients. In this paper these alternative update coefficients will be derived. The reason for this is that the algorithm presented in Fig. 3.4 uses a copy of the filter coefficients in the upper branch. This makes it necessary to also find a new expression for these filter coefficients. The alternative filter coefficients

will be written as $\hat{\mathbf{W}}$.
The alternative filter coefficients are defined as

$$\begin{aligned}\hat{\mathbf{W}}(n+1) &= \hat{\mathbf{W}}(n) - \mu \underline{x}'(n-(K_A-1)) \times \\ &\quad \underline{E}_{K_A-1}(n),\end{aligned} \tag{3.45}$$

with $\underline{E}_{K_A-1}(n)$ defined as the bottom row of $\underline{\mathbf{E}}$, which in turn is defined as

$$\underline{\mathbf{E}}(n) = \boldsymbol{\xi}(n) + \begin{bmatrix} 0_M^T \\ \overline{\underline{\mathbf{E}}}(n-1) \end{bmatrix}. \tag{3.46}$$

This result in the following update rule for $W(n)$

$$\mathbf{W}(n+1) = \hat{\mathbf{W}}(n+1) - \mu \overline{\mathbf{X}}(n) \overline{\underline{\mathbf{E}}}(n), \tag{3.47}$$

in which $\overline{\mathbf{X}}(n)$ is defined as the $K_A - 1$ left most columns of $\mathbf{X}(n)$. The derivation of Eqs. (3.46) and (3.47) can be found in Appendix A.

Let us study the relation between $\mathbf{E}(n)$ and $\mathbf{E}(n-1)$. First an expression for the first row of Eq. (3.44) will be derived. Since $\mathbf{W}(n)$ is not directly available, we use Eq. (3.47) and substitute it in the transpose of Eq. (3.22):

$$\begin{aligned}e''^T(n) &= d'^T(n) + \underline{x}'^T(n) \times \\ &\quad [\hat{\mathbf{W}}(n) - \mu \overline{\mathbf{X}}(n-1) \overline{\underline{\mathbf{E}}}(n-1)].\end{aligned} \tag{3.48}$$

This can be simplified into:

$$\begin{aligned}e''^T(n) &= d'^T(n) + \underline{x}'^T(n) \hat{\mathbf{W}}(n) - \\ &\quad \mu \underline{x}'^T(n) \overline{\mathbf{X}}(n-1) \overline{\underline{\mathbf{E}}}(n-1) \\ &= d'^T(n) + \underline{x}'^T(n) \hat{\mathbf{W}}(n) - \\ &\quad \mu r_{aa}^T(n) \overline{\underline{\mathbf{E}}}(n-1),\end{aligned} \tag{3.49}$$

with

$$\begin{aligned}r_{aa}(n) &= r_{aa}(n-1) + \underline{\mathbf{x}}'^T(n) x'(n) \\ &\quad - \underline{\mathbf{x}}'^T(n-I+1) x'(n-I+1),\end{aligned} \tag{3.50}$$

and

$$\underline{\mathbf{x}}'(n) = \begin{bmatrix} x'(n-1) & \dots & x'(n-K_A+1) \end{bmatrix}. \tag{3.51}$$

This results in the update equation for the error signal matrix:

$$\mathbf{E}(n) = \begin{bmatrix} d'^T(n) + \underline{x}'^T \hat{\mathbf{W}}(n) - r_{aa}^T \overline{\underline{\mathbf{E}}}(n-1) \\ (1-\mu) \overline{\underline{\mathbf{E}}}(n-1) \end{bmatrix}, \tag{3.52}$$

which has dimensions $K_A \times M$.

The output $y_w(n)$ of the upper filter $\mathbf{W}$ in Fig. 3.4 equals

$$y_w(n) = \mathbf{W}^T(n)\underline{x}(n), \tag{3.53}$$

with the delay line $\underline{x}(n)$ defined as

$$\underline{x}(n) = \left[\begin{array}{cccc} x^T(n) & \ldots & x^T(n-I+1) \end{array} \right]^T. \tag{3.54}$$

However, in the FAP routine $\mathbf{W}$ is not directly available. So Eq. (3.47) is substituted into Eq. (3.53) resulting in

$$\begin{aligned} y_w(n) &= [\hat{\mathbf{W}}(n) - \mu\overline{\mathbf{X}}(n-1)\overline{\underline{\mathbf{E}}}(n-1)]^T\underline{x}(n) \\ &= \hat{\mathbf{W}}^T(n)\underline{x}(n) - \mu\overline{\underline{\mathbf{E}}}^T(n-1)\overline{\mathbf{X}}^T(n-1)\underline{x}(n) \\ &= \hat{\mathbf{W}}^T(n)\underline{x}(n) - \mu\overline{\underline{\mathbf{E}}}^T(n-1)r_{ax}(n), \end{aligned} \tag{3.55}$$

with

$$\begin{aligned} r_{ax}(n) &= r_{ax}(n-1) + \underline{\mathbf{x}}'^T(n)x(n) \\ &\quad -\underline{\mathbf{x}}'^T(n-I+1)x(n-I+1). \end{aligned} \tag{3.56}$$

The inverse in Eq. (3.41), in the following denoted as $\mathbf{X}_{pi}$, was computed by taking the inverse of the autocorrelation matrix $\mathbf{X}_p$, where the latter is updated by the following recursive scheme:

$$\mathbf{X}_p(n) = \left[\begin{array}{cc} r_a & r_{aa}^T \\ r_{aa} & \overline{\mathbf{X}}_p(n-1) \end{array} \right], \tag{3.57}$$

in which

$$\begin{aligned} r_{aa}(n) &= r_{aa}(n-1) + \underline{\mathbf{x}}'^T(n)x'(n) \\ &\quad -\underline{\mathbf{x}}'^T(n-I+1)x'(n-I+1), \end{aligned} \tag{3.58}$$

$$\begin{aligned} r_a(n) &= r_a(n-1) + x'^T(n)x'(n) \\ &\quad -x'^T(n-I+1)x'(n-I+1), \end{aligned} \tag{3.59}$$

and in which $\overline{\mathbf{X}}_p(n-1)$ consists of the $K_A - 1 \times K_A - 1$-dimensional upper-left matrix of $\mathbf{X}_p(n)$.
The actual inverse is computed with a steepest-descent iteration [61], as follows. A square matrix $\mathbf{R}$ of size $K_A \times K_A$ is defined in which each column is a row vector r_i of length K_A, in which $i = 1..K_A$ indexes the column of $\mathbf{R}$:

$$\mathbf{R} = \left[\begin{array}{ccc} r_1 & \ldots & r_{K_A} \end{array} \right]. \tag{3.60}$$

Each column of the inverse $\mathbf{X}_{pi}$ will be denoted by a separate vector $x_{pi,i}$ with $i = 1..K_A$ the column index:

$$\mathbf{X}_{pi} = \left[\begin{array}{ccc} x_{pi,1} & \ldots & x_{pi,K_A} \end{array} \right]. \tag{3.61}$$

The steepest descent iteration is performed with the following algorithm:

$\mathbf{R} = \mathbf{I}_{K_A} - \mathbf{X}_p\mathbf{X}_{pi}$;
for $k = 1$ until K_A
$\alpha_k = \frac{r_k^T r_k}{r_k^T \mathbf{X}_p r_k}$;
$x_{pi,k} = x_{pi,k} + \alpha_k r_k$;
end for

The FAP-algorithm can be broken down into four steps. In the first step the output of the upper branch filter is calculated, using Eq. (3.55). In the second part, the auto-correlation matrix is efficiently calculated and updated, using Eq. (3.57). The third step will calculate the inverse of the auto-correlation matrix using a steepest descent iteration. And in the fourth and final step the error, the de-correlated error and the update rule are calculated by computation of Eqs. (3.52), (3.46), (3.41), and (3.45). One of the interesting features of the combination of RMFe with FAP is that the disturbance signals $d'(n)$ which are required for FAP are readily available from RMFe. The FAP-algorithm starts with the initialization of the different variables:

$$\begin{array}{lll} \mathbf{X}_p = \delta\mathbf{I}_{K_A} & \mathbf{X}_{pi} = \frac{1}{\delta}\mathbf{I}_{K_A} & r_a = \delta \\ r_{aa} = \underline{0}_{K_A-1} & r_{ax} = \underline{0}_{K_A-1}. & \end{array}$$

The inverse $\mathbf{X}_{pi}$ at a particular sample n will be initialized with the inverse as computed at a previous sample $n-1$.

The complexity of the different algorithms will be analyzed. First assume that the algorithms will be broken down in some basic operation and secondly assume that multiplication and additions are counted as one operation. The complexity of the inverse inner-factor is $O(2(N+M)^2)$. A reduction of complexity to $O(6NM + 2(N+M)M)$ is possible when an output normal form is used. The complexity of the transposed-conjugate outer factor is $O(2JLM)$ for PLMS and $O(4JLM)$ for RMFeLMS. The control filter W has a complexity of $O(2ILM)$. And the LMS update rule needs $O(3ILM)$ computations. The AP/FAP update rule can be broken down into three operations, the first one being the error vector with a complexity of $O(MK_A + 2KMK_AI)$ for AP and O(M+2KMI) for FAP. In the second step the inverse matrix and the de-correlated error vector are calculated with a complexity of $O(4K_A^3 + 2K_A^2 + 2K^2M)$. In the third step the update rule will be computated with a complexity of $O(2KMI + KMIK_A + MK^2)$ for AP and $O(3KMI + M)$ for FAP. The RMFeLMS and PLMS algorithm both have a higher complexity than FeLMS. The complexity of RMFeLMS is almost similar to that of PLMS where the extra complexity is due to the copy of W and the regularization part. The regularization ensures a practically feasible and stable algorithm. The performance and convergence properties of the Hybrid filtered error algorithm [25] are very close to those of the PLMS algorithm. However, the convergence properties and the mean square error of RMFeLMS are even better than that of PLMS [29]. It also must be noted that FeLMS and its derivatives

are a different kind of algorithms with different design trade offs. For instance, FeLMS usually needs longer filters than RMFeLMS.

3.4 Results

A configuration with 5 piezoelectric sensors and 5 piezoelectric actuators on a panel was used for testing of the regularized modified filtered-error algorithm. The control architecture was feedback, using the internal model control principle. On the top and the bottom of the panel, each 9 piezoelectric patches were attached. The piezoelectric patches in the center of the plate and the corners of the plate were used. The panel was mounted on a box made of perspex, as shown in Fig. 3.5. In the bottom of this box a loudspeaker was mounted that was used to generate the primary source signal. The honeycomb panel as shown in Fig. 3.5 was built up from epoxy faces of 0.4 mm and an epoxy honeycomb core of 5mm thickness. The boundaries of the plate were simply supported. The noise had a Gaussian amplitude distribution with a bandwidth of 1 kHz.

The sandwich panels as used for the experiments were designed in such a way that noise reductions could be obtained in the frequency range of interest using a relatively simple control objective based on a direct minimization of the signals from the piezoelectric sensors. Some of the considerations leading to the final design will be mentioned. Firstly, the actuator configuration was selected such that the structural modes radiating to the farfield [16] could be controlled. This led to a five-actuator configuration similar to that of Sors and Elliott [10]. Secondly, the sensors were placed collocated with respect to the actuators in order to minimize control spillover, taking into account possible in-plane coupling effects. In-plane coupling [18] between piezoelectric patch actuators and piezoelectric patch sensors, which can have negative consequences for the control of the acoustically relevant out-of-plane vibrations, was reduced by the honeycomb core of the sandwich structure. Finite Element simulations of the different configurations and a subsequent analysis based on pole-zero distances [18] showed that insertion of such a core between the piezoelectric patch actuator and the piezoelectric patch sensor improved the damping performance of a feedback control system, approximating the performance of a combination of a piezoelectric patch actuator and an accelerometer. Different types of sensors were also evaluated in real-time control experiments. The use of accelerometers led to noise reductions that were approximately 1 dB higher than obtained with the piezoelectric patch sensors. However, the use of piezoelectric patches was found to lead to more robust performance, probably due to the spatial averaging by these sensors [11] andor the reduced high-frequency content of the sensor signals. In this paper only the results for the piezoelectric patch sensors will be shown.

The controller was implemented on an embedded PC running an RT-Linux operating system. The controller was designed in Simulink/Matlab using custom blocks that were written in the C programming language. The code for the simu-

lation in Simulink was identical to the code to generate the real-time code, using the Realtime Workshop. Analog I/O for this system was implemented on a dedicated module providing 16 analog inputs and 16 analog outputs. This ADDA module samples at a higher sample rate than the control algorithm. This high sample rate, which was set to 100 kHz in the experiments, was reduced by means of a decimator on the input and increased by means of an interpolator at the output, using filtering in discrete-time. Only a simple filter in the analog domain is needed, reducing the need for complex analog electronic filtering. This setup makes it is possible to use different anti-aliasing and reconstruction filters with different cut-off frequencies. To reduce the latency for the feedback controller only the interpolation filter was enabled. The sample rate of the controller was set to 2 kHz. In the first test the convergence rate was obtained by measuring the error signal and storing it in a buffer on the platform. This buffer was then read by a host PC in order to perform the necessary post processing.

The plant was estimated using a sub-space identification technique [21]. These state-space models were converted into the output-normal form [62]. The advantage of the output-normal form is that it can be more efficiently calculated than the underlying state-space system. The state-space model used for the secondary path was of the 50^{th} order. The estimated secondary plant model had a variance accounted for (VAF) of 99.93%. The regularization parameter β was set in such a way that regularization was at a level of 30 dB below the largest frequency-domain peak of G. The length of the time reversed transpose inner-factor was set to J=80. The length of the LMS filters was set to I=40. The following parameters were set for the LMS algorithm: the step size was set to $\alpha = \frac{1}{20}$ and the leakage was set to $\gamma = 1 \times 10^{-5}$. All parameter settings were first tested by means of simulations.

The convergence curves for the five input channels can be found in Fig. 3.6. The convergence curve was obtained by averaging 32 realizations of the squared error signals. For the measurements the controller was disabled for 5 seconds and then switched on during 55 seconds. The data was normalized to the average disturbance level of input channel 3, thereby defining the 0 dB level in Fig. 3.6. The control system reached 10 dB reduction within 5 seconds, i.e. 10000 samples. The overall reduction for all 5 channels after 55 seconds was 12.4 dB for the LMS algorithm and 12.6 dB for a 4-th order FAP algorithm. The overall reduction for the 5 input channels was obtained from the ratio of the total power for the disturbance signals divided by the total power for the error signals. Increasing the FAP order to higher orders did not improve the performance anymore. It can be seen that there is almost no difference between RMFeLMS and RMFeFAP for this feedback control architecture. This can be expected because a feedback controller tries to make the reference signal white, in which case there is no advantage of FAP over LMS. The spectrum for steady-state operation can be found in Fig. 3.7. The overall steady-state reduction over all 5 input channels was 14.1 dB, which is somewhat higher than the 12.6 dB reduction after 55 seconds. The data used to obtain the steady-state reduction was also used to calculate the optimal Wiener filter solution, which is constrained to be causal. The Wiener filter was computed with the methods

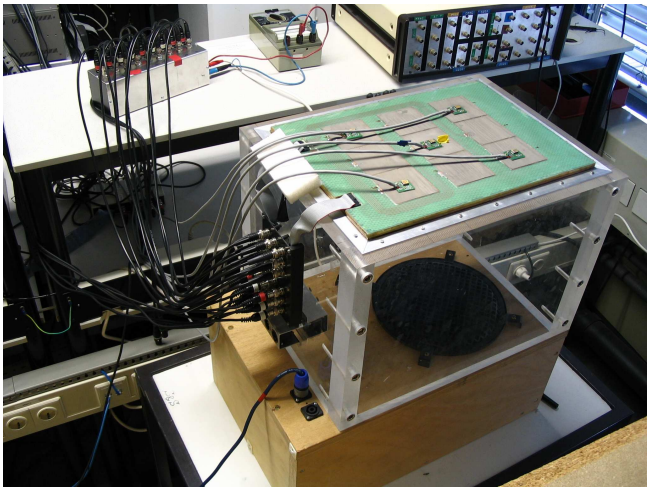


Figure 3.5: *Enclosure with noise source and panel with piezoelectric actuators; the dimensions are given in Fig. 3.8.*

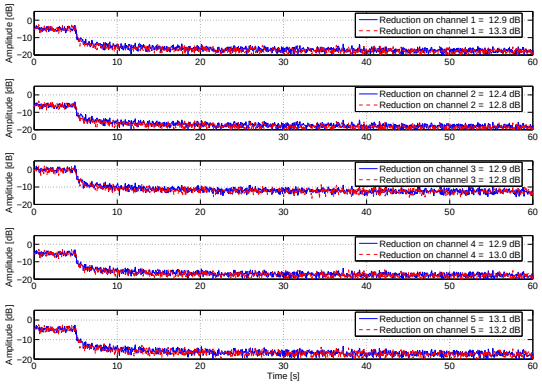


Figure 3.6: *Convergence curve for a real-time 5-input 5-output feedback control system; the reduction after 55 seconds for 5 channels is 12.4 dB (RMFeLMS, solid line) and 12.6 dB (RMFeFAP, dashed line).*

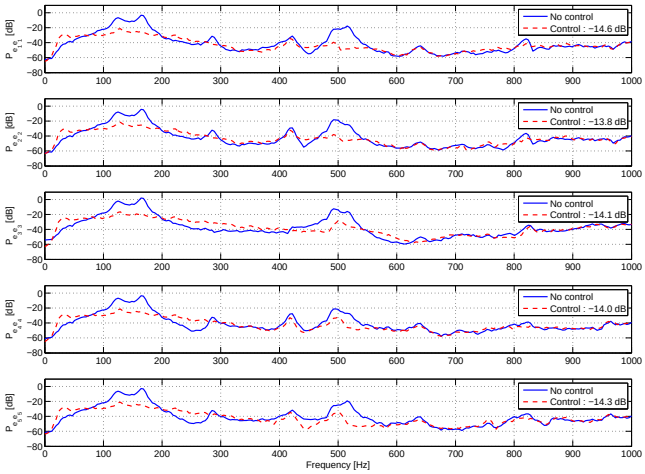


Figure 3.7: *Spectrum of the error signals for a converged real-time feedback controller using the RMFeFAP algorithm; the overall reduction is 14.1 dB at* $\alpha = \frac{1}{40}$.

described in Ref. [63]. The particular computation method for the Wiener filter was based on the use of a single controller W set to reduce the filtered error signals $e'(n) = \overline{G}_i^*(q)D(q)[e(n); G_{\text{reg}}(q)u(n)]$. The inputs for the Wiener filter computation were the reference signal $x(n)$, the secondary path $\overline{G}_i^*(q)D(q)\overline{G}(q)\overline{G}_o^{-1}(q)$, and the disturbance signal $\overline{G}_i^*(q)D(q)[d(n); G_{\text{reg}}(q)u(n)]|_{u(n)=0}$, in which the controller is switched off. The model of the secondary path to derive the Wiener filter is the transfer path from the output of the upper block W in Fig. 3.4 to the signal $e'(n)$ without the part consisting of the single delay block. This delay block was used for improved convergence of the modified adaptive algorithm but should be omitted in the secondary path for computing the optimum Wiener filter, since the optimum Wiener filter as computed here is based on the input signal $x(n)$ instead of $x'(n)$. For feedback control the reference signal was equal to the disturbance signal; in that case only the disturbance signal $d(n)$ was measured. For feedforward control the reference signal $x(n)$ was also measured. The resulting Wiener filter W was then used to compute the error signals $e(n)$. Subsequently, the optimal reduction using the Wiener filter was obtained by comparing $d(n)$ with $e(n)$. In this way it was possible to study the influence of different approximations and parameter settings in the RMFe-algorithm on the steady-state performance. The maximum theoretical overall reduction for the 5 error sensors according to the Wiener filter was 15.5 dB. The real measured overall reduction on the test setup was 14.1 dB. The difference between the estimated results and the real results can be explained by errors in the IMC-model, DC-offset, non-correlated noise sources and non-linearity in the measurement setup, of which the error in the IMC-model was the most important. Finally the standard filtered-error algorithm was tested

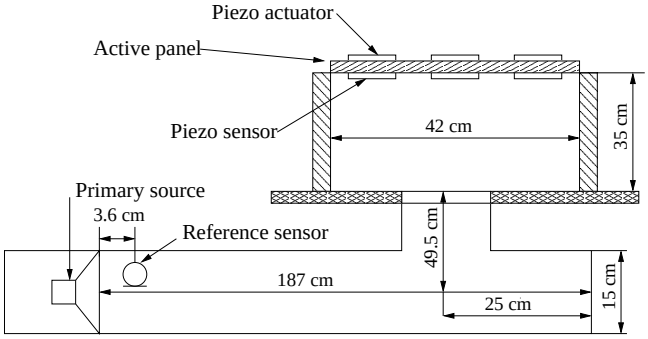


Figure 3.8: *Schematic overview of the setup used for testing of the feedforward RMFe algorithms consisting of a circular pipe connected to the rectangular perspex box of Fig. 3.5; the depth of the perspex box is 30 cm.*

on the feedback test setup, in order to compare it to the RMFe algorithm. Due to the inherent delay in the filtered-error algorithm and the absence of preconditioning the step size had to be set to a relatively small value. The controller already became unstable with a μ of $\frac{1}{1000}$. For the experiments a value of 5×10^{-4} was used, resulting in approximately 4 dB reduction after 55 seconds and more than 1 hour to reach the steady state reduction.

In order to test the FAP algorithm a modification of the test setup was devised. The modification consisted of a circular pipe connected to the bottom of the perspex box. A schematic overview of the setup can be found in Figure 3.8. With this setup the reference signal will be colored due to resonances in the pipe. The real-time results of the RMFeLMS and RMFeFAP implementations can be found in Figure 3.9.

For the FAP algorithm, the following settings were used: $\mu = \frac{1}{40}$, $\delta = \frac{1}{4}$ and $K_A = 28$. The LMS parameters were $\alpha = \frac{1}{10}$, $\gamma = 10^{-5}$. In both algorithms the same regularization parameter β as for the feedback control experiments was used. The length of the time reversed transpose inner-factor was J=80 and the length of the adaptive filter was I=350 taps. Both the LMS and the FAP algorithm used an IMC algorithm to compensate for the feedback from the actuators to the reference microphone. The IMC model was estimated together with the model from the secondary path using the same stimuli. It was found that IMC was necessary for proper operation, also for this feedforward configuration. The estimation accuracy (VAF) was 99.86% for the secondary path model G and 99.49% for the IMC-model G_{rp}. The order of both models was 50. Both models were transformed into the output-normal form. Complexity of the FAP-related computation became dominant for K_A orders of 32 and higher.

From the results in Fig. 3.10 it can be seen that FAP has better convergence properties than the LMS algorithm. After 55 seconds, the LMS algorithm reaches a reduction of 11.5 dB, whereas the FAP algorithm reaches a reduction of 13.5 dB.

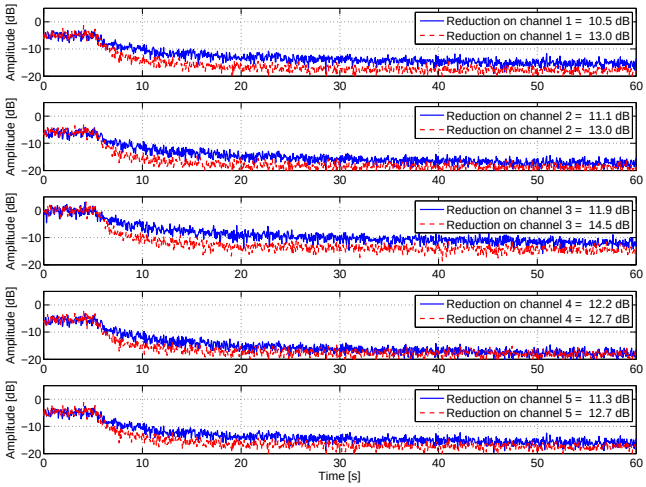


Figure 3.9: *Convergence curve for real-time feedforward control using RMFeFAP (dashed line) and RMFeLMS (solid line); after 55 seconds, RMFeFAP reached 13.5 dB overall reduction and RMFeLMS reached 11.5 dB overall reduction.*

Both algorithms are still not at their maximum reduction after 55 seconds. The optimal Wiener filter predicted a reduction of 14.0 dB. The real-time implementations of both the LMS algorithm and the FAP algorithm were able to realize this reduction. The spectra for the feedforward controller in steady-state conditions can be found in Fig. 3.10. Only the result for the RMFeFAP implementation is shown since the result for RMFeLMS is nearly identical. The maximum reduction reached on all 5 channels is 14.3 dB, which is 0.3 dB higher than the optimal Wiener filter solution. Due to the fact that only one observation is used there was some variance in a single observation. Furthermore it needs to be noted that especially the FAP algorithm can dynamically track small changes in the statistics of the source. The Wiener filter is only optimal for one observation and can not track changes in the source statistics.

In addition to the steepest-descent method for matrix inversion, also a conjugate gradient method was implemented. Use of the conjugate gradient method did not lead to significant performance advantages. A simplification of the FAP algorithm as described in the present paper is possible by assuming $\mu = 1$. Then also the full matrix inversion can be reduced to solving a single system of equations [35]. However, it was found from experiments that the performance deteriorated because of this simplification; the resulting error signals were approximately 1 to 2 dB higher. Finally, tests were performed in order to verify that the feedforward configuration indeed led to performance advantages as compared to the feedback controller. On the setup of Fig. 3.8, a feedback controller resulted in a reduction of 7.9 dB, i.e., substantially less than the 14.3 dB reduction using the feedforward

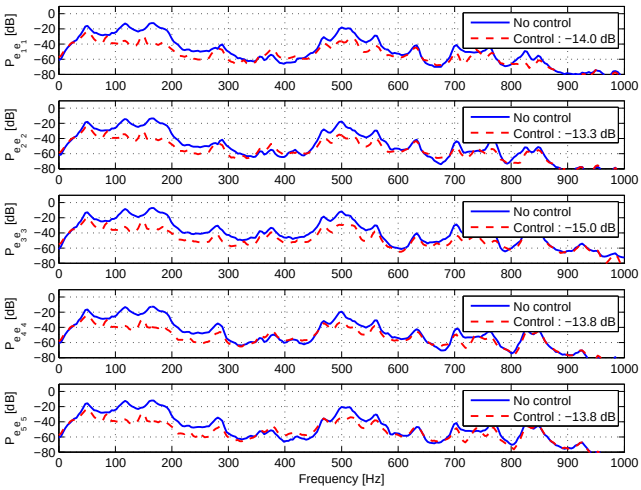


Figure 3.10: *Spectrum of the error signals for a converged real-time feedforward controller using the RMFeFAP algorithm; the overall reduction is 14.3 dB.*

controller.

In order to evaluate the noise reductions, additional tests were carried out on a similar panel with integrated lightweight electronics. Apart from the integrated electronics, the resonance frequencies and other properties of this panel were very similar to the other panel. The noise reduction of the panel was measured by positioning microphones at distances of 50 cm from the panel. The average noise reduction for a feedback control system was 8 dB; the reduction on the error sensors was 10.2 dB. For a feedforward control system the average noise reduction was 7 dB; the reduction on the error sensors was 15.5 dB. If higher noise reductions are to be obtained, particularly with feedforward control, then probably the cost function of the control algorithm should be modified in order to include a relationship between the sensor signals and the acoustic radiation from the panel , such as with a radiation mode based technique [12].

3.5 Conclusions

In this paper results were given of a real-time implementation of the regularized modified filtered-error algorithm. The algorithm was extended with the fast affine projection method in order to improve the speed of convergence for colored reference signals. It was shown how to combine the extension with the RMFe algorithm. In both feedback and feedforward control scenarios, the controller was used to reduce the vibrations of a panel by driving piezoelectric patches. The maximum difference of the predicted reduction and the reduction obtained in real-time ex-

periments was 1.4 dB for feedback control and 0.3 dB for feedforward control. It was shown that the RMFeFAP algorithm has better convergence properties than the RMFeLMS algorithm for the case of colored reference signals, which was verified in a feedforward control configuration. For feedback control no significant differences between the algorithms were found, as expected.

Chapter 4

The RMFeLMS algorithm and HAC/LAC

4.1 Introduction

In Chapter 3, it was shown that the regularized modified filtered error least mean square algorithm (RMFeLMS) convergence properties as compared to the filtered-reference and filterer-error algorithm. It also offers possibilities for improvements of the robustness. However, the controller is model-based and is therefore sensitive for mismatch between the model and the plant. This model mismatch reduces the overall performance of the controller. Model mismatch can be caused by variations in parameters such as temperature, boundary conditions etc. Furthermore, it is assumed that the physical system can be modeled using a linear time-invariant system. This is not necessarily the case due to the non-linear behavior of the piezoelectric patches. In Ref. [18], a high-authority and low-authority control (HAC/LAC) architecture is described. In this book, the goal of the low-authority controller is to add active damping to the structure. Active damping can be implemented using different strategies (See [18]), but in this chapter a collocated and dual sensor-actuator pair is used. The product of the quantities measured and exerted by a dual sensor actuator pair is power, which is the case when for instance velocity is measured and force is exerted. In this case a simple feedback controller would be sufficient due to the fact that the overall energy that is stored in the system will be reduced [18]. An actively damped system will only have reduced resonance and anti-resonance peaks, which are the eigen-frequencies for the poles and zeros of the system, respectively. This will only lead to a reduction of transients from the system (See Ref. [18]). A consequence of this is that the effect of active damping reduces for frequencies that do not coincide with the poles and zeros. To gain further reductions for such frequency components a model-based

[0]Parts of this chapter have been submitted for active 2009 (See ref. [53])

controller can be used. This high-authority controller can be realized using for instance the RMFeLMS, FeLMS, FxLMS, LQG or any other model-based control algorithm. A major advantage of the HAC/LAC structure is that it increases the robustness and performance of a model-based controller.

The object researched in this chapter is a smart-panel. Most publications concentrate on either multiple-input and multiple-output (MIMO) realization [29] or on a structure that uses multiple Single-Input Single Output (SISO) with active damping [46–48]. The MIMO-based control architecture is less robust under parametric uncertainty making it necessary to use a relatively high regularization level to prevent instability. This comes at the price of reduced performance. A SISO controller that uses a collocated-dual sensor-actuator pair can be realized using a simple feedback controller. However, if the gain of the controller is too high the panel behaves as if it is fixed at the location of the sensor-actuator pair. This results in a panel that has new but higher resonance modes thereby reducing the performance of the system. The use of multiple collocated-dual sensor-actuator pair has the advantage that there is no interaction between different loops due to the fact that overall energy in the system is reduced, thereby guaranteeing stability. However, a possible problem with this configuration is that the SISO sensor-actuator pair is not fully dual and collocated, making it necessary to carefully design a controller to prevent spillover or instability (See Ref. [47]). In the panel presented in this chapter, the LAC loop uses piezoelectric patch actuators and acceleration sensors which are not fully dual and collocated [47]. The piezoelectric patch doe not generate a force but line moments at the edges (See for instance [16, 18]). In [64], it was shown that this isn't necessarily a problem for lower frequencies. In this publication, a beam with a combination of a piezoelectric actuator and an acceleration sensor was evaluated extensively. The conclusion of this publication was that the collocated and dual property seems to be valid up to a certain frequency, ensuring that the phase of the plant will be in the range of -90 to +90 degrees up to this given frequency. The system therefore requires additional roll-off at higher frequencies preventing instability at higher frequencies.

In this chapter, results are presented of an actively controlled panel (smart-panel) that uses active damping (LAC) in combination with the RMFeLMS-MIMO based algorithm (HAC). The LAC structure uses a local high-speed digital controller running at 100 kHz. This structure uses a piezoelectric actuator and a piezoelectric sensor for the high-authority controller and an acceleration sensor for the low-authority controller. The HAC/LAC structure shares the same actuator for both loops. The sensor-actuator pairs of the LAC unit are not fully dual, but as stated before this does not have to be a problem if additional roll-off is added at higher frequencies. The HAC structure is realized as a model-based controller running at a lower-sample rate, in this case 2 kHz, resulting in a control bandwidth of 1 kHz. In this chapter, the results of the experiments are presented and the impact of the HAC/LAC structure on convergence rate, performance and robustness is evaluated.

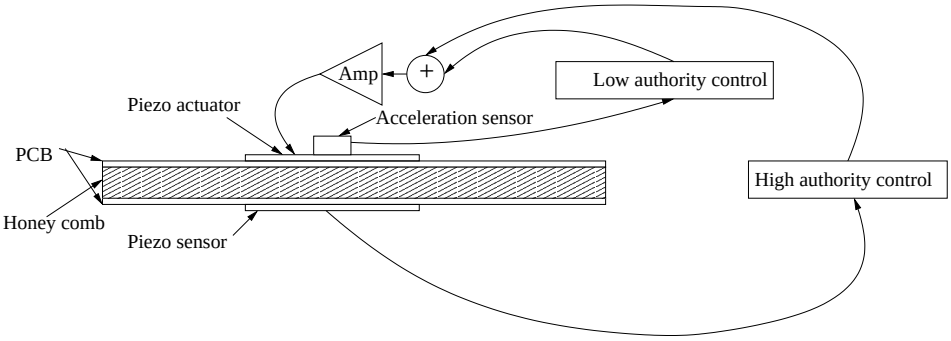


Figure 4.1: *Overview of the high-authority/low-authority control architecture of the first generation panel.*

4.2 The panel

In the experiments described in this chapter, two different panels were evaluated. The first panel consists of a sandwich of three layers. The outer layers consist of printed circuit boards; the middle layer consists of a honeycomb structure. These boards use piezoelectric sensors and piezoelectric actuators. This panel does not contain any conditioning electronics and does not contain acceleration sensors. Therefore, it was necessary to mount external acceleration sensors and use external signal conditioning electronics. The acceleration sensors were glued onto the panel using X60. An overview of this panel can be found in Figure 4.1. The acceleration sensor is directly mounted on top of the piezoelectric patch actuator.

The second panel is a fully integrated panel that includes all signal conditioning electronics. It was constructed as a sandwich panel, with printed circuit boards as the outer layers, and a foam layer in between. The two outer layers contain all signal conditioning electronics. The piezoelectric patches are mounted on the outside of the panel and the acceleration sensors are mounted inside the panel. The acceleration sensor is a micromechanical system (MEMS), which was selected due to its small form factor and small height. The panel only requires a controller and a power supply. An overview of this structure can be found in Figure 4.2.

Both panels were extensively tested. However, the panel with the integrated acceleration sensor had a relatively large phase shift. This was then compared to a panel with an externally attached acceleration sensor. The phase shift of this panel was smaller. The result presented in this chapter therefore use the panel with the separate acceleration sensors and no integrated signal conditioning electronics.

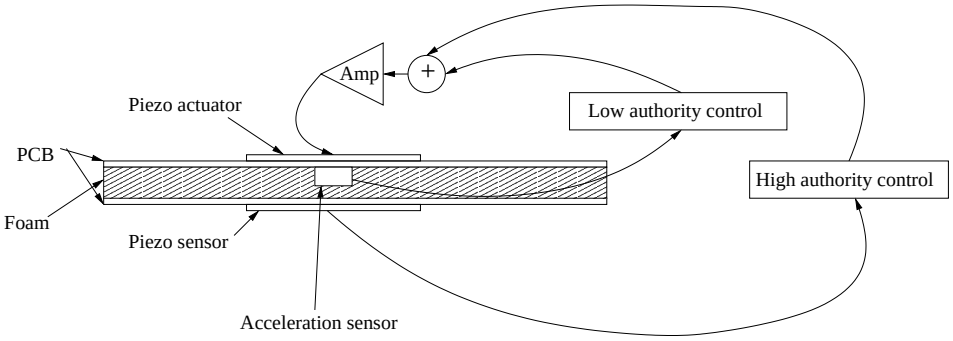


Figure 4.2: *Overview of the high-authority/low-authority control architecture and the second generation panel. The signal conditioning electronics is integrated on the panel, however the controller is still a separate unit.*

4.3 The low-authority controller

In this chapter, active damping is used to realize the LAC loop. In most publications, active damping is realized using an analog controller (See [46–48]). One of the advantages of an analog controller is its low delay when compared to a digital controller. In this thesis, a method is described that uses a digital controller with a high-sample rate that approximates an analog controller, such that he analog and digital controller have identical performance for frequencies within the control bandwidth. The control architecture designed during the research period provided a suitable environment for realization of such a high speed digital feedback controller. The interfacing board already contained ADDA converters at 100 kHz while the actual RMFeLMS algorithm was operating at 2 kHz. This made it possible to run the LAC and HAC controller at different sample rates. The interface between the PC that runs the HAC algorithm and the ADDA unit was implemented in a FPGA. The FPGA incorporates the following functional units: the decimation filters,the interpolation filters, the glue logic for the PCI bus interface and the low-authority controller. The design of the decimator filter and interpolator filter is described in Chapter 5.

4.3.1 The analog low-authority controller

The velocity signal as required for the low-authority controller was obtained by integration of the acceleration signal. This results in the following controller:

$$gD(s) = g/s, \tag{4.1}$$

in which g is the loop gain and $D(s)$ is the controller. In practice, such an integrator can not be realized, but it can be approximated by a low-pass filter with a

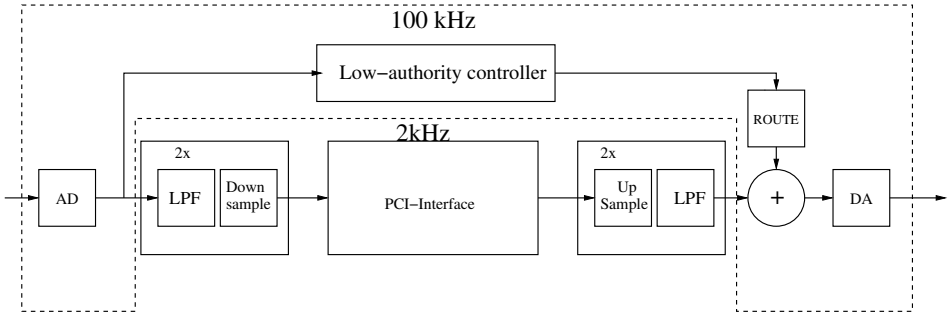


Figure 4.3: *Simplified overview of the control system containing the low-authority controller.*

sufficiently low cut-off frequency:

$$gD(s) = \frac{g}{\tau s + 1}. \tag{4.2}$$

In [47] a version was used that combines an integrator with a phase-lag compensator. This was necessary to prevent instability at certain frequencies. In our application it was necessary to reduce the loop gain for higher frequencies to prevent instability and spillover. This was realized by adding an additional low-pass filter with a cut-off of 1 kHz.

4.3.2 The digital low-authority controller

An overview of the controller and the implementation of the low-authority controller can be found in Figure 4.3. All parts within the dashed lines are running at 100 kHz. This includes the AD, DA and the low-authority controller. The low-authority control unit consists of three concatenated second-order infinite impulse response (IIR) sections which are independently programmable for each channel. A multiplier can be used to scale the input or output of each IIR section. Each input has a fixed low-authority controller assigned to it. However, each output can select its own input channel. This task is performed by the route unit.

The implemented architecture of the low-authority controller can be found in Figure 4.4. It consists of three IIR second-order filters and a multiplier. The position of the multiplier can be selected to be at the input, or at the output of any of the three IIR filters. In this architecture, the three second-order IIR filters and the multiplier always form a concatenation. The overall architecture processes all 16 channels sequentially. An IIR-unit is realized as the state-space implementation (See Eq. (4.4)). In this architecture, only one multiplier and adder is used to process all 16 channels, thereby reducing the hardware costs. An IIR-section can be bypassed by programming the $a_0 = 1$ and $b_0 = 1$ (Eq. (4.4)) and all the other coefficients with zero. The position of the multiplier is fixed for all 16

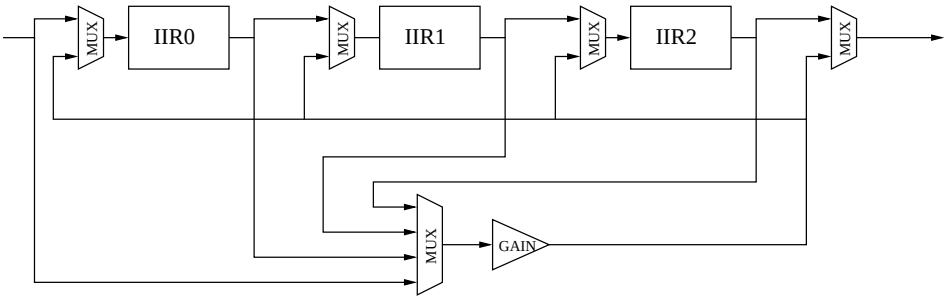


Figure 4.4: *The internal structure used for the low-authority controller. It consists of of three 2^{th}-order IIR filters and a multiplier that can be inserted at the output or input of one of the IIR sections.*

channels and can only be programmed once at the start of the sampling process. The multiplier can be used to add additional gain to the overall system. However, to prevent overflow, the position of the multiplier must be carefully selected. Each channels has its own gain factor.

The second order-section uses a state-space realization. This structure was implemented using fixed-point arithmetic. The coefficients are in 32 bits format with 30 fractional bits and 2 integer bits resulting in a range from -2 to almost 2. For stable minimum-phase filters, with the number of zeros being equal to or smaller than the number of poles (See page 56 in the textbook of Elliott[20]), this is sufficient. The equation for a second-order system is

$$a_0 y(n) + \sum_{i=1}^{2} a_i y(n-i) = \sum_{j=0}^{2} b_j u(n-j), \tag{4.3}$$

in which a_i and b_j are the filter coefficients. This can be written in the following form:

$$y(n) = \sum_{j=0}^{2} \frac{b_j}{a_0} u(n-j) - \sum_{i=1}^{2} \frac{a_i}{a_0} y(n-i), \tag{4.4}$$

in which all the coefficient are normalized by dividing them by a_0. A stable system only has poles inside the unit circle and a minimum-phase system ensures that all the zeros are inside the unit circle. A consequence of this is that the coefficients are in the range from -2 to 2. The proof is as follows. First, each point c inside the unit circle has the property $|c| \leq 1$ given that $c \in \mathbb{C}$. For a real system, with real signals, each pole or zero also has a complex conjugate. This means that for

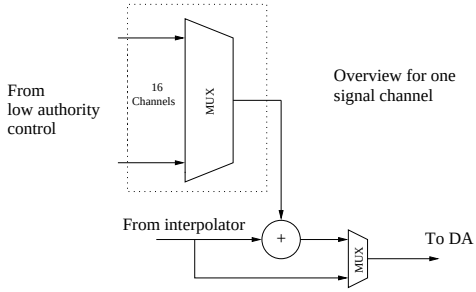


Figure 4.5: *The function of the route unit is to add a given input to an output.*

a second-order section the following holds:

$$\begin{aligned} H(z) &= \frac{(z-n)(z-\bar{n})}{(z-p)(z-\bar{p})} = \frac{z^2 - nz - \bar{n}z + n\bar{n}}{z^2 - pz - \bar{p}z + p\bar{p}} \\ &= \frac{z^2 - 2\Re\{n\}z + |n|^2}{z^2 - 2\Re\{p\}z + |p|^2} = \frac{b_0 z^2 + b_1 z + b_2}{a_0 z^2 + a_1 z + a_2}. \end{aligned} \tag{4.5}$$

From this it can be concluded that a each pole or zero pair always has real coefficients and that each coefficient is always smaller than or equal to 2.

The LAC-unit can be used to add additional gain to the loop. This was realized using a digital multiplier. The gain factor is a 32 bits fixed-point number with 16 integer bits and 16 fractional bits. The multiplier generates a 48 bits result. This result is truncated and saturated to a 16 bits result. All functional units require 16 bits input values and generate 16 bits output values.

Each LAC unit requires an input and output. The input to which each LAC-unit is assigned is fixed, however the output to which a LAC unit is assigned can be selected. This means that one input can be added to several outputs but not vice versa. An overview of the route unit can be found in Fig. 4.5. In the implementation on the FPGA the route unit is integrated into the interpolator. The values from the different low-authority control units are written into a buffer. In the final stage of the interpolation unit a buffer location is selected and added to the currently processed output channel. It is also possible to bypass the add operation. Some input channels may not require local feedback, for instance the reference input for the RMFeLMS algorithm.

In Section 4.3.1, an analog prototype filter was described. The technique for designing a digital low-authority controller is to translate the analog s-domain prototype filter into a digital z-domain filter using the bilinear transformation. This transformation from the s-domain to the z-domain can be performed by substitution of

$$s = \frac{2}{T}\frac{z-1}{z+1}, \tag{4.6}$$

into the transfer function, with T being the $10\mu s$ sample interval. The cut-off

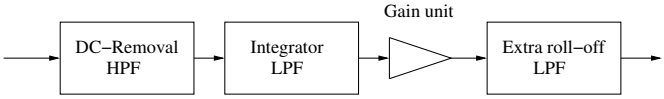


Figure 4.6: *An overview of the implemented low-authority control algorithm. Due to design constraints the order of the units is important.*

frequencies used in this chapter were relatively low when compared to the Nyquist frequency. Therefore, no pre-warping was needed.

Active damping is realized using a collocated dual pair and a simple gain. For the setup used and for the experiments described in this chapter, this was difficult to realize. This was especially the case at higher frequencies. Experimental results showed that an additional roll-off for frequencies above 1 kHz was sufficient to minimize spillover and increase stability. The acceleration sensor can measure static as well as dynamic gravitational influences. This means that the output already has a DC offset determimed by gravitation. The output signal of the acceleration sensor in the case of the smart-panel is considerably smaller for low frequencies than for high frequencies. These differences become less extreme after integration of the acceleration signals. The resulting velocity signal is a better quantity for scaling.

From the above, the four basic operations for the low-authority controller can be determined. First, the DC-component from the acceleration sensor must be removed using a simple high-pass filter. This prevents saturation of the integrator. Secondly, the signal from the acceleration sensor must be integrated using a simple low-pass filter. The cut-off frequency was selected to be at 10 Hz so that the pole was not too close to the unit circle in the z-plane, thereby preventing numerical instability of the IIR-filter. Thirdly, the output of the previous stage needs to be multiplied for introduction of additional loop gain. Finally, additional roll-off must be generated for higher frequencies. In the panel used it was concluded that a simple low-pass filter with a cut-off 1 kHz was sufficient to realize a stable control system. The overall structure of this algorithm can be found in Figure 4.6. The data-path is organized in such a way that for the application described in this thesis the numerical precision is sufficient.

4.4 Measurements

The results of a controller that combines the low-authority and high-authority control are described in this section. In this chapter only the result of the first generation panel is presented. The measurements of the fully integrated panel were too poor to obtain usable results. Therefore, it was decided to use the first-generation panel and mount external acceleration sensors.

4.4.1 The low-authority control loop

The overall control-system uses two different loops as depicted in Figure 4.1. In this section, the transfer function of the low-authority controller is studied.
For a fully collocated and dual system having a phase shift in the range from -90 to +90 degrees, the gain can be made arbitrarily large, at least in theory. In a HAC/LAC architecture this is not desirable. In this case only damping is required. If the gain becomes infinite the panel has new resonance modes which the HAC architecture can not control. Furthermore, a piezoelectric patch actuator combined with an acceleration sensor is not fully dual and collocated, which reduces the maximum loop gain. In Ref [47] it was shown that a piezoelectric patch actuator and an acceleration are dual and collocated up to a certain frequency. For higher frequencies this behavior does no longer apply. Therefore the gain of the system is limited and measures must be taken to introduce extra roll-off at higher frequencies.

In the first experiment the quality of the low-authority control loop up to 1 kHz was studied. For this measurement the "'SigLab 20-42"' network analyzer was used. The transfer function was measured from the piezoelectric actuator to the acceleration sensor. The high voltage amplifier driving the piezoelectric actuator had a 40 dB gain. However, the input was attenuated with 10 dB resulting in a overall gain of 30 dB. The acceleration sensors used were the 4393V of Brüel & Kjaer. These sensors have a bandwidth of 16.5 kHz at which it has a phase shift of +10 degrees. The acceleration sensors were connected to charge amplifiers made by TNO. A second set of amplifiers was used to increase the output level of these charge amplifiers. The gain of the second amplifiers was 40 dB. The results of these measurements can be found in Figure 4.7. For frequencies higher than approximately 50 Hz, the panel has a phase shift in the range of 0 to 180 degrees. However, this phase shift applies for the acceleration signal and not for the velocity signal. After integration, the phase shift is in the range of -90 to +90 degrees. In Figure 4.7, only the transfer functions for the collocated-dual actuator-sensor pairs are plotted. The influence of non-collated actuators on non-collocated sensors can be assumed to be minimal due to the property that the overall energy in a collocated and dual system decreases. Therefore, a collocated dual system using several SISO-loops that are mechanically coupled is asymptotic stable. The cross-coupled sensor-actuator pairs can not destabilize such a system, and are therefore not plotted in Figure 4.7.

In the second experiment the transfer function of the panel up to 10kHz was studied. This measurement was using a similar setup as that in the first experiment. The result of this measurement can be found in Figure 4.8. From the second measurement it becomes clear that phase shift is no longer in the range of -90 and +90 degrees for frequencies above 2.5 kHz. This makes it necessary to reduce the overall loop gain and to introduce extra roll-off for frequencies above 2.5 kHz.

From this measurement it can be concluded that the piezoelectric actuator and the acceleration sensor behaved dual and collocated for frequencies in the range of

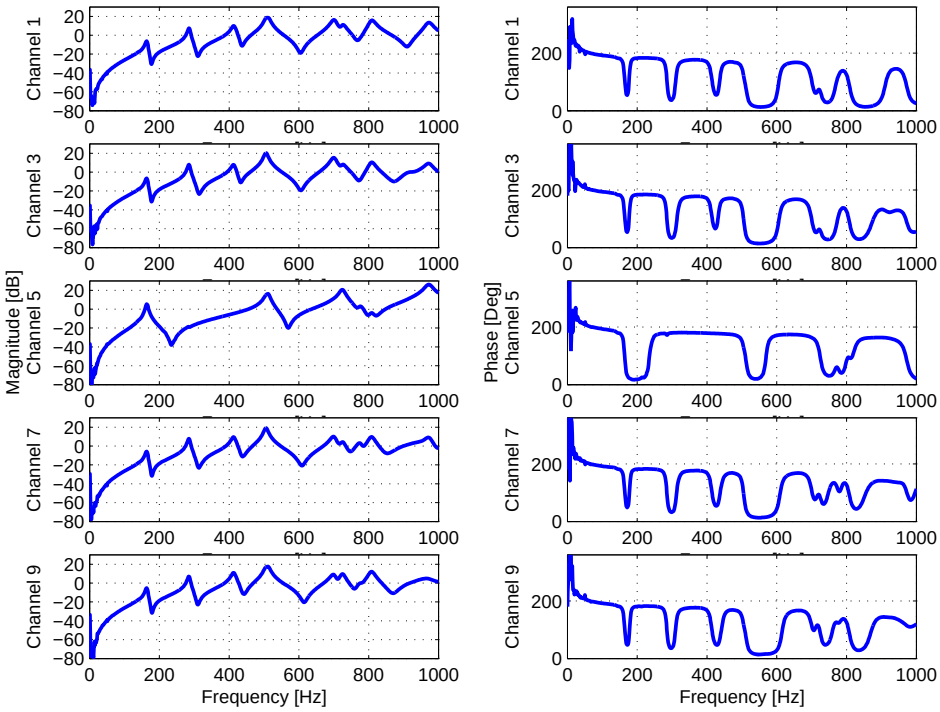


Figure 4.7: *The transfer functions for the collocated sensor-actuator pairs from 0 to 1 kHz; piezoelectric actuator to acceleration sensor.*

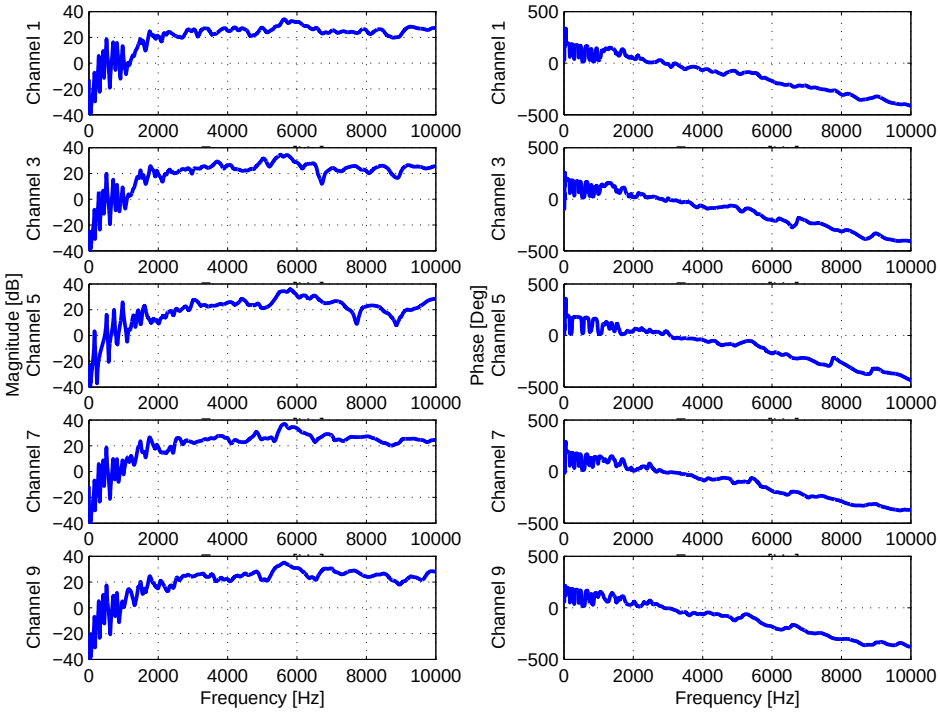


Figure 4.8: *The transfer functions for the collocated sensor-actuator pairs from 0 to 10 kHz; piezoelectric actuator to acceleration sensor.*

50 Hz to 2.5 kHz. The use of a piezoelectric patch actuator limits the bandwidth due to the fact that it does not exert forces but moments, for low frequencies the difference is almost non-existing (See ref [47]). The size of the piezoelectric patch actuator also introduces additional phase shift. The trade-off in this case is between the maximum moment that can be generated by a patch and the phase shift present in the patch.

4.4.2 The transfer function under low-authority control

In this section, the influence of the low-authority controller on the high-authority control loop is described. The transfer function of the high-authority control path was measured, while the low-authority controller was active. The input and output signals were then used to identify the plant using a sub-space identification technique. The plant is also needed for high-authority model-based controllers and for internal model control.

In this section, two different scenarios are compared. In the first scenario, the loop is closed from the piezoelectric actuator to the acceleration sensor and the transfer function is measured from the piezoelectric actuator to the piezoelectric sensor. In the second scenario, the loop is closed from the piezoelectric actuator to the acceleration sensor and the transfer function is measured from the piezoelectric actuator to the acceleration sensor. The advantage of the second scenario is that only one sensor is needed, in this case acceleration. The loop gain of the low-authority controller is adjusted to prevent instability of the overall plant. The acceleration sensor signal is amplified with a gain of 40 dB which reduces quantization artifacts and improves the signal integrity and quality. The setup used to realize the LAC architecture was described in Section 4.3.2. This architecture uses a digital control loop gain unit. The digital loop gain was set using three different configurations: 64 times for all sensor-actuator pairs , 64 times for the middle sensor-actuator pair, 32 times for the outer sensor-actuator pairs, and 32 times times for all sensor-actuator pairs. The integrator used in the LAC unit had a cut-off of 10 Hz followed by a low-pass filter with a cut-off of 1 kHz, which is needed to realize additional roll-off for higher frequencies. The controller does not use a decimation filter. The interpolator as described in Chapter 5 was programmed with an interpolator filter with a stopband attenuation of 50 dB. The first section had a length of 63 taps, increasing the sample rate with a factor 5 and the second section had a length of 123 taps, increasing the sample rate with a factor 10. The influence of the low-authority controller on the transfer function up to 1 kHz can be found in Fig. 4.9. The Variance Accounted For (VAF) of the estimated plants are for the configuration that uses a gain of 64 for all patches, a gain of 64 for the middle patch and a gain of 32 for the outer patches, a gain of 32 for all patches and finally without any low-authority control which results in 99.96%, 99.98%, 99.97% and 99.98% VAFs, respectively. Notice that the zeros of the transfer function shift to the right for the system with low-authority control. It is also clear that the magnitude of the poles and zeros becomes smaller under

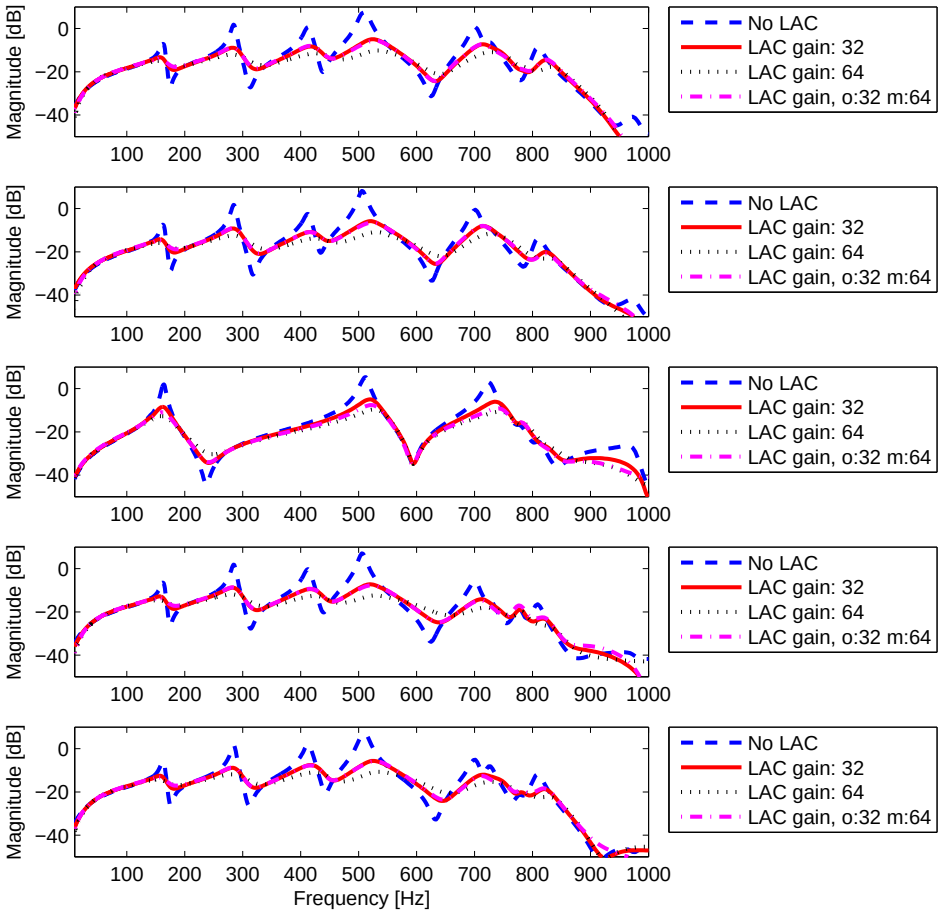


Figure 4.9: *Influence of the low-authority controller on the transfer function of the plant measured from the piezoelectric actuator to the piezoelectric sensor, while the feedback loop was from the actuator to the acceleration sensor. A decimation filter was not used; the interpolation filter had a stopband attenuation of 50 dB.*

the influence of the low-authority control loop. Apparently, the low-authority controller is able to add damping to the system. In a second measurement the transfer functions for frequencies up to 5 kHz were obtained. In this case, no decimation filter was used; and an interpolation filter with a stopband attenuation of 50 dB was used. The lengths of the interpolation filters used were 63 taps for both sections. The first section increases the sample-rate by a factor of 2 and the second section increases the sample-rate by a factor of 5. The transfer function was again estimated using a sub-space identification method. The influence of the LAC architecture on the transfer function up to 5 kHz can be found in Figure 4.10. The order of the state-space model is 145. The VAFs for the low-authority controller with a digital loop-gain of 64, with a loop-gain of 32 on the outer sensors and 64 for the middle sensor, with a digital loop-gain of 32 for all the sensors and finally no low-authority control are 99.75%, 99.79%, 99.80% and 99.90%, respectively. The phase-shift of the LAC loop for frequencies above 1 kHz is not within the -90 to +90 degrees range. In Figure 4.10, it can be clearly seen that spillover for frequencies in the range of 2 to 3 kHz appears for high digital loop gains. The gain therefore must be limited and extra roll-off at frequencies above 1 kHz is needed to prevent instability. Several filter combinations were tested but did not yield the expected results. The approach proposed in [47] only resulted in a system with a lower loop gain and a degraded overall performance. A first-order low-pass filter with a cut-off of 1 kHz resulted in minimum spillover. The digital loop gain was reduced to 32 for the outer piezoelectric patches and 64 for the middle piezoelectric patch. With this configuration, spillover for frequencies in the range from 2 kHz to 3 kHz was minimal while the damping was maximized. Furthermore, the spectral excitation in the spillover region was also minimal, making this configuration a good compromise.

In the previous configuration, two sensors were needed, one for the low-authority control loop and one for the high-authority control loop. This results in good performance however it is quite expensive. The goal of another set of experiments was to see if it was possible to obtain good results using only an acceleration sensor, i.e, the HAC and LAC loop both used the same piezoelectric actuator and acceleration sensor. The plant was estimated using a sub-space identification method. The spectral output of the acceleration sensor contains dominant frequency components above the 2 kHz sample rate of the HAC controller. This resulted in a large error in the estimated state-space realization when no decimation filter was used. This made it necessary to use a decimator at the input of the controller. The decimation filter had a stopband attenuation of 50 dB; both decimation sections had a length of 63 taps. The interpolator used two sections, one with 123 taps and one with 63 taps, the stopband attenuation was also 50 dB. The order of the estimated models was 50 at a 2 kHz sample rate. The VAF for the model with a gain of 64 was 99.90%, for a gain of 64 for the middle piezoelectric patch and 32 for the outer piezoelectric patches it was 99.96%, for a gain of 32 it was 99.94% and finally for no feedback at all it was 99.95%. The resulting transfer functions of this measurement can be found in Figure 4.11.

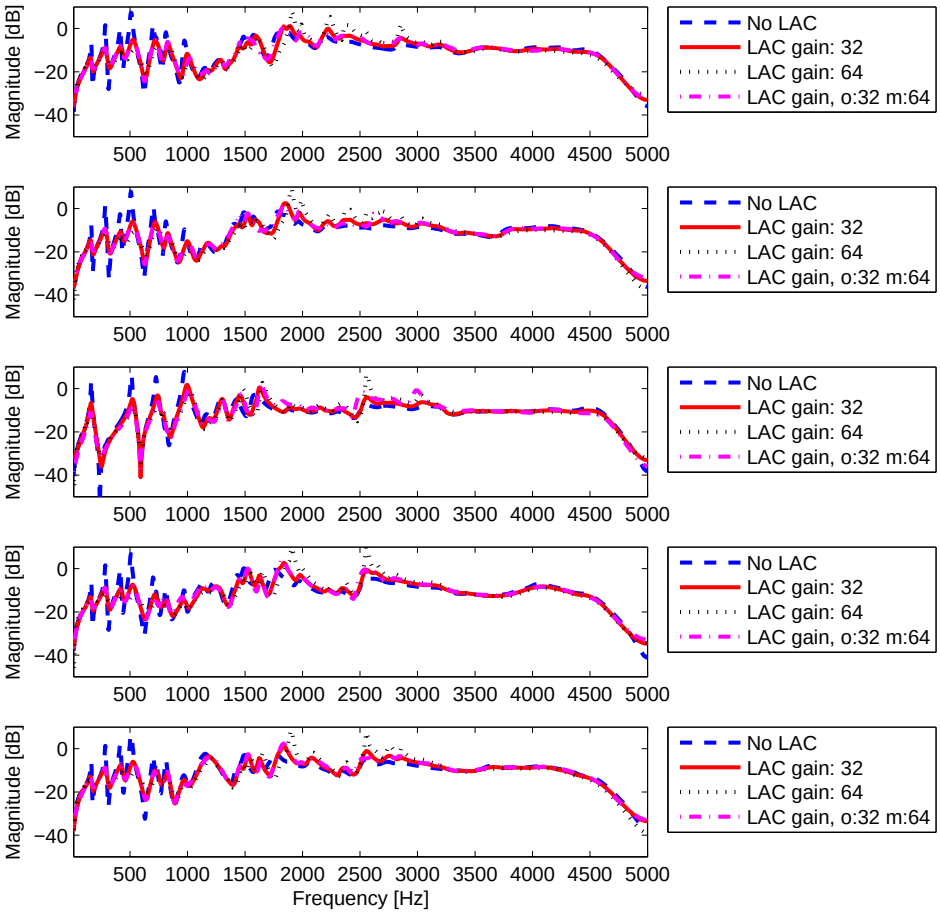


Figure 4.10: *Influence of low-authority controller on the transfer function of the plant, as measured from the piezoelectric actuator to the piezoelectric sensor, while the feedback loop was from the actuator to the acceleration sensor. No decimation filter and an interpolation filter with a stopband attenuation of 50 dB was used.*

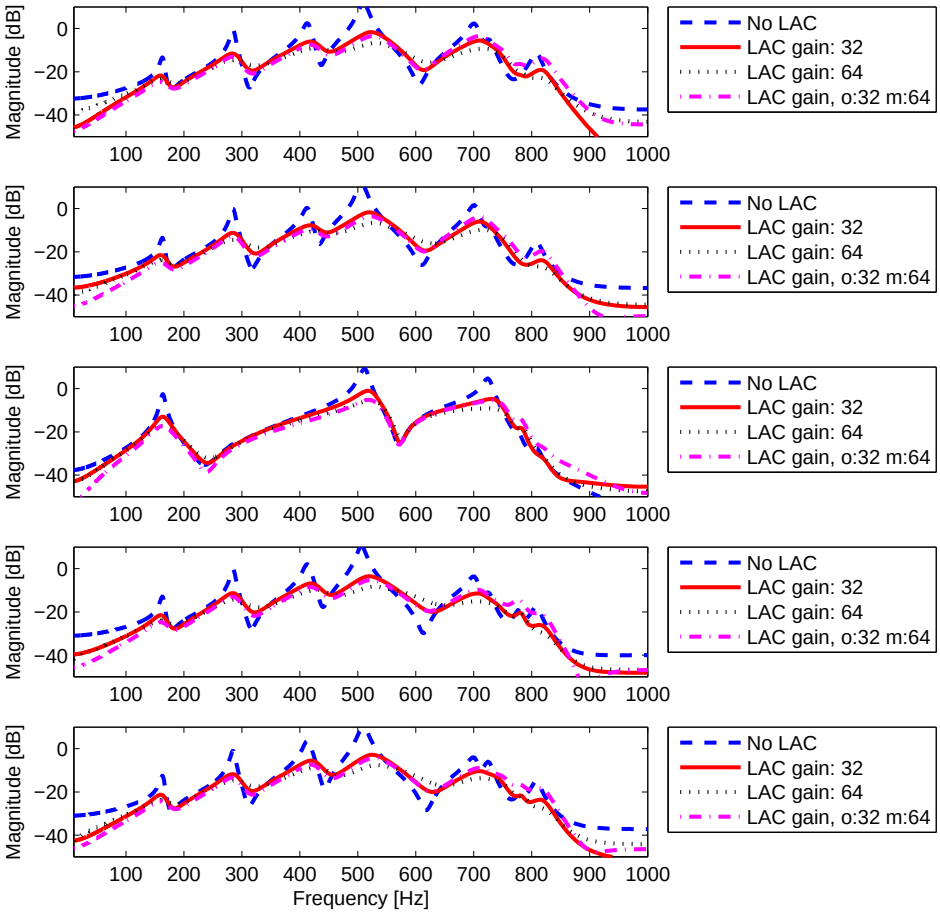


Figure 4.11: *Influence of low-authority controller on the transfer function of the plant, as measured from the piezoelectric actuator to the acceleration sensor, while the feedback loop uses the same path. Decimation and interpolation filters with a stopband attenuation of 50 dB were used.*

In a next experiment, the influence of spillover for the proposed piezoelectric/acceleration pair was evaluated. A sub-space identification method was again used to estimate the model. The order of this model was 145 at a 10 kHz sample rate. The decimation filters used a filter with a 50 dB stopband attenuation and consisted of two sections with 63 taps each. The interpolation filters used a filter with a 50 dB stopband attenuation and consisted of 2 sections with 63 taps. The VAF for a gain of 64 was 99.71%, for a gain of 64 for the middle sensor and 32 for the outer sensors it was 99.64%, for a gain of 32 it was 99.60% and finally for no low-authority control it was 99.60% The resulting transfer functions for this measurement can be found in Fig. 4.12.

In this section, the influence of low-authority control on the HAC transfer functions was shown. It became clear that a too high loop gain either resulted in instability or in spillover at higher frequencies. From the Figures 4.10 and 4.12, it is seems that the best gain is 32. However, such a loop gain is somewhat conservative, which results in poor reductions. A good compromise is to use different gains for each sensors-actuator pair. For this case, a gain of 64 for the middle sensor and 32 for the outer sensors seems to be a reasonable compromise. Two configurations were studied: one using two sensors and the other one using only one sensor. Although the one-sensor configuration seems to be very cheap it is not a good choice. The reason for this is that the second configuration always requires a decimation filter. The decimation filter is required to prevent instabilities for the RMFe algorithm. This is probably due to aliasing of high frequency components generated by the acceleration sensor, which are within the 50 kHz bandwidth of the LAC-unit but outside the 2 kHz bandwidth of HAC algorithm. This reduces the performance for feedback applications quite severely. Therefore, the first configuration using two sensors is used in the remainder of this chapter. This configuration results in good damping performance while spillover is reasonable.

4.4.3 Low-authority and high-authority control

In this section, the influence of the low-authority controller on the high-authority control algorithm is evaluated. Two different scenarios were studied. The first one uses the RMFeLMS algorithm in combination with internal model control to construct a feedback model-based controller. The second one uses the RMFeLMS in a feedforward scenario.

In the first experiment, the high-authority controller uses the RMFeLMS algorithm in combination with IMC to construct a feedback controller. For the low-authority controller, a setup was used as described in Section 4.4.2. In this case the loop gain was 64 times for middle patch, 32 times for the outer patches. The following parameters were used for the feedback controller: the RMFe regularization level β was to set -20 dB, the LMS step size to $\alpha = \frac{1}{80}$, the length of time-reversed transpose inner-factor was $N_D = 80$, the length of adaptive filter was $W_i = 80$ and the leakage was set to $\gamma = 10^{-5}$. The feedback controller did not

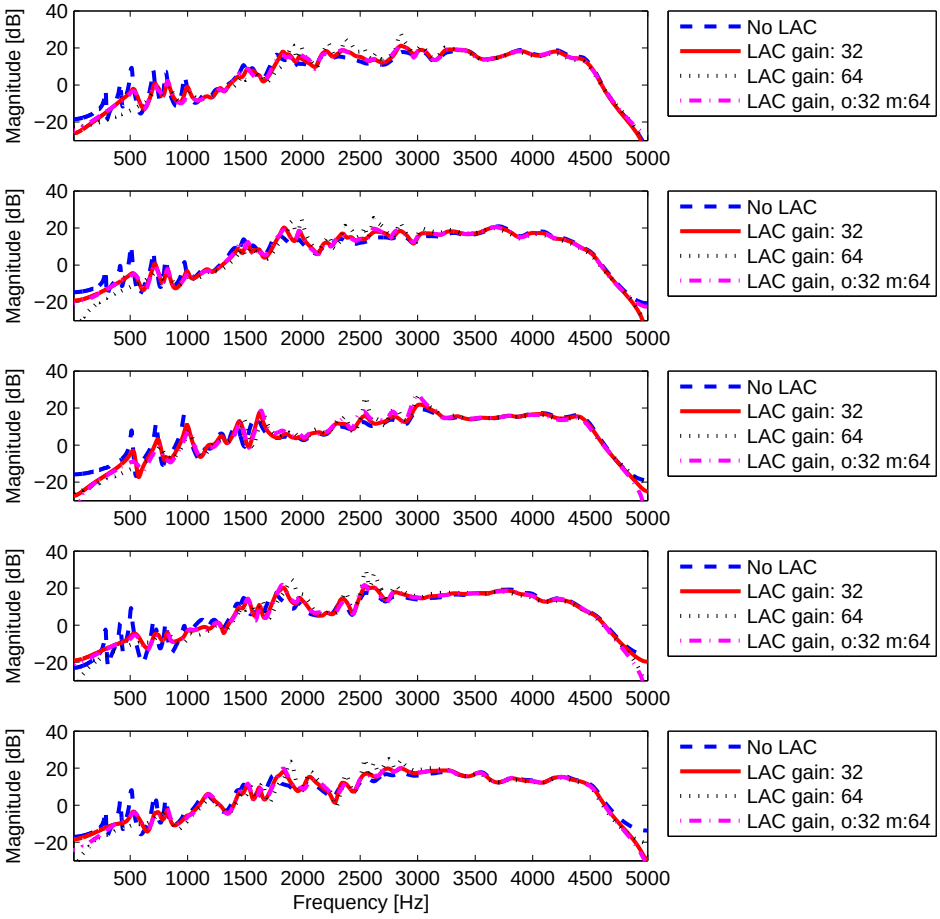


Figure 4.12: *Influence of low-authority control on the transfer function of the plant, as measured from the piezoelectric actuator to the acceleration sensor, while the feedback loop uses the same path. Decimation and interpolation filters with stopband attenuation of 50 dB were used.*

use a decimation filter and the interpolation filter had a stopband attenuation of 50 dB. The LAC algorithm is described in Section 4.3.2. The parameters for the LAC algorithm are the same as those of Section 4.3.2. The transfer functions of the panel with and without LAC can be found in Figure 4.9. The adaptive controller converged for a duration of three minutes. This time was sufficient for the controller to reach the steady-state. For longer periods no further improvements were measured, measurement on the convergence as presented in the next section show that the controller typically needs less than a minute to converge. The test involved MIMO-control only, LAC control on, LAC+MIMO control and finally no control at all. The order of the plant with LAC off was 50 and the VAF was 99.97%. The order of the system with LAC on was 50 and the VAF was 99.97%. The result of this measurment can be found in Fig. 4.13.

In the second experiment, the feedforward controller was studied. In this case the model order was again 50 and the VAF was 99.98%. The RMFe regularization level β was set to -30 dB, the LMS step size was $\alpha = \frac{1}{40}$, the length of time-reversed transpose inner-factor was $N_D = 80$, the length of the adaptive filter $W_i = 350$ and the leakage was $\gamma = 10^{-5}$. The feedforward controller took its reference signal directly from the signal generator. The result of this measurement can be found in Fig. 4.14.

It can be concluded that LAC reduces the MSE for feedforward control as well as for feedback control.

4.4.4 Convergence speed under influence of LAC

In this section, the influence of LAC on the convergence speed of the RMFeLMS algorithm is evaluated. First the parameters must be found at which the best possible reduction could be obtained. These were different for feedback and feedforward control. The goal of these experiments was to find the optimal adaptation speed given a certain step size α. In the next paragraph the results of these experiments are given.

The maximum reduction for a feedback controller depends on different variables. From previous experiments it was concluded that filter lengths $W_i = 80$ and $N_D = 80$ were good candidates for a feedback controller. This still leaves the possibility to optimize the regularization level β and leakage parameters γ. In several experiments, it was found that a leakage γ of 10^{-5} was a good compromise between robustness and performance of the LMS algorithm. The setting of the regularization-level β depends on the plant. The magnitude of the plant diagonal elements that was used in this measurements can be found in Figure 4.11. A regularization level β of -20 dB was selected. In the case of the RMFe structure, this resulted in a regularized plant model in which the magnitude can not get below -20 dB. The only tunable parameter left is the convergence parameter, the step size α of the LMS algorithm. In an experiment the influence of this parameter was studied. The error signals were measured for a duration of 60 seconds. At the beginning of each measurement the controller was switched off for 5 seconds and

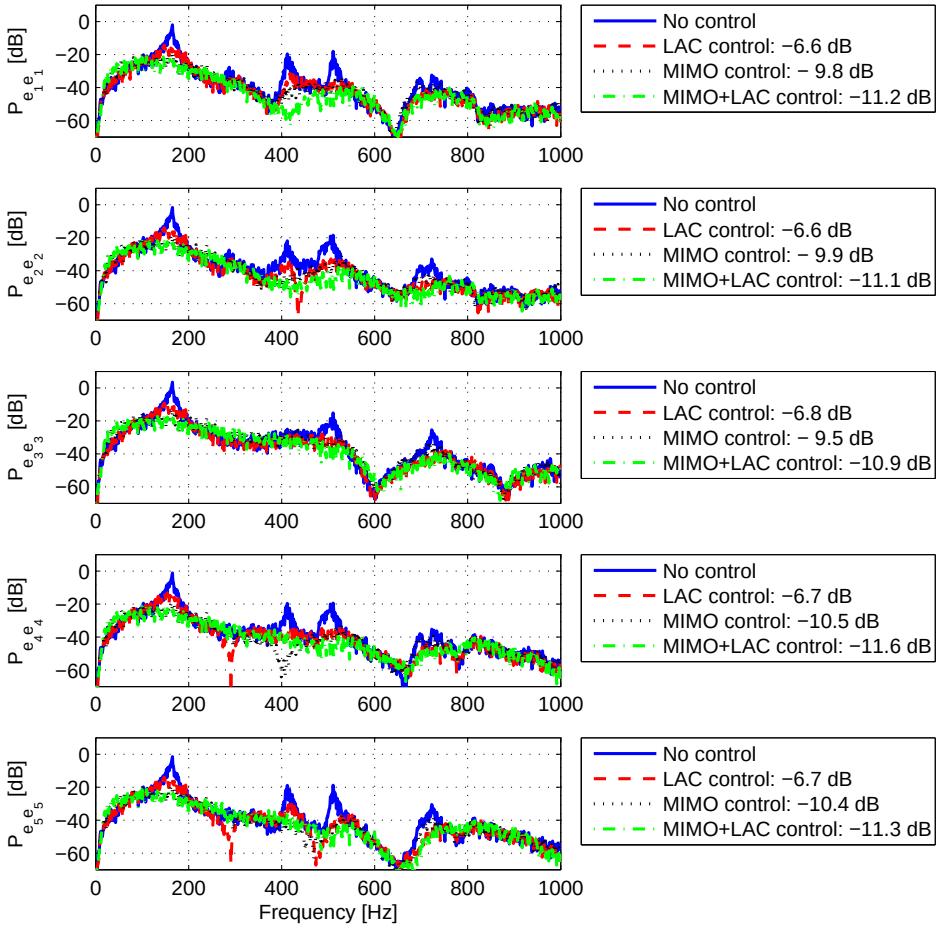


Figure 4.13: *Performance measured on the sensors for a feedback controller using IMC.*

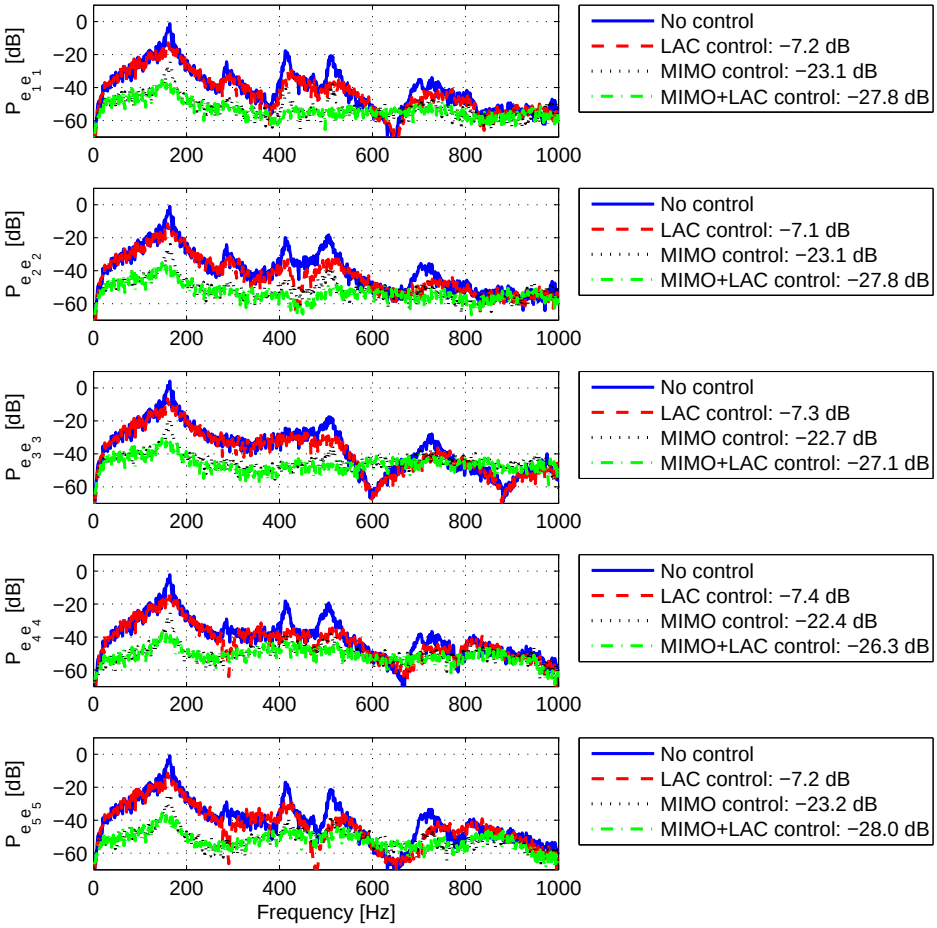


Figure 4.14: *Performance of the feedforward controller.*

then it was switched on for 55 seconds. The reduction was calculated by taking the first 4 seconds and the last 4 seconds controller switched off and controller switched on,respectively. This procedure was repeated 32 times and an ensemble average was calculated in order to obtain consistent and reproducible results. In

Table 4.1: *Reduction of the error signals for a feedback controller for different step sizes α after 60 seconds*

α	$\frac{1}{10}$	$\frac{1}{20}$	$\frac{1}{40}$	$\frac{1}{80}$	$\frac{1}{160}$	$\frac{1}{320}$
MSE reduction	7.8 dB	9.0 dB	9.4 dB	9.3 dB	9.0 dB	8.4 dB

Table 4.1, the result for a number of different step sizes α is given. A plot of the different convergence curves was not included due to the fact that the curves were almost identical: only the MSE errors were different. From Table 4.1, it can be concluded that $\alpha = \frac{1}{40}$ gives the best possible reduction after 60 seconds.

In the next experiment, the convergence behavior of the RMFeLMS algorithm in a feedforward scenario was evaluated. The parameter selected for the feedforward RMFeLMS algorithm were the filter lengths $W_i = 350$ and $N_D = 80$. The leakage γ was set to 10^{-5}, being a compromise between stability and robustness. The regularization level was selected to be -30 dB. The procedure used to measure these results was the same as used in the feedback scenario described in the previous paragraph, however only 16 iterations were used. The different MSE values after convergence for different values of α can be found in Table 4.2. The conver-

Table 4.2: *Reduction of the error signals for a feedforward controller using different step sizes α after 60 seconds*

α	$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{5}$	$\frac{1}{10}$	$\frac{1}{20}$	$\frac{1}{40}$	$\frac{1}{80}$
MSE reduction	19.4 dB	21.2 dB	22.1 dB	22.3 dB	22.4 dB	22.6 dB	21.8 dB

gence curves for different values of α can be found in Figure 4.15. This Figure was constructed using 16 iterations over which the MSE was calculated. From these results, it was concluded that an α of $\frac{1}{40}$ was a good trade-off between convergence speed and steady-state MSE.

The question that arises from this is how LAC influences the overall performance in terms of speed of convergence. From the previous experiments it was decided to set α to a value of $\frac{1}{40}$ for the feedback controller as well as for feedforward controller. These values were selected to obtain good MSE values in combination with good convergence properties. The first case which was studied was the feedback controller. The LAC unit was configured as in Section 4.3.2 and 4.4.1. The convergence under influence of LAC using a feedback HAC strategy can be found in Figure 4.16. To show the impact of the LAC clearly, it was decided to leave the LAC switched off for the first 5 seconds and switched on for the remainder of

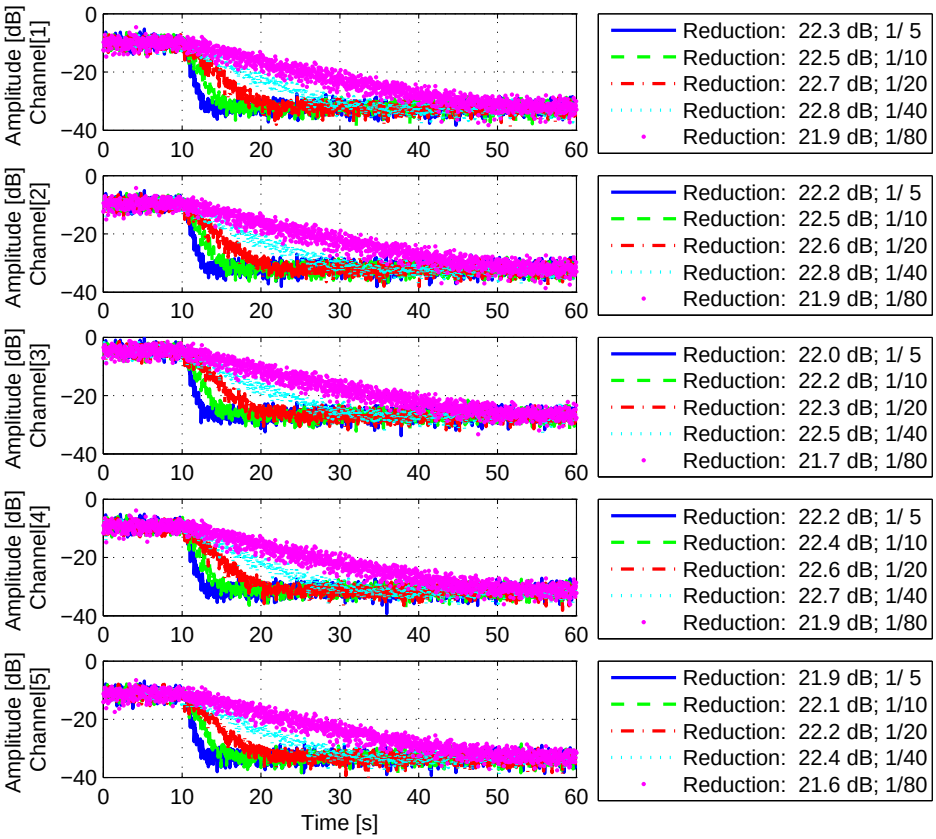


Figure 4.15: *Convergence curves for different α's using a feedforward controller. The format in the legend is reduction: step size α.*

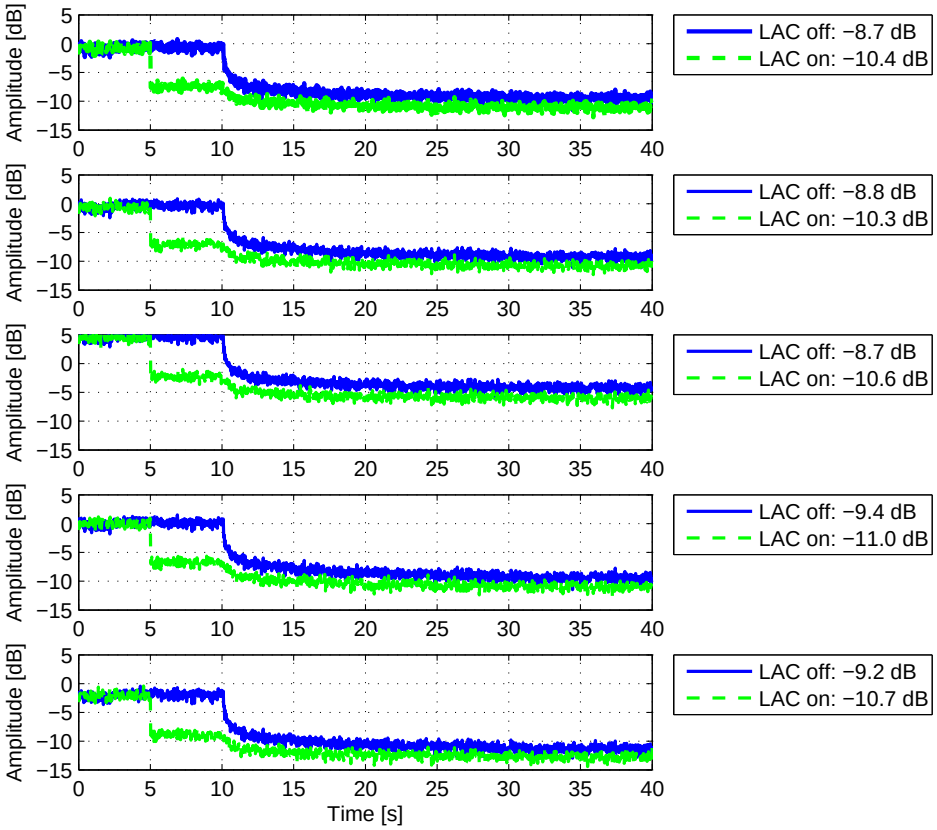


Figure 4.16: *Convergence curves for MIMO feedback with and without LAC.*

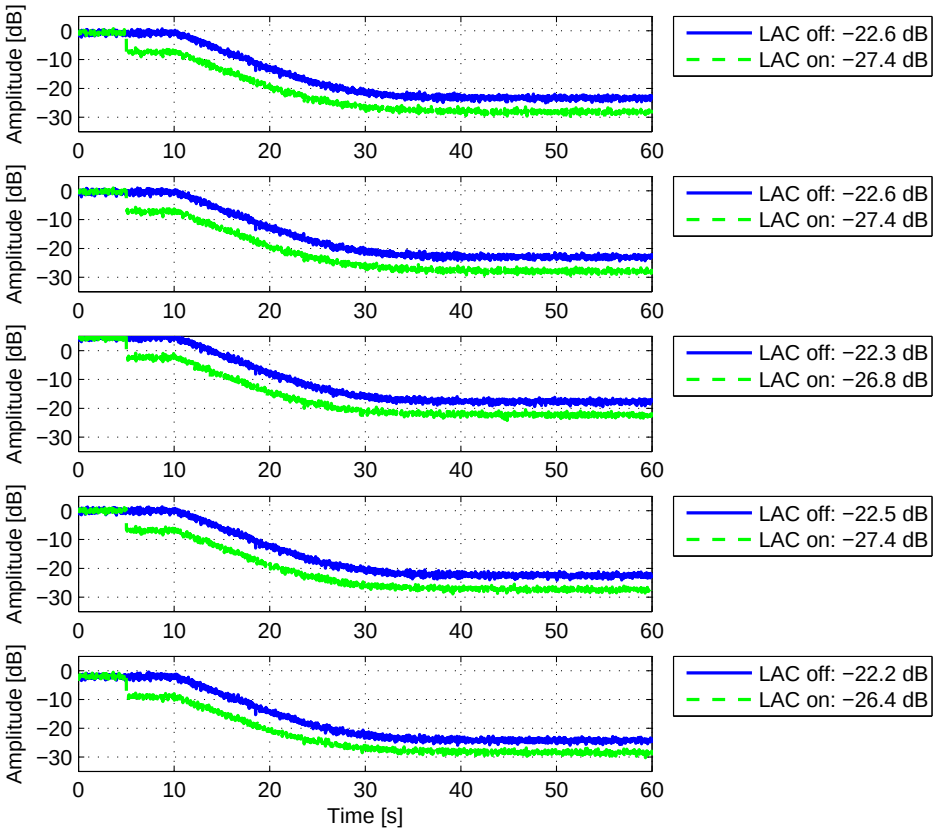


Figure 4.17: *Convergence curves for MIMO feedforward control with and without LAC.*

the time. The two plots in the Figure 4.16 demonstrate the difference between a high-authority controller with and without LAC. The plots show that LAC does not significantly influence the speed of convergence. However, the MSE improves with approximately 1.5 to 2 dB. This is probably due to the reduced phase of the plant under influence of the LAC-unit.

The same experiment was also carried out for the feedforward HAC algorithm. In this scenario, the reference signal was directly taken from the noise generator. All settings were similar to that of the previous measurement. The results of this measurement can be found in Fig. 4.17. In this scenario, the LAC unit was again switched off for the first 5 seconds and then switched on for the remainder of the duration. It can be concluded that the LAC unit does not influence the speed of convergence. However, it does improve the MSE with approximately 5 dB.

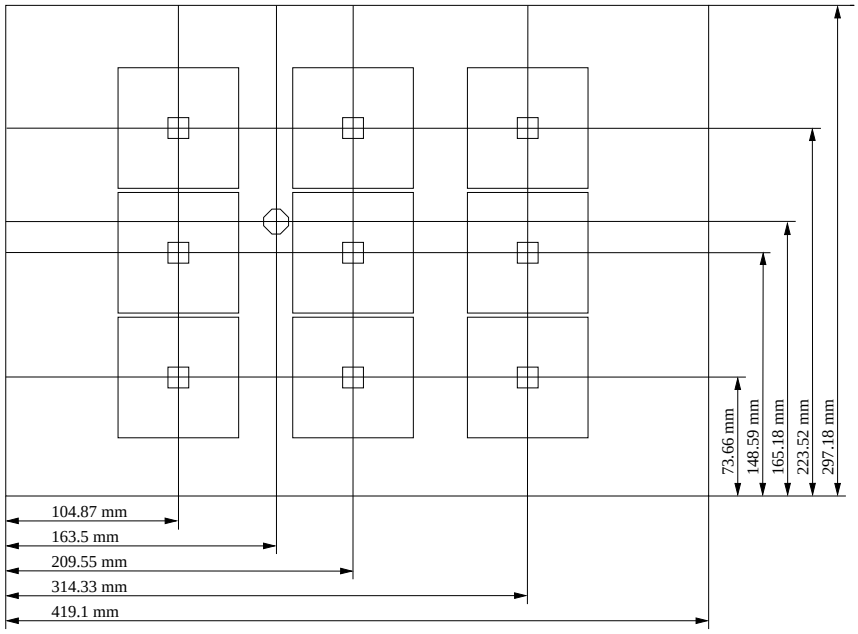


Figure 4.18: *Dimensions of the active panel used in the experiments. A mount with a thread can be used to add extra weight to the panel.*

4.4.5 Robustness under influence of LAC

In this section, the result of a study involving the robustness of a controller using a HAC/LAC architecture is presented. The LAC architecture adds active damping and the HAC architecture is constructed using the RMFeLMS algorithm. The robustness was assessed by modifying the dynamics of the panel. This was realized by adding weight to the panel. The MIMO controller is model-based and therefore the dynamics of the panel, modified by the additional weight, and the initially-estimated model that was calculated off-line, will now be different. The impact of this mismatch was studied for the case of feedforward and feedback control.

In the first experiment, the influence of the additional weight on the transfer function is presented. The weight could be mounted on the panel by means of a mount with a thread, where the mount was glued to the panel. The dimensions of the panel and the location of the mount can be found in Figure 4.18. The experiments were carried out with increasing weights (see the first column of Table 4.3). The model was estimated using a state-space identification method. The order of each model was 50 and the VAFs were all approximately 99.97%. The results of this can be found in Fig. 4.19. In this scenario, the LAC unit was switched off. The experiment was repeated but now with the LAC unit switched on. Using the same model order, the VAF's remained approximately equal. The

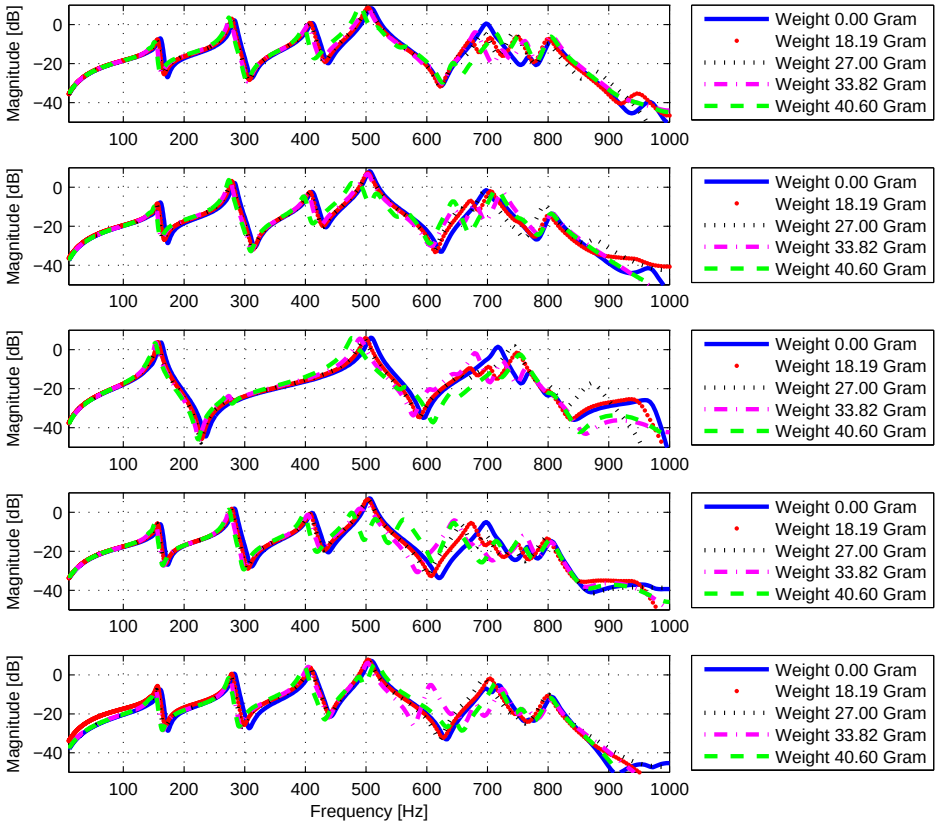


Figure 4.19: *Influence of additional mass on the transfer function of the panel without LAC.*

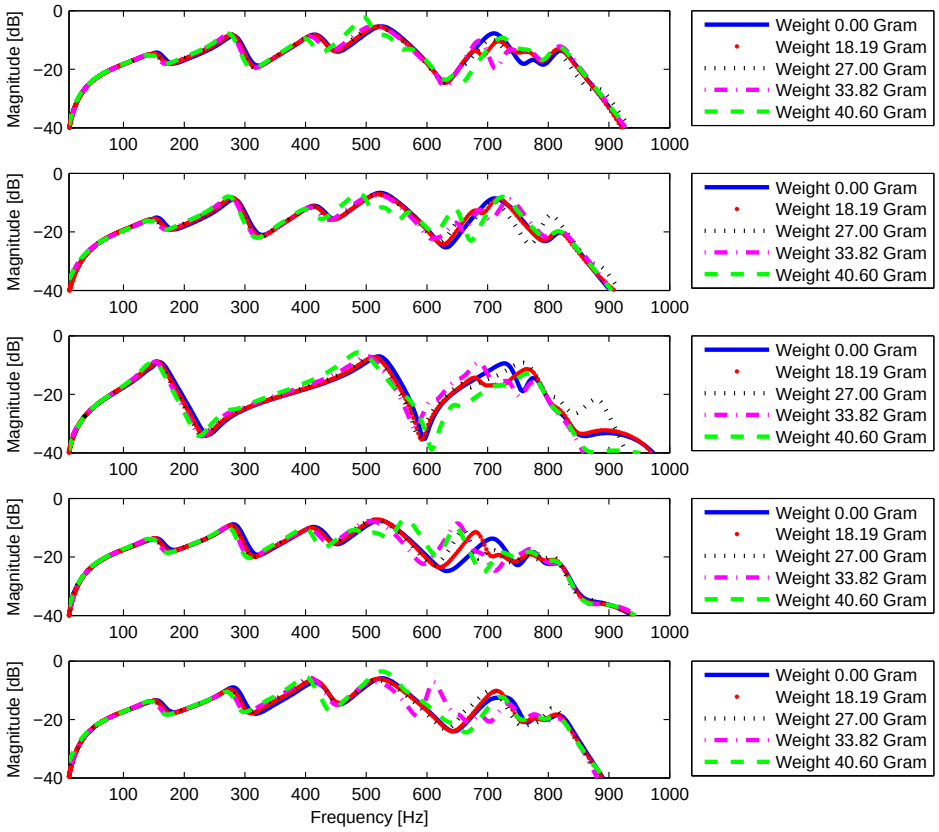


Figure 4.20: *Influence of additional mass on the transfer function of the panel with LAC.*

results of this can be found in Fig 4.20. It can be noticed that the resonance and the anti-resonance peaks become less extreme. This is a result of the active damping. A better method of visualizing the difference of the system with and without weight consists of plotting the phase difference between the two systems. In the first experiment, no LAC was used and the system transfer function was measured once with no added weight and once with 40.60 gram additional weight. In a second experiment, these measurements were repeated but now with LAC switched on. The results of this can be found in Fig. 4.21. From this, it can be concluded that LAC reduces the overall phase shift. This improves stability and robustness.

The robustness of the RMFe algorithm can be tuned by modifying the regularization-level β. However, this comes at the price that the reduction of the error signals is also decreased. This means that a conservative regularization

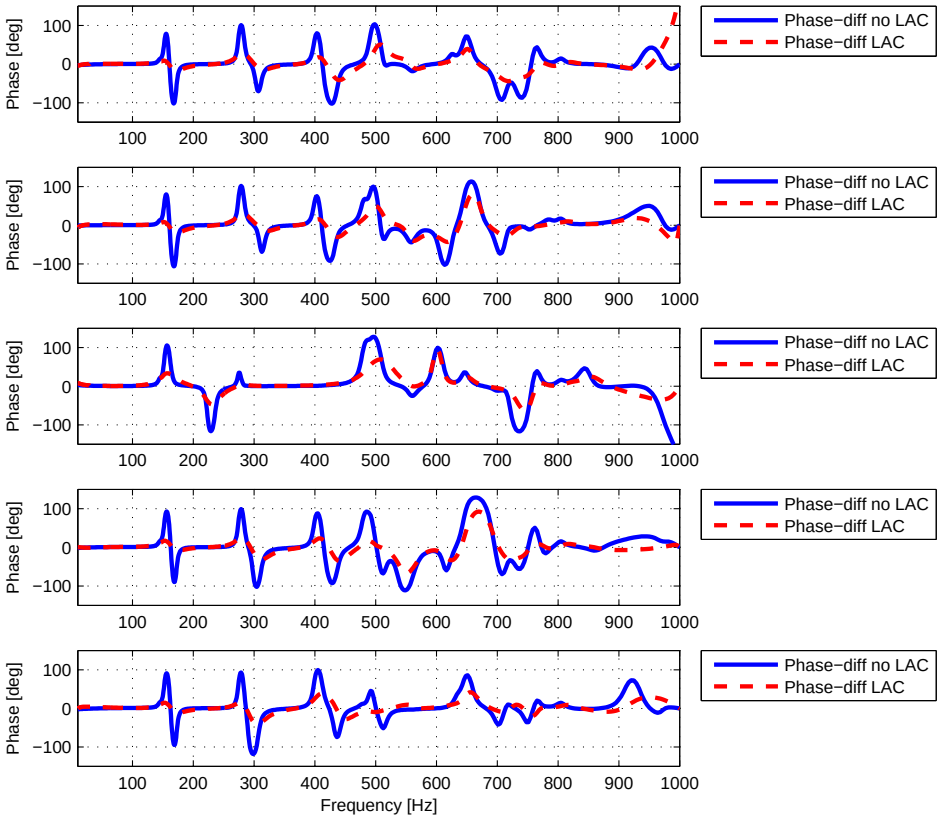


Figure 4.21: *A plot of the difference in phase between systems with and without additional weight. The first curve uses no LAC and the second one has LAC enabled. In this case a weight of 40.60 gram was added to the panel.*

Table 4.3: *Mean square error in dB; influence of added weight on the performance of a feedback controller. The reduction was measured after 180 seconds and was the average MSE over all 5 sensors. A step size $\alpha = \frac{1}{40}$ was used. The model for IMC and the RMFeLMS algorithm were identified without the additional weight.*

Weight	β = -20 dB		β = -25 dB		β = -30 dB	
[gram]	LAC	LAC off	LAC	LAC off	LAC	LAC off
00.00	11.42	10.42	11.98	10.50	12.11	11.77
18.17	11.45	10.39	12.36	10.94	13.01	11.63
27.00	11.55	10.32	12.54	10.95	12.38	11.29
33.82	11.56	10.35	12.66	11.22	12.82	11.59
40.60	12.19	-	11.83	-	12.81	-
47.32	11.50	-	11.76	-	-	-
54.17	12.18	-	12.43	-	-	-
61.05	11.51	-	12.34	-	-	-
67.85	12.03	-	12.71	-	-	-
74.96	11.87	-	12.02	-	-	-
81.67	11.83	-	12.21	-	-	-
88.41	11.67	-	12.62	-	-	-
95.22	11.89	-	-	-	-	-

level β results in a robust system but with a relatively poor reduction. In this paragraph, the influence of a relatively low regularization-level combined with LAC is evaluated.

In the first test, the RMFeLMS algorithm was used in combination with IMC to construct a feedback controller. The parameters were set to: $\alpha = \frac{1}{40}$, $\gamma = 10^{-4}$, $N_D = 80$, $W_i = 80$. The regularization-level β was set to -20 dB, -25 dB and -30 dB. The result from this measurement can be found in Table 4.3. The controller was switched on for 3 minutes and then the reduction was measured. A configuration with LAC on and LAC off was also tested. The reduction increased when the regularization level β was set to a lower value. From this table it is clear that a level of -25 dB yields the best reduction as well as good robustness properties. The use of LAC results in 1 dB additional reduction, as already seen before in the convergence curves.

In a subsequent test, the influence of LAC on the performance and robustness of a feedforward controller was evaluated. In this setup, the controller used the following parameter: $\alpha = \frac{1}{40}$, $N_D = 80$, $W_i = 350$ and regularization levels β of -25 dB , -30 dB and -35 dB. A regularization level β of -20 dB restricts the controller to much. The measurements were again repeated for several weights. The result of this experiment can be found in Table 4.4. The feedforward controller shows improvement of the robustness but not as noticeable as for the feedback controller. The regularization level β now limits the maximum reduction and a regularization level β of -25 dB or -30 dB seems to be a good compromise. A regularization level β

Table 4.4: *Mean square error in dB; influence of weight on the performance of a feedforward controller. The performance was measured after 180 seconds. A step size $\alpha = \frac{1}{40}$ was used. The model for the RMFeLMS algorithm was identified without additional weight.*

Weight	β = -25 dB		β = -30 dB		β = -35 dB	
[gram]	LAC	LAC off	LAC	LAC off	LAC	LAC off
00.00	25.42	21.35	27.53	22.17	27.62	22.57
18.17	25.27	21.59	26.94	21.51	28.68	22.83
27.00	24.70	-	26.24	-	27.33	-
33.82	25.21	-	22.61	-	-	-
40.60	-	-	-	-	-	-

of -25 dB gives the highest possible reduction but not the best possible robustness. It seems that a feedforward controller depends more critically on the model than the feedback controller, however the reduction of the feedforward controller is also higher than that of a feedback controller.

It can be concluded from the previous section that LAC has a positive effect on the robustness with respect to parametric uncertainties and the reduction of error signals. However, the benefit of LAC was stronger for the feedback controller than for the feedforward controller. This resulted in the following question: "What happens if the model used for the RMFeLMS algorithm is the one with LAC on but that the external LAC-unit is switched off". The result of this experiment can be found in Table 4.5. The reduction is approximate 1 dB lower than that of a real LAC-setup. The robustness is not as good as that of a real LAC-setup. However, this setup already gains some robustness and increase in reduction. It seems that the reduced phase of the model results in a system with better robustness properties. This is probably due to the reduced phase shift in the loop. In this case the reduction of phase shift improves the system more than the model mismatch degrades it.

4.5 Conclusions

In this chapter it was shown that recent advances in signal processing and integrated circuit design make it possible to implement a HAC/LAC architecture. A HAC/LAC architecture was realized as a high-speed decentralized controller running on a FPGA and a medium speed centralized controller running on a central processor unit. In this chapter it is shown that this is feasible and that for a suitable configuration, it improves the performance of adaptive algorithms.
The HAC/LAC control scheme improves the general performance. The major improvement is a further reduction of the MSE. However, it does not improve convergence speed. Another advantage of the HAC/LAC structure is that it also improves the robustness. These advantages come at the price of an increased com-

Table 4.5: *Mean square error in dB; influence of weight on the performance of a feedback controller. The performance was measured after 180 seconds. A step size $\alpha = \frac{1}{40}$ was used. In this case the LAC unit was switched off but the model used LAC and was used for IMC and the RMFeLMS controller. This model was identified without additional weight but with LAC switched on. The regularization level β was set to -25 dB.*

Weight [gram]	00.00	18.17	27.00	33.82	40.60	47.32	54.17	61.05	67.85
Reduction [dB]	11.25	11.73	11.94	11.63	11.74	11.35	11.36	11.45	-

plexity in the form of additional inputs and additional sensors. However, for the rapid prototyping system as described in this thesis, the hardware needed on the platform can be implemented on the hardware which is already needed to interface with the high-authority controller that is running on the embedded computer.
The increase in robustness is most noticeable in a feedback controller. This is probably due to the reduced phase shift. This reduction in phase shift is the main contributor to the increased robustness and improved MSE. This can be clearly seen in the test were only the model that uses a LAC controller was used. The real physical system had no LAC-control enabled. The robustness and MSE already improved with this scenario. This seems to be consistent with previous studies concluding that the phase shift is the important property governing the stability of a feedback controller (See [20]. In the case of the feedforward system, the robustness did not increase that much. The feedforward control scheme was found to be more sensitive to model errors. This can be explained by the notion that an accurate prediction of the disturbance signal is critical for the reduction of the MSE in feedforward controllers.

Chapter 5

Development system hardware

5.1 Introduction

In this chapter a controller and control architecture is introduced that can be used as a development system for active noise and vibration control[1]. In particular, the architecture enables the comparison of different strategies for the control of panels with piezoelectric actuators. One such control strategy is based on distributed control [65, 66]. The control architecture in this chapter is capable of evaluating the algorithms and corresponding hardware that are required for such a distributed architecture. However, the realization is concentrated in a single control module, which still allows evaluating the key features of a distributed system. A relatively low sample rate is used for centralized control whereas a high sample rate is used for localized control, such as for the HAC/LAC methodology as described in Chapter 4. The LAC algorithms, which are relatively simple but run at high sample rates can be realized efficiently on reconfigurable hardware, such as an FPGA. The HAC algorithms, which are relatively complex and which run at relatively low sample rates, are more efficiently realized on a general purpose Central Processing Unit (CPU). To be able to use the system for mobile demonstrators, the requirements are that the system should be compact, and that the power consumption should be low. Additionally, the system has to allow stand-alone operation. These requirements can be fulfilled with a CPU on a PCI-104-based embedded PC, in this case a Lippert Coolrunner 4 module.

Several functional blocks are realized on the FPGA, such as the functional unit that performs low-authority control, the communication with the PCI bus [67], but also the decimation and interpolation processing required for sample rate

[1]The material presented in this chapter is based on InMAR deliverable "*D275: Description of a hardware platform and the communication with an RT-Linux/Simulink interface*

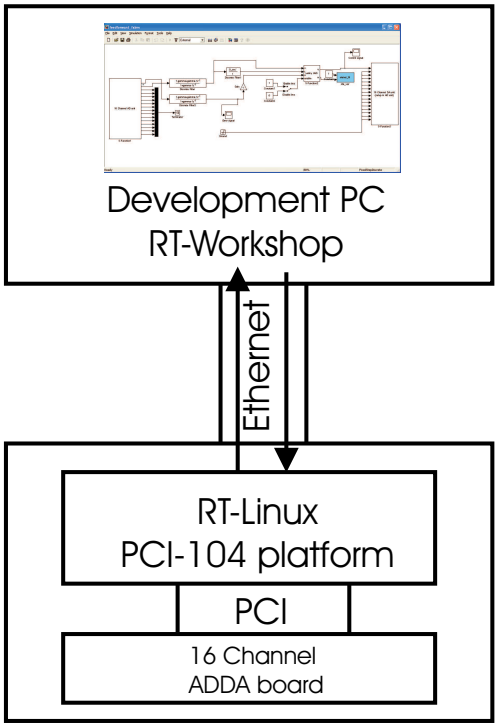


Figure 5.1: *System overview of interface between the development system and the PCI-104 platform.*

conversion for high-authority control by the CPU. The FPGA combined with AD and DA converters and analog filtering are combined on a separate PCB, the analog interfacing card.

An important objective in the design of the architecture was to minimize the overall delay in the controller. Critical components that determine the delay in the controller are the decimation and the interpolation filters [20], as well as the delay due to the order and timing of data acquisition, processing, and output signal generation [41, 68]. The flexibility of the interpolation and decimation filters can be used to realize a system that is an compromise between performance and delay in the system (See ref. [69]).

In this chapter, the description of this hardware platform that works with a Simulink/RT-Linux environment is given. An overview of the interface between Simulink and RT-Linux environment can be found in Figure 5.1 and a picture of the system realized and described in this chapter can be found in Figure 5.2. In a series of measurements the correct operation of the platform is shown. The delay is estimated and is given for a system with and without decimation and

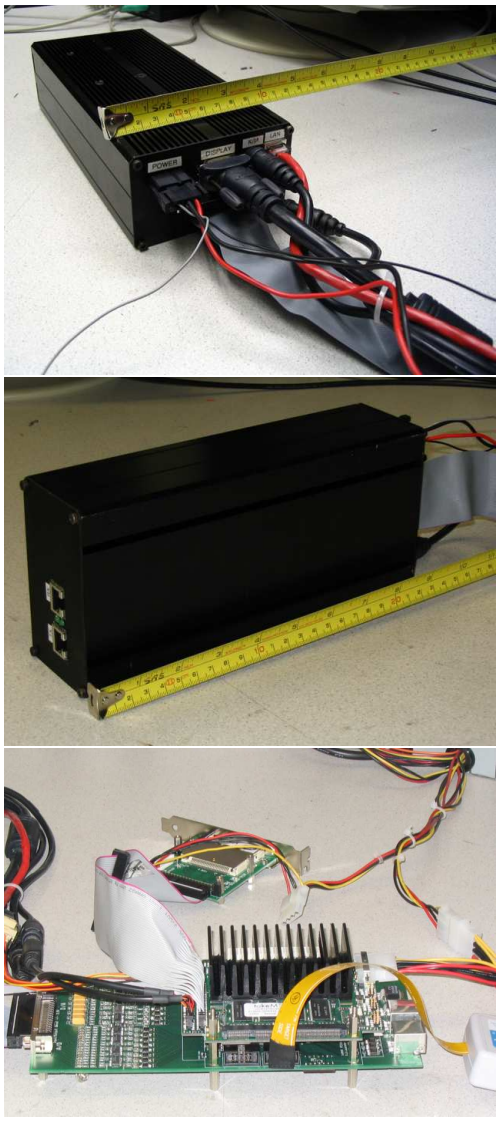


Figure 5.2: *Picture of the controller in its enclosure above and without enclosure below.*

interpolation filters and for a sub-sample delay sampling procedure. This chapter only contains the design of the high-authority control path.

5.2 Hardware architecture

The design of the hardware tries to combine the following constraints and objectives,

It needs to offer sufficient computational power;

The system must be flexible in terms of ease of development;

It needs to be small in terms of the physical size of the system.

The first item has been addressed by using a RT-Linux platform that has sufficient computational performance. The second item can be covered by using a Simulink/RT-Workshop interface. The third and final one is more of a challenge. In most ADDA systems the anti-aliasing and reconstruction filters are left out. This results in a system that still requires a lot of external analog components. The filters should reduce anti-aliasing errors and reconstruction errors as much as possible, but for many applications, the group delay of the filters should be as low as possible. In order to obtain the optimal trade-off between stopband attenuation and group delay, the filter characteristics should be fully programmable. In this design a combination of a fully programmable digital filter and a fixed analog filter is used. The digital filters have a relatively low cut-off frequency while the analog filters have a relatively high cut-off frequency. The cut-off frequency of the analog filters was fixed to 20 kHz. The AD and DA units are now operating at a higher sample rate and filtering at low frequencies is performed in the digital domain. Lower sample rates are obtained from higher sample rates using integer frequency steps.

The architecture of the system can be found in Figure 5.3. The overall system consists of a PCI interface, a timer unit, an interrupt unit, an AD de-serializer, a decimator, IO registers, an interpolation unit and a DA serializer. The interrupt controller is used to register the interrupt source and to determine which unit has generated the interrupt. The timer unit contains two timers. The timers can be used separately or in the master-slave mode. The interpolation and decimator unit are realized as two concatenated stages. The interpolator or decimator unit can be bypassed if required.

The decimation unit is programmable and consists of two concatenated units. It basically consists of a low-pass filter followed by a down-sample unit. Each low-pass filter can be programmed with a maximum of 64 coefficients. The filter structure is based on an FIR structure. The decimation ratio is programmable and can have any integer fraction. The total fraction is the product of both stages. The delay line of each section is 64 taps long. The down-sample unit will remove

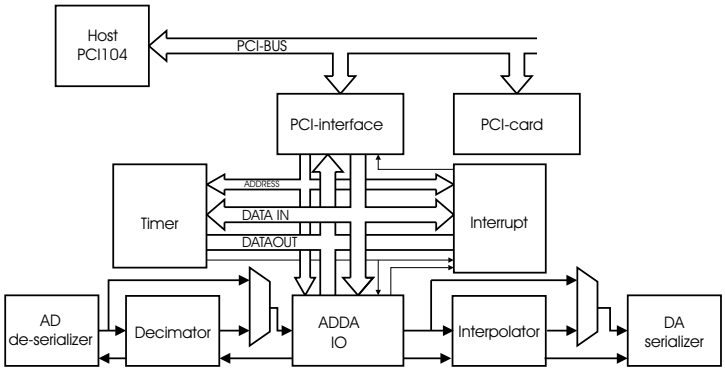


Figure 5.3: *Architectural overview of the FPGA.*

$n-1$ samples on every n samples. Each section can be programmed with different filter coefficients, filter length and the appropriate decimation ratio.

The interpolation section is somewhat different and can be programmed with a maximum filter length of 256 coefficients. The up-sample unit and the low-pass filter are combined in a so called polyphase implementation. This structure is again based on an FIR filter structure. The delay line in this FIR filter has a length of 64 taps. However, in this delay line only the input samples are stored. In a normal interpolator zeros are inserted and then the output is filtered. If the up-sampler and the filter are combined it is only necessary to store the input elements and not the inserted zeros. This results in a much longer delay line, for instance if the interpolation factor is 5 then the real length is 64*5=320 taps. Due to this advantage the coefficient memory was expanded to 256 coefficients. The delay line is again 64 taps. However, the coefficient memory is 256 entries long.

The sample rate of both sections is directly related to the interrupt rate. Each section multiplies the interrupt rate by the decimation rate and interpolation rate, respectively. This construction has some setbacks in controlling and adjusting the sample rate but makes it possible to switch the interpolation and decimation on and off if needed. It also has the advantage that the interrupt-based mode can be switched off, so that an external interrupt controller can be used. This structure makes it possible to reduce the group delay by switching-off the anti-aliasing or reconstruction filters.

5.3 Analog electronics specifications

The specifications of the analog part of the ADDA board can be found in Tables 5.1 and 5.2. The frequency bandwidth is measured from the low frequency -3 dB point to the high frequency -3 dB point. The output specifications depend on the

Table 5.1: *The specifications of the analog input of the 16 channel ADDA board.*

INPUT		
low-pass filter	-3 dB cut-off freq.	23 kHz
	slope	40 dB/dec
high-pass filter	-3 dB cut-off freq.	1 Hz
	slope	20 dB/dec
max voltage(peak-to-peak)		4.8 V
impedance		30 kΩ

Table 5.2: *The specifications of the analog output of the 16 channel ADDA board.*

OUTPUT			
		50 Ω load	1 MΩ load
low-pass filter	- 3 dB cut-off freq.	23 kHz	23 kHz
	slope	40 dB/dec	40 dB/dec
high-pass filter	- 3 dB cut-off freq.	1 Hz	1 mHz
	slope	20 dB/dec	20 dB/dec
Max voltage (peak-to-peak)		2.4 V	4.8 V
impedance		50Ω	

connected load. In this table, two loads were characterized, one at 50 Ω and the other at 1 MΩ.

5.4 Verification

The platform described in the previous sections was validated and verified This was done by series of simple experiments. First, the basic interface between Simulink and the platform was verified. A more concise explanation can be found in Chapter 6. In this section, we are going to have a look at the platform itself. The platform was first tested by writing a driver that tested the RT-Linux environment. The purpose of this driver was simple and consisted of a simple feedthrough operation. It was confirmed in a series of simple experiments involving some basic measurement equipment that the overall system was working fine. The basic test involved testing the timer, interrupt generation, the DA unit without any interpolation filter, the AD unit without an anti-aliasing filter, the path from the AD to DA and finally the interpolation and decimation filters. From this series of experiments it was concluded that the overall system worked as expected. The code used to debug the hardware was then used to develop and implement a driver that provided a simplified interface to the hardware. How the interface with this driver works is the subject of Chapter 6. The driver that was implemented is now part of the RT-Linux environment and is loaded during the booting of the platform. The

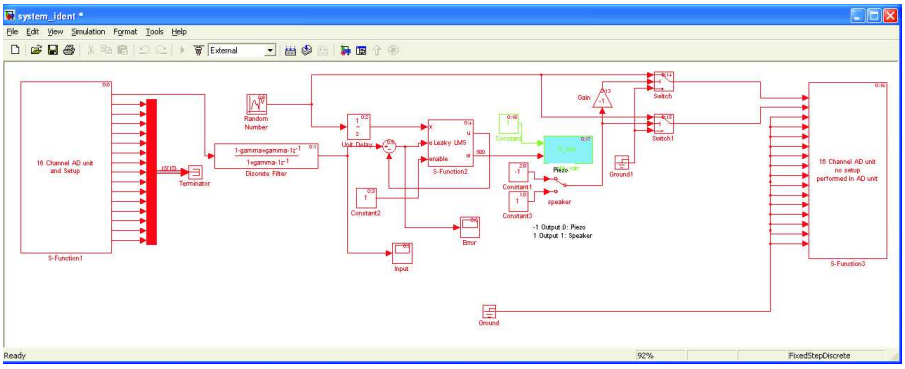


Figure 5.4: *Simulink model used for system identification.*

driver can control up to 4 cards and uses a simple function call-based interface. A callback interrupt routine can be installed to handle interrupts from the card. The architecture is built up in such a way that an interrupt is generated when an analog to digital conversion has been performed. The RT-Linux platform, together with the Simulink interface, provides a basic interface to handle this interrupt concept. The idea is that the interrupt generation can be used to generate the timer ticks for the realtime thread that handles the control algorithm.

5.5 System identification

The first test consisted of a simple system identification procedure. The output of the system was directly connected to the input of the system. No filtering at the input or output of the system was performed. The idea is now that a delta pulse will be measured with some delay. The Simulink model that was used for this purpose can be found in Figure 5.4. The procedure is that noise is supplied to the system that needs to be identified. A simple LMS algorithm is used that estimates the plant. The plant is then modeled by means of a moving average (MA) identification process. The platform supports three operational modes; 1. The mode which will be called one-sample delay mode. 2 The direct feedthrough mode. 3. The sub-sample delay mode. The first mode is a classical concept in which a sample is acquired, subsequently the algorithm is started and when it has finished, the output is written into the DA registers. This value is translated into an analog value when the next analog to digital conversion is started. This classical approach generates a one-sample delay. The second mode writes the value that has been calculated in the algorithm into the DA register but will also directly start the DA conversion. The advantage of this mode is that system delay is shorter [41, 68]. However, it is likely that some phase jitter will be introduced. The third mode is a mix between the direct feedthrough mode and the one sample

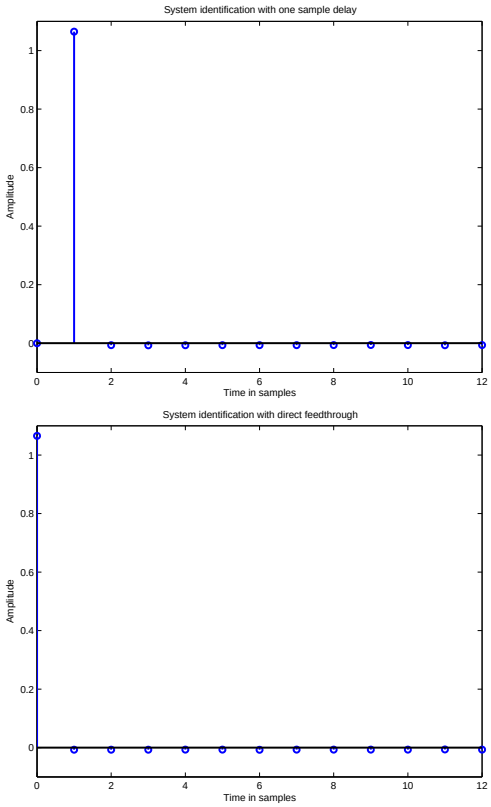


Figure 5.5: *Impulse response resulting from system identification with the scheme of Figure 5.4; One-sample delay mode (left) and direct feedthrough mode (right).*

delay mode. In this mode a master timer is programmed with the sample rate. A second timer, the so called slave timer, is programmed with an offset. The idea is that the one sample delay from the one-sample delay mode can now be replaced with a fractional sample. In this test only the first two modes will be verified. Tests of the third mode are described in Section 5.6. The assumption is that the one-sample delay mode introduces a one-sample delay in the identification and that the direct feedthrough mode introduces a negligible delay. Testing the sub-sample delay mode is not very useful and is identical to the direct feedthrough if the slave time is adjusted correctly. From the result in Figure 5.5 it can be concluded that the overall system works as expected.

5.6 The delay of the system

For this test a network analyzer was used to measure the delay with a higher accuracy. But first lets have a look at the expected delay in the system. The system uses simple anti-aliasing and reconstruction filters with a cut-off frequency of 20kHz. From simulation it was derived that the delay of the anti-aliasing and reconstruction filter is approximately 10 μs resulting in an total delay of 20 μs. In the one-sample delay mode the delay introduced due to the sampling process is one sample and the delay introduced due to the hold of the DA conversion is 0.5 sample. In the tests, the sample rate of the algorithm was set to 2 kHz. This results in a total of 500 μs + 250 μs + 20 μs = 770 μs delay. For the direct feedthrough mode this is 250 μs + 20 μs = 270 μs. In practice this last figure is somewhat more and it depends on the amount of time spent within the algorithm running on the platform. Variation of computational time introduces jitter. In the FPGA, a system was implemented that is able to reduce this jitter by means of a master-slave timer principle. This mode will be called sub-sample delay mode. In this system the master timer triggers the AD conversion and generates an interrupt. At the same time as the AD conversion is started, a second timer, the so called slave timer, is started. This timer triggers the DA conversion. The advantage of this system is that the slave timer is programmable and that it can be programmed with a time that is a fraction of a sample period. This reduces the overall group delay. The output sample rate has an equidistant sample period resulting in little to no phase jitter. The current driver adds support for this system. It is necessary to first measure the time needed for the realtime thread. The slave time consists of this time and the time needed for the communication and overhead of the realtime kernel. The estimated value was $\frac{1}{25000}s = 40\mu s$. This results in three different algorithms: one-sample delay mode, direct feedthrough mode and sub-sample mode. In the first test, of all three algorithms, no interpolation or decimation was used. In the second test the interpolator was switched on and the delay of the system was measured again. The delay was again measured for all three algorithms. The filter that was used in this series of measurements is the equivalent of a zero order hold function. The filter consists of an FIR filter that contains N coefficients that are all 1. Such an FIR filter has a delay of $\frac{N-1}{2}$ samples, which can be easily verified with the `grpdelay` command of Matlab. The delay of the total system is now exactly the same as the delay of a 2 kHz system. The interpolator is actually implemented as a two-stage realization. This means that the interpolation ratio is now $N = N_0 N_1$. The real delay is the output sample time, in this system $\frac{1}{100e3\ Hz} = 10\mu s$, times the interpolation rate. Assuming $N_0 = 5$ and $N_1 = 10$, the delay is $\frac{5\times 10}{100e3\ Hz} = 500\mu s$ which is again equal to the delay of the 2 kHz system. This delay also includes the delay of the hold unit running at 100 kHz. The proof of this is not very hard but has been left out for sake of simplicity. The Simulink model used for the measurements can be found in Figure 5.6. In this experiment, the measurements of the overall system without any decimation or interpolation is shown. The network analyzer

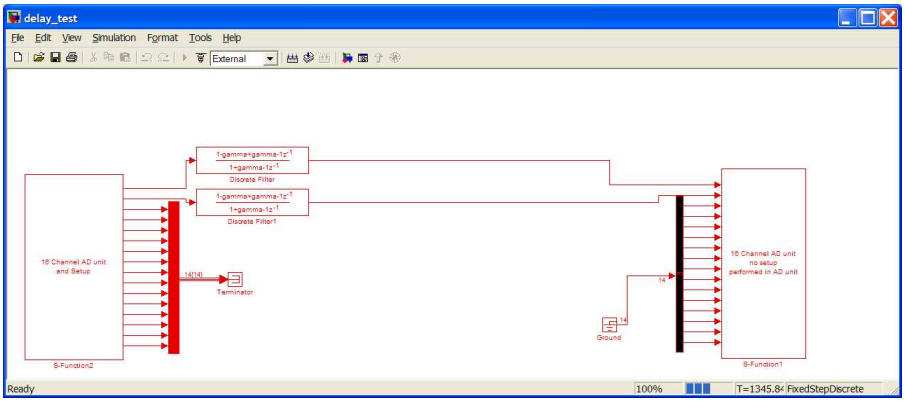


Figure 5.6: *Simulink model used for the measurement of the group delay.*

as used for the measurements, uses a sine wave to measure the transfer function and the group delay. The output signal is band filtered and only the frequency of interest is analyzed. This means that it is no problem to operate the system without anti-aliasing or reconstruction filters. The measurements of a system running at 2 kHz without anti-aliasing and reconstruction filters can be found in Figure 5.7. In this Figure, the magnitude and the group delay are plotted. For the one-sample delay mode and the sub-sample delay mode, the irregular curve for the group delay is caused by the imperfect reconstruction filter. The direct feedthrough mode suffer from additional group delay variations caused by jitter of the start of the AD-conversion. The measurements of a system in which the reconstruction filter emulates a zero order hold (ZOH) can be found in Figure 5.8. In this system no decimation filter is used. The system works internally with 2 kHz and has a 2 kHz input but the interpolator and therefore the output generates a 100 kHz signal. From the measurements it is clear that both systems have the same performance. It seems that the 2 kHz system with no interpolation and decimation is equivalent to the 100 kHz system in which the interpolator emulates a ZOH and no decimation filter is present. From Figure 5.7 and Figure 5.8 it can be seen that the trend for the group delay is a constant value. The direct feedthrough mode has a low group delay. However, it comes at a high price. The group delay is a very irregular function of the frequency, which is caused by the jitter of the start of the DA-conversion. This could explain why such systems, which have been known for a relatively long time (See, e.g. Ref. [68]) are still not widely used [41]. The sub-sample delay mode seems to be a better approach. In this mode the advantages of the one-sample delay mode, a group delay with a small variance, is combined with the advantages of the direct feedthrough mode, a low group delay. In practice the delay is difficult to estimate and must be obtained by measuring the maximum value for the execution time of the algorithm. In the

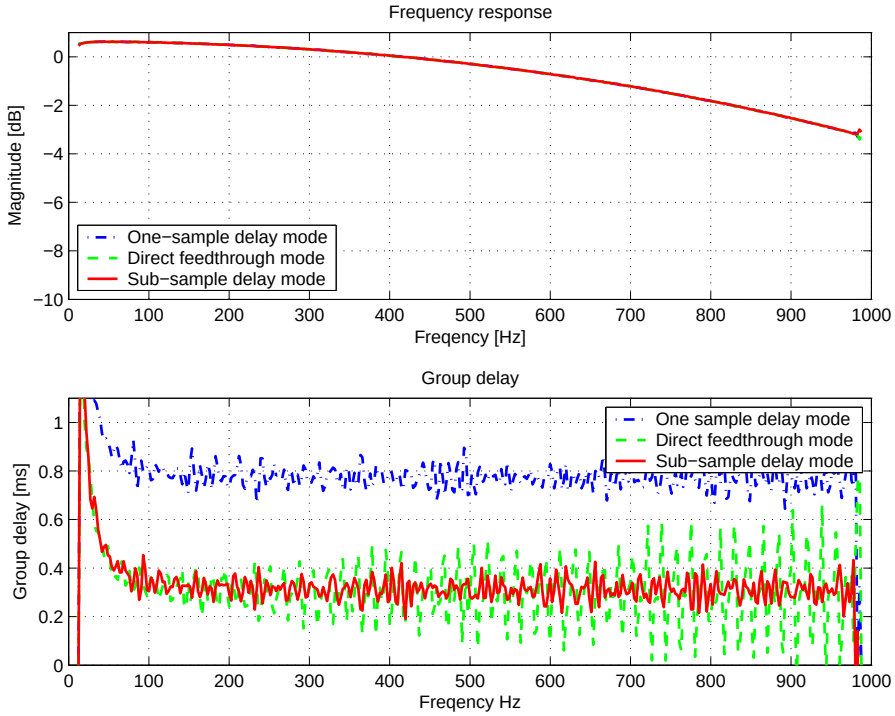


Figure 5.7: *Delay and magnitude of the system running at a 2 kHz sample rate.*

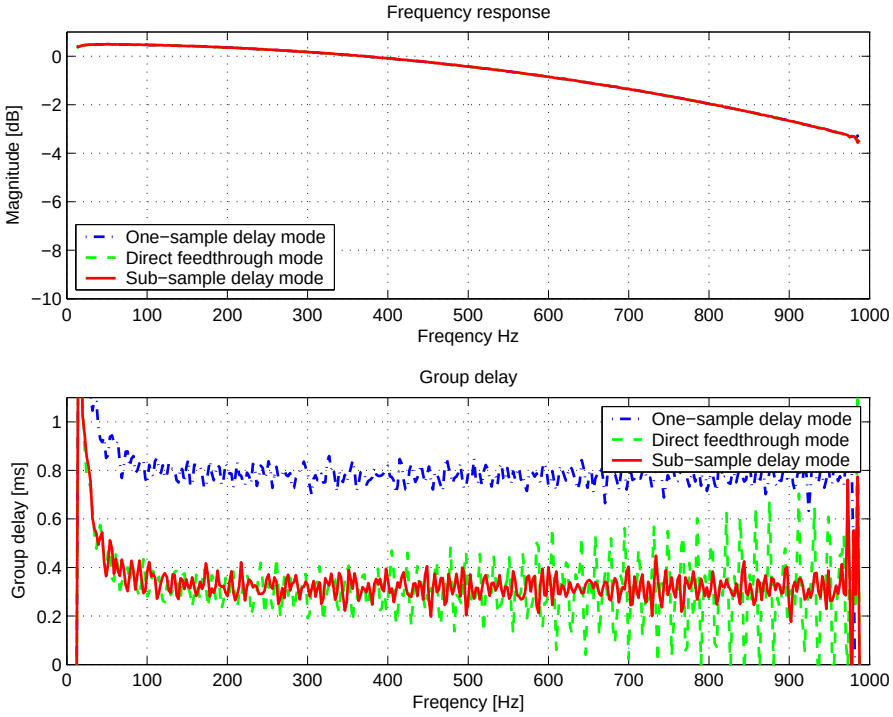


Figure 5.8: *Delay and magnitude of the system with the zero order hold interpolation filter enabled running at 100kHz and no decimation filter (2kHz internal).*

case of a simple FxLMS feedforward controller it is necessary to filter the reference signal with the plant model. The transfer function of this plant will be influenced by the delay present in the control system. This means that it is not possible to dynamically estimate the slave time and adjust it. This makes it necessary to estimate it together with the transfer function of the plant.

5.7 Testing the decimator and interpolator

The final test consisted of a system that used a minimum-phase filter with 30 dB stopband attenuation. The magnitude and group delay were again measured with the network analyzer. The measured filter characteristics were very close to the simulated characteristics from Matlab. The results of this measurement can be found in Fig. 5.9. The plot contains the magnitude and the group delay. For these results, the interpolator and the decimator were enabled. Three scenarios are plotted: the one-sample delay mode; the direct feedthrough mode and the sub-sample delay mode. The results were obtained using a direct pass-through operation that included a high-pass filter (see Figure 5.6). The purpose of the high-pass filter is to remove any DC component present on the input signal from the AD converters.

The group delay for the one-sample delay is a regular and continuous function of frequency. However, the group delay for the direct feedthrough mode is a more irregular function of frequency. It seems as if the jitter influences the group delay and makes the function more irregular. In the case of the sub-sample delay mode the regular continuous function properties of the one-sample delay mode are combined with the low group delay of the direct feedthrough mode. The delay chosen for the master slave mode is now $\frac{1}{16000}\ s = 62.5\ \mu s$. The slave time was increased when compared to that of Section 5.6. The total slave interval time is the sum of the following times: time needed by the decimation unit, communication to the PCI-104 platform i.e. reading from registers and interrupt handler response time, processing time of the control algorithm, time needed to write the value in the DA registers.

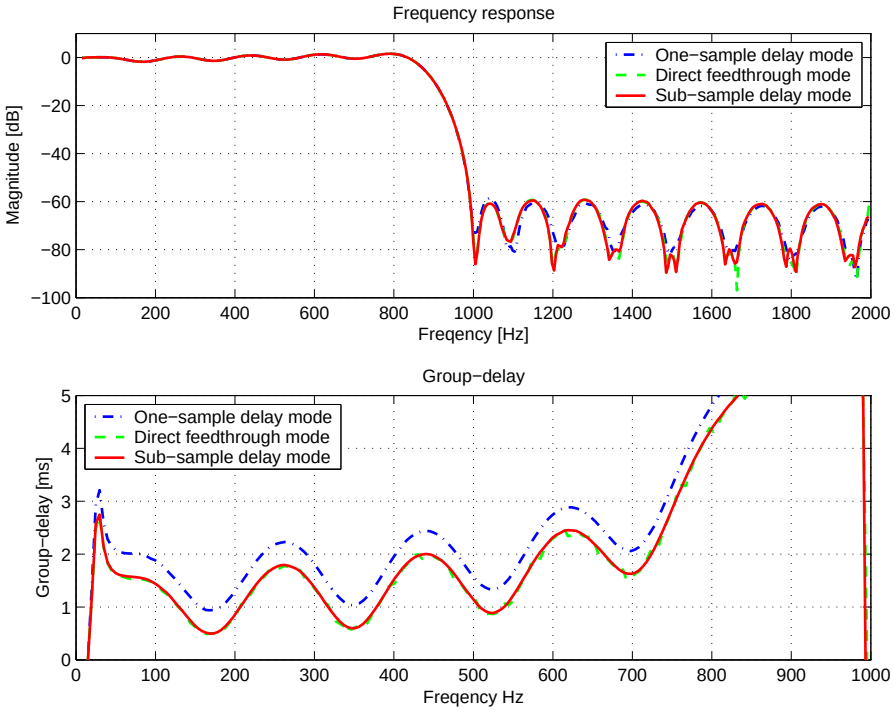


Figure 5.9: *Amplitude and delay of the system with resp. one-sample delay, direct feedthrough mode and master slave mode. For all three scenarios the decimator and interpolator where enabled.*

Chapter 6

Rapid prototyping software

6.1 Introduction

The rapid prototyping platform is based on a RealTime Linux distribution (RT-Linux) that interfaces with a Matlab Simulink/Realtime Workshop environment (RT-Workshop) [1]. This framework was developed by TNO (See Ref. [70]). With this framework, code can be automatically generated from a Simulink sheet using the Matlab RT-workshop. Only a limited subset of blocks is supported; in order to add more functionality so-called S-Functions can be added. The platform in its current configuration supports S-Functions written in C. The exact details of the hardware platform can be found in Chapter 5. In this chapter only software-specific subjects are covered. An overview of the system can be found in Figure 5.1 and a picture of the system in Figure 5.2. In this overview it can be seen that the system consists of several components. The development system is based on a Microsoft Windows machine hosting the Simulink environment. The code is generated using the RT-workshop. It is then sent to the RT-Linux PCI-104 based platform and compiled with the GCC compiler on this platform. The binary is downloaded and stored on the development system. When the user decides to run such a module it is uploaded to the RT-Linux environment and executed. From the Simulink environment it is now possible to start the module that was uploaded to the RT-Linux target system. Normal tools, like for instance the scope from Simulink are now available. For the AD- and DA-components special S-Functions are required. These are written with RT-Linux in mind. If these S-Functions are compiled on a Microsoft Windows machine they will contain empty placeholders. For future extension and ease of development a special interface is designed between the ADDA-unit and the S-Function. This makes it possible to exchange the hardware and driver without the need to replace the Simulink S-Function.

[1] The material presented in this chapter is based on InMAR deliverable "*D276: Description of an RT-Linux/Simulink interface and communication with a hardware platform*"

6.2 Software interface on the PCI-104 platform.

The PCI-104 platform runs a RT-Linux kernel. The realtime thread always has the highest priority and runs in kernel space. To separate the hardware and the software, a device driver is used. Such a device driver abstracts the hardware details and provides a generic interface. An overview of this system can be found in Figure 6.1. The driver can also be used to build a separate stand-alone kernel

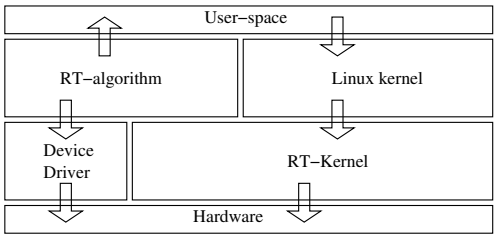


Figure 6.1: *Architectural overview of the realtime platform.*

module for the realtime system. In our application however, this module is built using the Matlab Realtime Workshop and Simulink.

6.3 Integration within Simulink

The driver is not part of the Simulink environment. An advantage of this approach is that it is now possible to design a high level interface, making it possible to exchange the hardware without the need to replace the high level Simulink interface. The hardware details are abstracted for the Simulink component. The device driver is therefore permanently installed on the system. The main task of the driver is to initialize ans setup the hardware. The function calls that are provided are the subject of a separate section. The driver provides functions calls for starting, stopping and setting up of the hardware. The hardware generates an interrupt and if an interrupt is generated the driver calls an user-defined function. This interface can be used to inform the Simulink component that a AD-conversion is ready. A simple function call is provided to install this callback function.

In Simulink, it is necessary to build a sink and source component to interface with the hardware system. The source component represents the AD-unit and the sink component represent the DA-unit. In the design, it was decided to let the AD-component perform the setup of the hardware. If the components are compiled for the realtime target, they contain all code needed to interface with the device driver. However, when they are compiled on a Windows host they act as empty place holders. This makes it possible to partially test the Simulink model on the host platform before moving it to the realtime system.

Table 6.1: *Implemented device driver functions.*

Function name
PCI_ADDA16_Start(int card, enum PCI_ADDA16_SAMPLE_MODE mode);
PCI_ADDA16_StartDA(int card);
PCI_ADDA16_Stop(int card);
PCI_ADDA16_GetADDA(int card,PCI_ADDA_P *p);
PCI_ADDA16_InstallIrqHandler(int card, void (*f) (int ,PCI_ADDA_P));
PCI_ADDA16_RemoveIrqHandler(int card);
PCI_ADDA16_SetDecimator (int card, int enable, PCI_ADDA_DEC_P);
PCI_ADDA16_SetInterpolator (int card, int enable, PCI_ADDA_INT_P);
PCI_ADDA16_SetSlaveTimerOffset(int card, int ticks);

6.4 The device driver

The device driver is a hardware abstraction layer. The functions of the driver are to:

Enumerate all cards. Up to 4 cards are supported;

Initialize each card;

Provide the possibility to upload different decimation and interpolation filters;

Switch the decimation filter on or off;

Switch the interpolation filter on or off;

Supply support for call-back function making it possible to support interrupts;

Support the different sample modes;

Start and stop the algorithm.

Each function is implemented as a function call. The cards are automatically detected and initialized. It is still possible to upload other filter coefficients. The coefficients become active on the moment that the sample process is stared. The driver is interrupt driven; an interrupt is generated after an analog to digital conversion. Three different sampling scenarios are available: one-sample delay mode; direct feedthrough mode and sub-sample delay mode. The exact definition can be found in Chapter 5 and will not be repeated here. The one-sample delay and the sub-sample delay mode do not need any further attention. The direct feedthrough mode makes it necessary to start the digital to analog conversion by means of a function call. The driver uses function calls to access the hardware. All functions can be found in Table 6.1. The different function are described in

next paragraphs.

The first group consists of function that modify the parameters of the card. These function should be used when the card is not actively running.

PCI_ADDA16_InstallIrqHandler(...);
The first parameter is the card number and the second one is a function pointer to a call-back function. The call-back function should also have a pointer as an argument. The call-back function will be called after a successful analog to digital conversion. The argument of the call-back function is an address to the IO register of AD- and DA-registers. The first 16 addresses are the AD-channels and the next 16 addresses are the DA-channels. This function should be called when the system is not running.

PCI_ADDA16_RemoveIrqHandler(...);
Remove an installed call-back function. Only call when the system is not running.

PCI_ADDA16_SetInterpolator (...);
This function should be used to upload new interpolator filter coefficients. The interpolator consists of two sections, h0 and h1. The order of these sections is: DA-registers → h1 → h0 → outside world. Each section requires its own filter coefficients. The function takes three arguments. The first one is the card number. The second argument enables or disables the interpolator. The third argument is a pointer to a structure that contains the filter coefficients for each section h0 and h1, the length, the interpolation factor and, a normalization flag. If the normalization flag is false the filter coefficient need to be normalized. Normalization is required due to the fact that the interpolation process reduces the output energy. For example an interpolator with an interpolation factor of 5 reduces the output amplitude with a factor of 5. The coefficients will therefore be multiplied by the interpolation factor if the normalization flag is false.

PCI_ADDA16_SetDecimator (...);
This function should be called to upload new decimator filter coefficients. As with the interpolator, the decimator consist of two sections, h0 and h1. The order of these sections is: outside world → h0 → h1 → DA-register. The function takes three arguments. The first argument is the card number. The second argument enables or disables the decimator. The third argument is a pointer to the structure that contains the coefficients for h0 and h1, the length and the decimation factor.

PCI_ADDA16_SetSlaveTimerOffset(...);
In sub-sample delay mode, this function must be called with the number of clock ticks needed. Each tick is 30 ns. The delay in ticks can be calculated by measuring or estimating the worst case execution time and dividing it by 30 ns and subtracting one from this number. It is also necessary to include the time needed for the system to respond to an interrupt.

Functions that start and stop the ADDA-conversion process.

PCI_ADDA16_Start(...);
This function starts the AD, and, depending on the mode, the DA-conversion. The sample modes supported are DIRECT_FEEDTHROUGH, ONE_SAMPLE_DELAY or MASTER_SLAVE_MODE. In the one-sample delay

mode and the sub-sample delay mode, no further action needs to be taken. It is sufficient to read the samples to perform processing and write back the results in the DA register. However, in direct feedthrough mode it is necessary to start the DA conversion manually. The **PCI_ADDA16_StartDA(int card);** function can be used to perform this task.

PCI_ADDA16_Stop(...);

Stop a running system.

Function(s) that get additional information.

PCI_ADDA16_GetADDA(...);

This function returns a pointer to the AD- and DA-register. This is the same pointer as provided in the call-back routine. In Simulink, the call-back routine is only used to trigger the Simulink model. The triggered system again calls a function. This function makes it possible to get a pointer to the AD and DA IO-space from within in this routine.

If a function call fails, it will return an error code in the form of an integer enumerate. The error codes are not described in full detail, but they can be taken from the header file 'adda16.h'.

6.5 Simulink interface

In this section, two topics are covered. The first topic is about developing a driver component for Simulink and the second one is about the function provided to realize an interface to the realtime target.

In the previous section, the driver functions were described. It is clear that the hardware must be initialized and configured before the card is started. The initialization of the hardware is abstracted by the driver. However, some measures must be taken to prevent problems. The platform is based on the RealTime Linux operating system. The realtime platform uses a simple time-discrete fixed-step solver that is running in single-tasking mode. A special clock thread is running on the RT-Linux platform that synchronizes the controller thread. The thread that is running the control algorithm is usually in sleep mode and will be periodically awakened by the clock thread. This method seems to be quite complex but it has the advantage that the clock thread can be replaced by another periodic thread, making it possible to use another clock to synchronize the controller thread.

In Simulink a model is always built as a flow from a source to a sink. The source in this case is the AD-converter and the sink is the DA converter. The AD-unit therefore must dictate the pace. The Realtime platform as implemented by TNO provides a function to dictate the pace. This function must be periodically called to run the model. A pointer to this function can be fetched using the **mdlInitExternalTrigger(&pFnTrigger)** function. This pointer must be stored into a pointer variable and called periodically. An interrupt handler is installed and from within this interrupt handler the trigger function is called. To stop this triggering mode it is necessary to call the **mdlTermExternalTrigger()** function.

The interrupt handler will then wake the controller thread and will execute the model. Normally, the call-back function for the interrupt service routine has the address of the ADDA registers as a parameter. However, in this case the interrupt routine is only used to awaken the thread. This makes it necessary to get this address separately using the **PCI_ADDA16_GetADDA** function. A second slave-like DA-component is used to write the output value into the DA-register. Only the AD-component is allowed to setup the card, the DA-component may only write to the DA-unit and start a conversion when the direct feedthrough mode is selected.

6.6 Support software

The realtime environment contains some commands to start, stop and upload modules. A special interface is provided that makes it possible to remotely set and get data from the platform. This section gives a brief introduction into the most commonly used commands on the realtime target platform. The commands that interface with the platform start with **srt_**. An overview of these commands can be found in Table 6.2. The target system is running a Samba server. The

Table 6.2: *Simulink Realtime target (SRT) matlab commands.*

Function name	Description
srt_mdl_boot	Specify the model that must be started at boot time
srt_mdl_build	Generate code and compile the code on the target system
srt_mdl_get	Get the specified application from the target system
srt_mdl_kill	Kill the model that is currently running on the target system
srt_mdl_list	List the files in the target's application directory
srt_mdl_put	Put an application on the target system's harddisk
srt_mdl_rm	Remove a file from the target's application directory
srt_mdl_run	Run the specified application on the target system
srt_reboot	Reboot the target system
srt_save_blocks	Put the current subdirectory 'blocks' on the harddisk
srt_status	Show the current status of the target
srt_target	Create a target handle object. The target handle is required for all commands that interact with the target system

share that is used is $\backslash\backslash inmar\_2\backslash srt$. In this document it has been assumed that the share is connected to the $s:$ drive letter. The directories, on this share, that are of interest can be found in $s:\backslash blocks$. In this directory, the following sub-directories can be found: $platform$, $rtllinux$ and $user$. The directory $platform$ contains various functions for remote data acquisition and other platform specific functions. It also contains a Simulink library 'librtlinux' that provides functions for remote data acquisition. The $rtlinux$ directory also contains some support functions. Finally, the $user$ directory contains the code for the custom build

simulink c-code components. Further details can be found in the documentation provided by TNO. In the next paragraph it is explained how to compile and build the code/components. A Simulink model is built in C as described in the Simulink manuals. All platform-specific code must be placed between the correct conditional compilation directives in C, as follows:

```
static void mdlStart(SimStruct S)
{
#ifndef MATLAB_MEX_FILE
   Realtime code
#else
   Windows code
#endif  MATLAB_MEX_FILE
   common code
}
```

The code is copied to the *user* directory. When the code is ready it needs to be compiled and uploaded to the platform. This can be accomplished with the following sequence of commands:

`plat = 'inmar_2';`	Name of the platform
`user = 'simulink';`	Name of the user on the specified platform
`model = 'system_ident';`	Name of the model to use
`t = srt_target(plat,user);`	Get a handle for the target
`srt_mdl_build(t,model);`	Compile and build the realtime kernel module
`srt_mdl_run(t,model);`	Upload and run the model (It is not yet started)
`eval (model);`	Open the simulink model such that the user can interact with it

The RealTime Workshop is used to generate the code for the Simulink model. The custom built functions are not included in this code. After the realtime workshop has built the code it is uploaded to the platform. This code together with the C-code stored in the *user* directory is used to generate the final kernel module using the platform-specific compiler. This module is then downloaded and stored on the host computer. The module can be uploaded later; this is the task of the `srt_mdl_run` command. When the module is successfully uploaded it can be started using the remote Simulink interface. To accomplish this, it is necessary to first connect to the platform and later start it. It is also possible to permanently upload a module and start it after the boot procedure.

The environment also provides the possibility to communicate with the platform by means of a network connection. It is possible to read a single variable, a vector and even an array of different types. This environment is realized using the remote data acquisition interface (RDA). The **librtlinux** library provides several Simulink components to realize this. The three components provided for this functionality are **rda_write**, **rda_read** and **rda_var**. The first

two components can be used for simple double scalar values. The last one is more complex and can be used for scalars and different types, but it also adds support for vectors and matrices. The value of an RDA variable can be read or written with a program called **rdaview**. This program makes it possible to inspect the different variables. However, the platform also provides a series of commands for accessing the variables from within Matlab. The list of commands in Table 6.3 is not complete but shows the most important commands. In the next paragraph a simple example of the use of these functions

Table 6.3: *List of supported RDA matlab commands for use with the Simulink Realtime Target SRT.*

Function name	Description
rda_connect	Gets a handle that needs to be used to access the target
rda_disconnect	Disconnects from the target
rda_list	Lists all the RDA variables available on the target
rda_getvar	Get the content of a RDA variable

is given. The example used is based on the model in Figure 6.2. In this model an RDA variable called G_sec is used; this is a vector with a specified length. The variable can be read remotely with the following sequence of commands:

```
t=rda_connect('inmar_2')                  Connect to the srt target
                                          with the name "inmar_2"
t=rda_list(t,'*')                         List all RDA variables
..                                        Other variables
float64 system_ident.G_sec(500)           Name of the variable
..                                        Other variables
G_sec = ...
 rda_getvar(t,'system_ident.G_sec');      Get the value of the variable
```

Finally, there are some RDA variables which are always present. These can be read out to get for example the execution time of the algorithm, the period time. It indicates the minimum, the maximum, the average and the standard deviation of these quantities. This can be used to see if the model is running correctly. Several other variables are available but only the most important ones area explained. The name of the variable and their meaning can be found in Table 6.4. All variables have the 'fwk' prefix. The variables can be found in the module. For instance, the average period time for the **system_ident** can be read using the notation **system_ident.fwk_dt_step_mean**.

6.7 Proof of concept

The overall system was tested by means of a simple verification. The system used was a single-channel filtered-reference feedforward controller. The test consisted of two parts: first the impulse response of the system was estimated; secondly the

FXLMS algorithm was executed. The system was identified by means of a LMS algorithm. After a certain adaptation time, the coefficients of the LMS-algorithm contain a moving average (MA) model of the plant. The Simulink model used for the system identification can be found in Figure 6.2. The transfer function of the plant can be found in Figure 6.4. With the impulse response found in the previous step, the FXLMS algorithm can be used to cancel the disturbance. The FXLMS algorithm model used in Simulink can be found in Figure 6.3. The system was tested by means of a simple sine wave. The frequency was adjusted to one of the resonance frequencies and the controller was switched on. The reduction in such a resonance frequency was approximately 30 dB. This simple test shows that the hardware and the interface with Simulink work as expected.

Table 6.4: *List of RDA variables exported in a Simulink Realtime Target (SRT).*

Function name	Description
float64 fwk_dt_step_max	Maximum sampling time
float64 fwk_dt_step_min	Minimum sampling time
float64 fwk_dt_step_mean	Average sampling time
float64 fwk_dt_step_sigma	STD of the sampling time
float64 fwk_dt_period_max	Maximum execution time
float64 fwk_dt_period_min	Minimum execution time
float64 fwk_dt_period_mean	Average execution time
float64 fwk_dt_period_sigma	STD of the execution time
uint32 fwk_num_overruns	Number of overruns
uint32 fwk_num_steps	Executed time steps
string fwk_simulink_error_status(100)	Error status string
string fwk_model_status(100)	Model state string

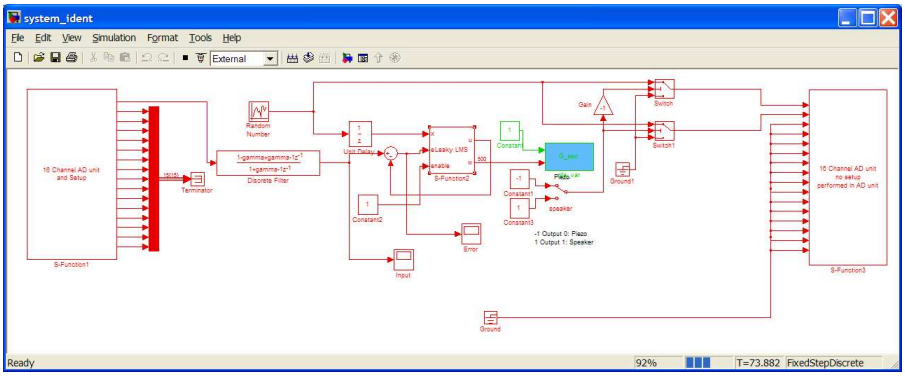


Figure 6.2: *Simulink model used for system identification.*

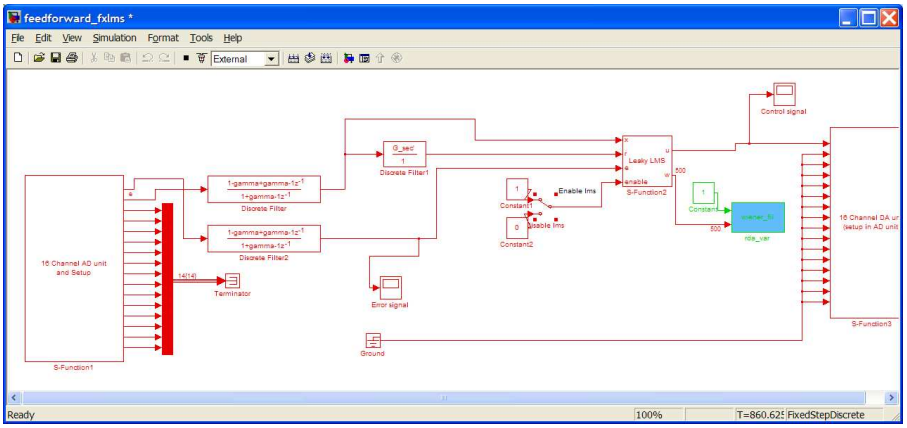


Figure 6.3: *Simulink model used for the FxLMS algorithm.*

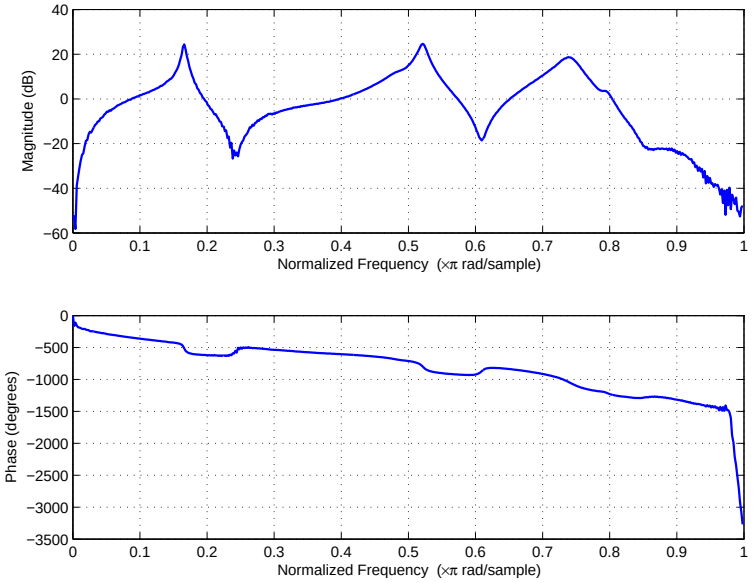


Figure 6.4: *Magnitude and phase of the estimated secondary path.*

Chapter 7

Conclusions

7.1 Conclusions

This thesis describes a rapid prototyping system for broadband multichannel active noise and vibration control. Although the system was designed as a development system for different applications in active noise and vibration control, the focus was on the application of piezoelectrically excited panels for reduction of sound transmission. This panel was used as a demonstrator which was built for the InMAR project so that its performance and applicability could be evaluated. A testing environment was needed to quickly evaluate different control strategies, for example centralized and decentralized control. From this the following research question was derived: "**To design a multi-channel active noise and vibration control system. It should be suited for different applications including mobile demonstrators such as in cars, it should provide a rapid prototyping environment and it should provide a possibility for stand-alone operation.**" This overall research question can be broken down into several other questions which are listed in 1.4.1 and will be addressed separately.

What kind of platform is needed to quickly evaluate different algorithms using a rapid prototyping approach?
The system used to implement new algorithms is based on Simulink/Matlab and the Matlab Real-Time Workshop (RTW). In this environment a simulation model from within Simulink can be used to automatically generated code for a given target device. The target device used in this thesis uses Real-Time Linux (RTLinux) as its operating system. Simulink uses a graphical programming interface and predefined components, which can be used to realize complex diagrams. New components can be realized by either combining simple component or by using an foreign language interface. The programming languages used to interface with Simulink was C. The C-code is translated into a new component which can be used in Simulink. For the RTLinux target this code is used to build a real-time

kernel module. The main advantage of this approach is, that new components can be simulated before they are tested on the real-time environment, reducing development time and errors during the prototyping phase.
The target device is an embedded PCI-104 platform, which is basically a PC in a different form factor. The PCI-104 card uses a Pentium-M running on 1.4 GHz. A FPGA is used to connect the PCI-104 interface to the analog to digital (AD) and digital to analog (DA) converters. The FPGA uses an interpolator and decimator therefore making it possible to realize different sample rates for the RTLinux platform. The interpolator is used as a reconstruction filter and the decimator is used as an anti-aliasing filter. The output and input sample rate are 100 kHz, if the interpolator and decimator are both used. It is also possible to switch off the decimator and/or the interpolator, resulting in a sample rate that is equal to the RTLinux sample rate. An advanced triggering system is provided that allows for a fractional delay, without the need of starting the conversion process in software. This was realized by a master-slave timer system. The main advantage of this approach is that it results in a lower sample jitter. Two prototypes were designed during the research period. The second generation prototype also features a local feedback circuit. It consists of a concatenation of 3 second-order filters and a scaling unit. The scaling unit can be selected to be at the input our output of one of the second-order filters. The output to which a given local feedback path should be added can be selected. This architecture was used to realize a low authority control (LAC) architecture. The hardware is suitable for many applications and a wide range of sample rates. The PC platform provides sufficient computational power to control the test setup without problems.

What kind of control algorithm is needed? FxLMS, FeLMS, RMFeLMS, robust control such as H_∞ and so on.
An adaptive algorithm was selected in order to be able to track non-stationary signal statistics. The filtered-error and the filtered-reference least mean square algorithms (FeLMS and FxLMS, respectively) are the most commonly used adaptive algorithms for active noise and vibration control. The FxLMS algorithm suffers from poor scaling if the number of reference signals is large. The FeLMS algorithm is better suited for multiple reference signals. In the case of the FxLMS algorithm each reference signal must be filtered with each of the transfer functions, which is not necessary in the case of FeLMS algorithm. The noncausal nature of the FeLMS algorithm makes it necessary to introduce a delay in the adaptation loop and the reference signals, making the algorithm causal again. This reduces the adaptation speed of the algorithm. Both algorithms suffer either from poor performance due to computational complexity or from poor convergence properties. The RMFeLMS algorithm was selected since is does not suffer from the afore mentioned problems. A method was derived that estimates the theoretical performance based on a finite impulse response Wiener filter realization. The difference between the measured and the predicted mean square error (MSE) is 1.4 dB in the case of the feedback algorithm and 0.3 dB in the case of the feedforward algorithm. The convergence rate and the MSE of the RMFeLMS algorithm degrade if the reference signals are

colored. A derivation was given of an extension to circumvent this limitation, in which the LMS part of the RMFe structure was replaced with an affine projection (AP) algorithm. The computational complexity of the AP algorithm made it necessary to introduce a fast version of the affine projection method (FAP). In order to be able to test the AP and FAP extension, a measurement setup was devised that introduced colored reference signals. It was shown, by means of simulations and real-time experiments, that FAP/AP improves the convergence speed in the case of strongly colored reference signals. The FAP/AP method only improves convergence speed for feedforward control. Convergence speed did not improve for feedback control due to the inherent whitening properties of a feedback controller, as expected.

How should the architecture of the controller look like? Should the controller be centralized or decentralized. Is it necessary to use a HAC/LAC architecture?

The actively controlled panel uses a MIMO control architecture. In the case of a multiple-SISO system the design of a decentralized controller can become quite challenging due to the fact the different loops may counteract each other. This can be circumvented if the the sensor-actuator pairs are collocated and dual and if fixed gain feedback is used. In this case i.e. for arbitrary feedback gains, the stored energy in the system is always reduced, ensuring stability of the overall system. The platform introduced in this thesis is suited for centralized as well as for decentralized control. However, the systems presented in this thesis use centralized MIMO control or a mixture of decentralized control and centralized MIMO control. The high-authority and low-authority (HAC/LAC) control architecture is an example of a mixture of centralized and decentralized control. The main advantages of such a structure is that it decreases the overall MSE and increases the robustness of a system. In a normal HAC/LAC architecture the LAC architecture is realized as a simple analog controller. In this thesis a method is introduced that uses multi-rate signal processing. The analog LAC architecture is then approximated by means of oversampling. An analog LAC architecture consists of a simple first-order or second-order network. A simple second-order digital filter can be used to approximate such a LAC architecture. The high-authority architecture is running at a lower sample rate, typically 250 Hz to 12 kHz. The HAC/LAC method was verified on an actively controlled panel. For the LAC architecture, an acceleration sensor and a piezoelectric actuator were used. This pair was dual and collocated for frequencies up to approximately 1 kHz. A carefully designed LAC architecture prevents instability for higher frequencies. The HAC loop used the same actuator as the LAC loop, and a separate piezoelectric sensor that was aligned with the actuator, but on the other side of the panel. In the case of feedforward control, the HAC loop used a model-based MIMO controller. The feedback HAC loop was realized using IMC in combination with a model-based feedforward controller. The LAC controller introduces additional damping into the structure. In the case of a feedforward controller, the MSE improved with approximately 4 to 5 dB. The robustness also improved. This was verified

by adding weight to the panel. The model used for the controller in this case is that of the system without any additional weight, which resulted in a mismatch between the real plant and the model. The MSE of the system with and without LAC was measured under influence of this weight. The RMFeLMS algorithm is already quite robust and therefore it already works with some additional weight. However, it was shown, that, depending on the regularization parameter β, the weight that could be added increased between 1.4 to 1.8 times. In the case of a feedback controller that used internal model control (IMC) the MSE only improved slightly with approximately 1 dB. However, the improvement in robustness is better than that of the feedforward controller. This is already true when no LAC is used, in this case the weight that can be added to the panel is already 1.8 times higher than that of the feedforward controller. The experiments carried out were similar to that of the feedforward controller. The improvement again depended on the regularization parameter β, but the weight that could be added now increased between 1.4 to 2.8 times. The best robustness was realized with a normalized regularization value β of -20 dB, thereby increasing the maximum MSE but improving the overall robustness. It can be concluded that a HAC/LAC architecture, if applicable for a certain application, increases the robustness and reduces the MSE for both feedback as well as for feedforward control. The phase-shift of the secondary plant reduces if the LAC loop is realized as a collocated dual pair introducing active damping. The LAC loop also reduces the influence of the phase shift under parametric uncertainty. This was shown in an experiment where the phase difference was measured for a system with and without weight. The experiment was repeated twice in the first experiment the LAC unit was disabled and in the second experiment the LAC unit was enabled. The phase difference in the experiment using LAC was less than that of the system without LAC, resulting in a system that is less sensitive for parametric uncertainty. A disadvantage of this architecture is that it requires two sensors instead of one. The digital HAC/LAC architecture as presented in this thesis results in a overall more robust system and the overall MSE improves. It only requires moderate hardware changes and most of these were already present in the existing architecture.

Appendix A

Derivation of the FAP algorithm

Starting with the affine projection update rule Eq. (3.40), expressing the current filter taps in the original filter estimate and all subsequent $\mathbf{X}(n)$ and $\boldsymbol{\xi}(n)$, and assuming that the system starts at $n = 0$ with the delay lines set to zero, results in:

$$\mathbf{W}(n+1) = \mathbf{W}(0) - \mu \sum_{i=0}^{n-1} \mathbf{X}(n-i)\boldsymbol{\xi}(n-i). \tag{A.1}$$

The vector-matrix product is expanded into

$$\mathbf{W}(n+1) = \mathbf{W}(0) - \mu \sum_{i=0}^{n-1} \sum_{j=0}^{K_A-1} \underline{x}'(n-j-i)\xi_j(n-i), \tag{A.2}$$

in which $\xi_j(n)$ is defined as the $j-1$-th row of $\boldsymbol{\xi}(n)$, i.e., $\boldsymbol{\xi}(n) = \left[\xi_0^T(n)\xi_1^T(n)\ldots\xi_{K_A-1}^T(n)\right]^T$. Exchanging the summations and substitution of $k = j + i$ and $i = k - j$ leads to

$$\mathbf{W}(n+1) = \mathbf{W}(0) - \mu \sum_{j=0}^{K_A-1} \sum_{k=j}^{j+n-1} \underline{x}'(n-k)\xi_j(n-k+j). \tag{A.3}$$

The second sum is split into two parts: one from $k = j$ to $k = K_A - 1$ and one from $k = K_A$ to $k = j + n - 1$, such that

$$\begin{aligned} \mathbf{W}(n+1) &= \mathbf{W}(0) - \mu \sum_{j=0}^{K_A-1} \sum_{k=K_A}^{j+n-1} \underline{x}'(n-k)\xi_j(n-k+j) \\ &- \sum_{j=0}^{K_A-1} \sum_{k=j}^{K_A-1} \underline{x}'(n-k)\xi_j(n-k+j). \end{aligned} \tag{A.4}$$

It can be shown that the j in the upper bound of the second sum of the first double summation in Eq. (A.4) can be removed because $\underline{x}'(n) = 0$ for $n \leq 0$. Substitution of the upper bound in $\underline{x}'(n-k)$ results in $\underline{x}'(-j+1)$, i.e. $\underline{x}'(1)$ for $j = 0$ and $\underline{x}'(0)$ for $j = 1$ and so on. This means that the terms with $j > 0$ will not contribute to the summation and can therefore be removed. The first double summation, denoted as S_1, can thus be written as

$$S_1 = \mu \sum_{k=K_A}^{n-1} \underline{x}'(n-k) \sum_{j=0}^{K_A-1} \xi_j(n-k+j). \tag{A.5}$$

The second double sum in Eq. (A.4), denoted as S_2, can also be expressed in a form similar to that of Eq. (A.5) after reordering of the different terms. Expansion of the second double summation shows that:

$$\begin{aligned} S_2 = \underline{x}'(n)\xi_0(n) + \underline{x}'(n-1)\xi_0(n-1) + \ldots + \\ \underline{x}'(n-K_A+1)\xi_0(n-K_A+1)) + \\ \underline{x}'(n-1)\xi_1(n) + \underline{x}'(n-2)\xi_1(n-1) + \ldots + \\ \underline{x}'(n-K_A+1))\xi_1(n-K_A+2) + \\ \vdots \\ \underline{x}'(n-K_A+1)\xi_{K_A-1}(n). \end{aligned}$$

From the latter expansion it can be seen that the second double summation can be written as

$$S_2 = \mu \sum_{k=0}^{K_A-1} \underline{x}'(n-k) \sum_{j=0}^{k} \xi_j(n-k+j). \tag{A.6}$$

Using Eqs. (A.5) and (A.6), Eq. (A.2) can be expressed as

$$\begin{aligned} \mathbf{W}(n+1) &= \mathbf{W}(0) - \mu \sum_{k=0}^{K_A-1} \underline{x}'(n-k) \sum_{j=0}^{k} \xi_j(n-k+j) \\ &- \mu \sum_{k=K_A}^{n-1} \underline{x}'(n-k) \sum_{j=0}^{K_A-1} \xi_j(n-k+j). \end{aligned} \tag{A.7}$$

Equation (A.5) and $\mathbf{W}(0)$ are used to define $\hat{\mathbf{W}}(n)$:

$$\begin{aligned} \hat{\mathbf{W}} \quad (n) = \mathbf{W}(0) - \\ \mu \sum_{k=K_A}^{n-1} \underline{x}'(n-k) \sum_{j=0}^{K_A-1} \xi_j(n-k+j). \end{aligned} \tag{A.8}$$

Equation (A.6) will be rewritten as

$$S_2 = \mu \sum_{k=0}^{K_A-1} \underline{x}'(n-k) h(n,k), \tag{A.9}$$

in which

$$h(n,k) = \sum_{j=0}^{k} \xi_j(n-k+j). \tag{A.10}$$

In the right-hand side of Eq. (A.9) a simple sum of base vectors $\underline{x}'(n-k)$ multiplied by a scaling coefficient $h(n,k)$ can be recognized. Let us define the following vector of scaling coefficients:

$$\begin{aligned} \underline{\mathbf{E}}(n) &= \begin{bmatrix} h(n,0) \\ \vdots \\ h(n,K_A-1) \end{bmatrix} = \\ & \begin{bmatrix} \xi_0(n) \\ \xi_1(n)+\xi_0(n-1) \\ \vdots \\ \xi_{K_A-1}(n)+\ldots+\xi_0(n-(K_A-1)) \end{bmatrix}. \end{aligned} \tag{A.11}$$

The base is now equal to the $\mathbf{X}(n)$ matrix, so that the right-hand side of Eq. (A.9) can be written as the matrix-vector product $\mu\mathbf{X}(n)\underline{\mathbf{E}}(n)$. Then Eq. (A.7) can be written as

$$\mathbf{W}(n+1) = \hat{\mathbf{W}}(n) - \mu\mathbf{X}(n)\underline{\mathbf{E}}(n). \tag{A.12}$$

Equation (A.8) is expressed as an iterative update rule

$$\begin{aligned} \hat{\mathbf{W}}(n+1) &= \hat{\mathbf{W}}(n) - \mu\underline{x}'(n-(K_A-1)) \times \\ &\quad \sum_{j=0}^{K_A-1} \xi_j(n-(K_A-1)+j) \\ &= \hat{\mathbf{W}}(n) - \mu\underline{x}'(n-(K_A-1)) \times \\ &\quad \underline{E}_{K_A-1}(n), \end{aligned} \tag{A.13}$$

in which $\underline{E}_{K_A-1}(n)$ is defined as the bottom row of $\underline{\mathbf{E}}(n)$. Equations (A.12) and (A.13) are used to express the updated controllers as

$$\mathbf{W}(n+1) = \hat{\mathbf{W}}(n+1) - \mu\overline{\mathbf{X}}(n)\overline{\underline{\mathbf{E}}}(n), \tag{A.14}$$

in which $\overline{\underline{\mathbf{E}}}(n)$ are the K_A-1 top-most rows of $\underline{\mathbf{E}}(n)$ and $\overline{\mathbf{X}}(n)$ are the K_A-1 left-most columns of $\mathbf{X}(n)$.
Using Eq. (A.11), this can be written recursively as

$$\underline{\mathbf{E}}(n) = \boldsymbol{\xi}(n) + \begin{bmatrix} \underline{0}_M^T \\ \overline{\underline{\mathbf{E}}}(n-1) \end{bmatrix}, \tag{A.15}$$

in which $\underline{0}_M$ is a column vector filled with M zeros and $\overline{\underline{\mathbf{E}}}$ are the upper K_A-1 rows of $\underline{\mathbf{E}}$.

Bibliography

[1] Y. Osada, "An overview of health effects on noise," *Journal of Sound and Vibration*, vol. 127, pp. 407–410, 1988.

[2] H. Sano, T. Inoue, A. Takahashi, T. Yamashita, M. Nakamura, K. Terai, and Y. Nakamura, "Development of active control system for low frequency road noise: Solutions for practical use and system configuration," *Acoustical Science and Technology*, vol. 22, pp. 378–379, 2001.

[3] U. Emborg, "Cabin noise control in the Saab 2000 high-speed turboprop aircraft," in *Proceedings of the 23rd International Conference on Noise and Vibration Engineering, ISMA*, Heverlee, B-3001, Belgium, 1998, pp. 1–13, Katholieke Universiteit Leuven.

[4] S. J. Elliott, P. A. Nelson, I. M. Stothers, and C. C. Boucher, "In-flight experiments on the active control of propeller-induced cabin noise," *Journal of Sound and Vibration*, vol. 140, pp. 219–238, 1990.

[5] S. Johansson, I. Claesson, S. Nordebo, and P. Sjösten, "Evaluation of multiple reference active noise control algorithms on Dornier 328 aircraft data," *IEEE Transactions on Speech and Audio Processing*, vol. 7, pp. 473–477, 1999.

[6] S. Johansson and I. Claesson, "Active Noise Control in Propeller Aircraft," in *Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, Ronneby, Sweden, 2001, pp. 111–120, The Knowledge Foundation.

[7] W. S. Gan and S. M. Kuo, "Integrated active noise control communication headsets," in *Proceedings of the 2003 IEEE International Symposium on Circuits and Systems*, New York, NY, 2003, pp. IV–353–IV–356, IEEE.

[8] Y. Nakaji, S. Satoh, T. Kimura, T. Hamabe, Y. Akatsu, and H. Kawazoe, "Development of an Active Control Engine Mount System," *Vehicle System Dynamics*, vol. 32, pp. 185–198, 1999.

[9] L. J. Eriksson and M. C. Allie, "Use of random noise for on-line transducer modeling in an adaptive active attenuation system," *The Journal of the Acoustical Society of America*, vol. 85, pp. 797–802, 1989.

[10] T. C. Sors and S. J. Elliott, "Modelling and feedback control of sound radiation from a vibrating panel," *Smart Materials and Structures*, vol. 8, pp. 301–314, 1999.

[11] A. P. Berkhoff, "Piezoelectric sensor configuration for active structural acoustic control," *Journal of Sound and Vibration*, vol. 246, pp. 175–183, 2001.

[12] G. P. Gibbs, R. L. Clark, D. E. Cox, and J. S. Vipperman, "Radiation modal expansion: Application to active structural acoustic control," *The Journal of the Acoustical Society of America*, vol. 107, pp. 332–339, 2000.

[13] O. E. Kaiser, S. J. Pietrzko, and M. Morari, "Feedback control of sound transmission through a double glazed window," *Journal of Sound and Vibration*, vol. 263, pp. 775–795, 2003.

[14] A. Jakob and M. Möser, "Active control of double-glazed windows Part I: Feedforward control," *Applied Acoustics*, vol. 64, pp. 163–182, 2003.

[15] A. Jakob and M. Möser, "Active control of double-glazed windows. Part II: Feedback control," *Applied Acoustics*, vol. 64, pp. 183–196, 2003.

[16] C. R. Fuller, S. J. Elliott, and P. A. Nelson, *Active Control of Vibration*, Academic Press, 24-28 Oval Road, London, NW1 7DX, UK, 2nd edition, 1997.

[17] B.-T. Wang, C. R. Fuller, and E. K. Dimitriadis, "Active control of noise transmission through rectangular plates using multiple piezoelectric or point force actuators," *The Journal of the Acoustical Society of America*, vol. 90, pp. 2820–2830, 1991.

[18] A. Preumont, *Vibration Control of Active Structures*, Kluwer Academic Publishers, Dordrecht, 2nd edition, 2002.

[19] P. A. Nelson and S. J. Elliott, *Active control of sound*, Academic Press, London, 1992.

[20] S. J. Elliott, *Signal Processing for Active Control*, Academic Press, Harcourt Place, 32 Jamestown Road, London NW1 7BY, UK, 2001.

[21] P. Van Overschee and B. De Moor, *Subspace identification for linear systems*, Kluwer Academic Publishers, P.O. Box 17, 3300 Dordrecht, The Netherlands, 1996.

[22] D. Morgan, "An analysis of multiple correlation cancellation loops with a filter in the auxiliary path," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, pp. 454–467, 1980.

[23] S. J. Elliott, I. Stothers, and P. Nelson, "A multiple error LMS algorithm and its application to the active control of sound and vibration," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, pp. 1423–1434, 1987.

[24] E. A. Wan, "Adjoint LMS: an efficient alternative to the filtered-X LMS and multiple error LMS algorithms," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Piscataway, NJ, USA, 1996, pp. 1842–1845, IEEE.

[25] V. E. DeBrunner and Z. Dayong, "Hybrid filtered error LMS algorithm: another alternative to filtered-x LMS," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 53, pp. 653–661, 2006.

[26] S. Ishimitsu and S. J. Elliott, "Improvement of the convergence property of adaptive feedforward controllers and their application to the active control of ship interior noise," *Acoustical Science and Technology*, vol. 25, pp. 181–187, 2004.

[27] J. G. Cook and S. J. Elliott, "Connection between multichannel prediction error filter and spectral factorisation," *Electronics Letters*, vol. 35, pp. 1218–1220, 1999.

[28] M. R. Bai and S. J. Elliott, "Preconditioning multichannel adaptive filtering algorithms using EVD- and SVD-based signal prewhitening and system decoupling," *Journal of Sound and Vibration*, vol. 270, pp. 639–655, 2004.

[29] A. P. Berkhoff and G. Nijsse, "A rapidly converging filtered-error algorithm for multichannel active noise control," *International Journal of Adaptive Control and Signal Processing*, vol. 21, pp. 556–569, 2007.

[30] S.J.Elliott, *Active Sound and Vibration Control*, , edited by M.O.Tokhi and S.M.Verse, chapter 3, Adaptive methods in active control, pp. 57–72, IET, Londen, 2002.

[31] K. Ozeki and T. Umeda, "Adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electronics and Communications in Japan (English translation of Denshi Tsushin Gakkai Zasshi)*, vol. 67, pp. 19–27, 1984.

[32] S. L. Gay, "Fast projection algorithms with apllication to voice echo cancellation," Ph. D. thesis, The State University of New Jersey, 1994.

[33] S. L. Gay and S. Tavathia, "The fast affine projection algorithm," in *ICASSP-95, IEEE International Conference on Acoustics, Speech, and Signal Processing - Proceedings*, New York, NY, 1995, pp. 3023–3026, IEEE.

[34] M. Bouchard and F. Albu, "The Gauss-Seidel fast affine projection algorithm for multichannel active noise control and sound reproduction systems," *International Journal of Adaptive Control and Signal Processing*, vol. 19, pp. 107–123, 2005.

[35] D. Heping, "A stable fast affine projection adaptation algorithm suitable for low-cost processors," in *ICASSP '00, IEEE International Conference on Acoustics, Speech, and Signal Processing - Proceedings*, New York, NY, 2000, pp. 360–363, IEEE.

[36] S. Oh, D. Linebarger, B. Priest, and B. Raghothaman, "A fast affine projection algorithm for an acoustic echo canceller using a fixed-point DSP processor," in *ICASSP '97, IEEE International Conference on Acoustics, Speech, and Signal Processing*, New York, NY, 1997, pp. 4121–4124, IEEE.

[37] D. Heping, "Fast affine projection adaptation algorithms featuring stable symmetric positive-definite linear system solvers," in *2005, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, NY, 2005, pp. 166–169, IEEE.

[38] S. C. Douglas, "Efficient approximate implementations of the fast affine projection algorithm using orthogonal transforms," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Atlanta, GA, USA, 1996, pp. 1656–1659, IEEE, Piscataway, NJ, USA.

[39] M. Rupp, "A family of adaptive filter algorithms with decorrelating properties," *IEEE Transactions on Signal Processing*, vol. 46, pp. 771–775, 1998.

[40] R. Fraanje, "Robust and fast schemes in broadband active noise and vibration control," Ph.D thesis, University of Twente The Netherlands, 2004.

[41] K. J. Åström and R. M. Murray, *Feedback systems : an introduction for scientists and engineers*, Princeton [etc.] : Princeton University Press, Princeton and Oxford, 2008.

[42] S. Herold, D. Mayer, and H. Hanselka, "Transient Simulation of Adaptive Structures," *Journal of Intelligent Material Systems and Structures*, vol. 15, pp. 215–224, 2004.

[43] G. V. Borgiotti, "The power radiated by a vibrating body in an acoustic fluid and its determination from boundary measurements," *The Journal of the Acoustical Society of America*, vol. 88, pp. 1884–1893, 1990.

[44] S. J. Elliott and M. E. Johnson, "Radiation modes and the active control of sound power," *The Journal of the Acoustical Society of America*, vol. 94, pp. 2194–2204, 1993.

[45] O. N. Baumann and S. J. Elliott, "Decentralized control using multiple velocity feedback loops with inertial actuators," in *Proceedings of Active 2006 the 2006 International Symposium on Active Control of Sound and Vibration*, Adelaide, 2006, School of Mechanical Engineering, University of Adelaide.

[46] P. Gardonio, E. Bianchi, and S. J. Elliott, "Smart panel with multiple decentralized units for the control of sound transmission. Part I: theoretical predictions," *Journal of Sound and Vibration*, vol. 274, pp. 163–192, 2004.

[47] P. Gardonio, E. Bianchi, and S. J. Elliott, "Smart panel with multiple decentralized units for the control of sound transmission. Part II: design of the decentralized control units," *Journal of Sound and Vibration*, vol. 274, pp. 193–213, 2004.

[48] E. Bianchi, P. Gardonio, and S. J. Elliott, "Smart panel with multiple decentralized units for the control of sound transmission. Part III: control system implementation," *Journal of Sound and Vibration*, vol. 274, pp. 215–232, 2004.

[49] A. P. Berkhoff, "Broadband radiation modes: Estimation and active control," *Journal of the Acoustical Society of America*, vol. 111, pp. 1295–1305, 2002.

[50] A. P. Berkhoff and J. M. Wesselink, "Multichannel active noise control systems and algorithms for reduction on broadband noise," in *Proceedings of Adaptronic Congress 2007*, Gottingen, Germany, 2007, pp. 1–4, Adaptronic Congress Veranstaltungs GbR.

[51] A. P. Berkhoff, Wesselink J.M., and T. G. H. Basten, "Adaptive reductions of the vibrations from a vacuum pump for high-precision equipment," in *Proceedings of Adaptronic Congress 2008*, Berlin, Germany, 2008, pp. 53–56, Adaptronic Congress Veranstaltungs GbR.

[52] A. P. Berkhoff, J. M. Wesselink, and T. G. H. Basten, "Active vibration control applied to a vacuum pump for high-precision equipment," in *37th International Congress and Exposition on Noise Control Engineering*, Shanghai, China, 2008, Institute of Noise Control Engineering.

[53] A. P. Berkhoff and J. M. Wesselink, "Combined MIMO adaptive and decentralized controllers for broadband active noise and vibration control," in *Proceedings of ACTIVE 2009*, Ottawa, Canada, 2009, p. Submitted, Institute of Noise Control Engineering.

[54] S. J. Elliott, "Optimal controllers and adaptive controllers for multichannel feedforward control of stochastic disturbances," *IEEE Transactions on Signal Processing*, vol. 48, pp. 1053–1060, 2000.

[55] A. H. Sayed, *Fundamentals of Adaptive Filtering*, IEEE Press, A John Wiley & Sons, Inc, United States of America, 2003.

[56] E. A. Robinson, *Multichannel time series analysis with digital computer programs*, Holden-Day, San Francisco, 1978.

[57] S. M. Kuo and R. D. Morgan, *Active Noise Control Systems*, John Wiley & Sons, Inc, United States of America, 1996.

[58] J. M. Wesselink and A. P. Berkhoff, "Fast affine projections and the regularized modified filtered-error algorithm in multichannel active noise control," *The Journal of the Acoustical Society of America*, vol. 124, pp. 949–960, 2008.

[59] M. Vidyasagar, *Control system synthesis: A factorization approach*, MIT Press, Boston, 1985.

[60] V. Ionescu, C. Oara, and M. Weiss, *Generalised Riccati theory and robust control : a Popov function approach*, John Wiley & Sons Ltd, Chichester, England, 1999.

[61] G. H. Golub and C. F. Van Loan, *Matrix Computations*, John Hopkins, Baltimore, 2 edition, 1989.

[62] R. A. Roberts and C. T. Mullis, *Digital signal processing*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1987.

[63] A. P. Berkhoff, "Control strategies for active noise barriers using near-field error sensing," *Journal of the Acoustical Society of America*, vol. 118, pp. 1469–1479, 2005.

[64] G. Gatti, M. J. Brennan, and P. Gardonio, "Active damping of a beam using a physically collocated accelerometer and piezoelectric patch actuator," *Journal of Sound and Vibration*, vol. 303, pp. 798–813, 2007.

[65] A. P. Berkhoff and J. M. Wesselink, "Centralised and decentralised configurations for panels with piezoelectric actuators," in *Proceedings of Euronoise 2006*, Tampere, Finland, 2006, p. 429, European Acoustics Association.

[66] J. M. Wesselink, A. P. Berkhoff, G. J. Laanstra, and H. Kuipers, "Implementation issues of a high-speed distributed multi-channel ADDA system," in *International Workshop on Acoustic Echo and Noise Control IWAENC*, Eindhoven, 2005, IWAENC.

[67] T. Shanley and D. Anderson, *PCI System Architecture*, Mind Share, inc, 2000.

[68] K. J. Åström and B. Wittenmark, *Computer-controlled systems : theory and design*, Prentice Hall, Upper Saddle River, NJ, 3th edition, 1997.

[69] J. M. Wesselink and A. P. Berkhoff, "Optimization of the reconstruction and anti-aliasing filter in a Wiener filter system," in *Proceedings Euronoise 2006*, Tampere, 2006, European Acoustics Association.

[70] W. F. W. Mulckhuyse and S. E. Skolnik, "Simulink to RTLinux code generation environment," Tech. Rep., TNO, 2003.

Acknowledgments

The first time I heard of the InMAR project was in the summer of 2003. I knew that Arthur (my assistant promoter) was looking for a Ph.D student. At that time I worked at the university as part of the technical support staff performing work for other Ph.D students. I knew Arthur from some projects in which I also designed some hardware for active noise applications. When Arthur told me that he needed a Ph.D. student I responded and I got the job. At the end of 2003 my contract finished and I still needed to wait for three months before the Ph.D. position started. In march 2004 the project called InMAR finally started. It also was the start of my Ph.D work. It became quickly clear that it would be a hard and at some times dark path. I would like to thank Arthur for giving me this great opportunity and to guide me through this difficult process.
The InMAR project was a really great experience. The project included several foreign partners and required frequent traveling and giving presentations. I would like to thank the European commission and all partners which were involved in the InMAR project for giving me this great experience. I especially would like to thank Marko Antila from VTT for his nice talks and being great company.
It became quickly clear to me that the hardware realized in previous project would not suffice. To circumvent this a new architecture was designed and implemented based on an embedded PC. At that time TNO offered us the possibility to use their realtime system which was based on RealTime Linux and Simulink/Matlab. This helped enormously, making it 'only' necessary to develop a software driver to access our hardware platform. For this I would like to thank Wouter Mulckhuyse, Sandor Skolnik and Remco den Breeje. The interfacing card and all of its analog problems were solved by Geert Jan Laanstra and Henny Kuipers. I would not have been able to finish the work in time without their support. I also would like to thank Niels Moseley for his insight in IIR filters and PLLs.
Furthermore, I would like to thank all my colleagues for their support in all possible ways. Special thanks go to Joost Kaufman and Roel Schiphorst for providing the template for this thesis. Finally, I would like to thank my parent and my sisters without whom the difficult process of finishing my thesis would not have been successful.