UNIVERSITEIT·STELLENBOSCH·UNIVERSITY
jou kennisvennoot • your knowledge partner

Feedforward Active Noise Reduction for Aircraft Headsets

By

Corné J. Smith

Thesis presented in partial fulfilment of the requirements for the degree of Masters in Engineering at the University of Stellenbosch

Promoter

Prof. Johan G. Lourens

Date

December 2003

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own work, unless otherwise stated, and has not previously in its entirety or in part been submitted at any university for a degree.

C.J. Smith ______________________________ Date __________________

Synopsis

Active noise reduction (ANR) is a method of cancelling acoustic noise in a defined enclosure. Two methods exist to implement ANR, they are the analog feedback method and the digital feedforward method. Commercial ANR systems employing feedback methods have been around since the 1980's. Feedforward methods have however only become practically implemental with the age of fast real time digital signal processing. In current systems, feedback ANR is used to attenuate broadband noise whilst feedforward methods are used to attenuate narrow band or tonal noise [2].

This thesis investigates feedforward ANR to cancel broadband acoustic noise in aircraft headsets. Different adaptive filters, optimal configuration of adaptive filters and practical limitations to broadband attenuation for headsets are addressed.

Results from this thesis show that at least 10dB noise energy attenuation is attainable over a bandwidth of 2.5kHz. A number of areas for further research are also identified.

Opsomming

Aktiewe geraas beheer (AGB) is 'n metode om akoestiese geraas te kanselleer in 'n gedefinieerde omgewing. Twee metodes bestaan om AGB te implementeer. Hulle is die analoog terugvoer en digitale vorentoevoer metode. Kommersiële AGB wat die terugvoer metode gebruik is al in gebruik van die 1980's. Vorentoevoer metodes is egter eers sedert vinnige intydse digitale sein prosessering moontlik. In huidige stelsels word terugvoer AGB gebruik vir die attenuasie van wyeband geraas terwyl vorentoevoer metodes gebruik word om nouband of enkel toon geraas te kanselleer [2].

Die tesis ondersoek vorentoevoer AGB om wyeband akoestiese geraas te kanselleer in vliegtuig kopstukke. Verskillende aanpasbare filters, optimale opstelling van aanpasbare filters en praktiese beperkings tot wyeband attenuasie vir kopstukke word ondersoek.

Resultate van die tesis wys dat ten minste 10dB geraas energie attenuasie behaal kan word oor 'n bandwydte van 2.5kHz. 'n Aantal areas vir verder navorsing is ook geïdentifiseer.

To Dad

" Swaar kry, met lekker kry, klaar kry "

C.J. Langenhoven

Acknowledgments

I would like to thank the technical personnel of the electronic engineering faculty for their constant support and technical contributions, in particular Nick van Graan, Wessel Croukamp, Ralf Dreyer and Asley Cupido.

I praise my Father for giving me the opportunity to study further and to grow in His grace and will for my life.

I thank my promoter Prof. Johan Lourens for his positive attitude and inspiring talks that made me explore this thesis subject further.

I thank the support of my family and friends who were always enthusiastic to know what my progress in this regard is.

Finally I would like to thank the personnel and friends of the South African Air Force at 5 Air Support Unit of Waterkloof air force base that initiated and supported the cause of this thesis.

Contents

List of Figures

List of Tables

Abbreviations

ac	Alternating current
A/D	Analog to Digital converter
ANR	Active noise reduction
ARMA	Autoregressive mean average
CS	Chip select
D/A	Digital to Analog converter
dB	Decibels
dc	Direct current
DGND	Digital ground
DSP	Digital Signal Processor
FIR	Finite impulse response
Hz	Hertz
ICA	Independent component analysis
I/O	Input output
KSPS	Kilo samples per second
kHz	Kilohertz
KP	Kaczmarz projection
LDAC	Load analog to digital converter
LMS	Least mean squares
LS	Least squares
MSPS	Mega samples per second
Ω	Ohm
Pa	Pascal
R/C	Conversion enable
RLS	Recursive least squares
SAAF	South African Air Force
SCI	Serial communications interface
V	Volt
WR	Write

Chapter 1

1 Introduction

Active noise reduction (ANR) is a method of cancelling unwanted noise by electronically producing a signal that resembles the unwanted noise. The reproduced noise signal is phase shifted by 180 degrees. This signal is better known as anti-noise.

Transmitting the anti-noise in the same cavity as the noise will cause destructive interference between the noise and anti-noise resulting in noise reduction.

Active noise reduction dates back to 1936 with experiments by Lueg [1] to cancel noise in air ducts. Active noise reduction has however only been implemented for commercial applications from the early 1980's.

Thesis background

The South African Air Force (SAAF) acquired their first ANR headset in 1998. These headsets where employed into the current SAAF fleet by 2000. Since their introduction, it was found that noise cancellation produced by Air Force headsets is inadequate for helicopter use.

Two basic methods are employed to produce anti-noise in headsets. These methods are analogue feedback control or digital feedforward control [2]. The current SAAF headsets employ the analogue feedback control method.

The aim of this thesis is to investigate digital feedforward control as a solution for improved broadband noise cancellation in military headsets.

1.1. Digital feedforward control method

Roure, Eriksson and Allie [3], [4] demonstrated that adaptive feedforward control could perform noise reduction from the early 1980's. Operation of the digital feedforward control method in a headset topology can be explained as follow.

Two microphones and one loudspeaker must be mounted on a headset as shown in Figure 1.1. The outside microphone is known as the reference microphone and the inside microphone is known as the error microphone.

The aim of the feedforward method is to estimate a digital filter that resembles the acoustic transfer of noise from the outside to the inside of the headset earpiece. Anti-noise is created by filtering the reference outside noise with the estimated filter and inverting the signal to give it a 180 degree phase difference from the inside noise. The anti-noise is transmitted to the inside of the headset cavity via the loudspeaker.

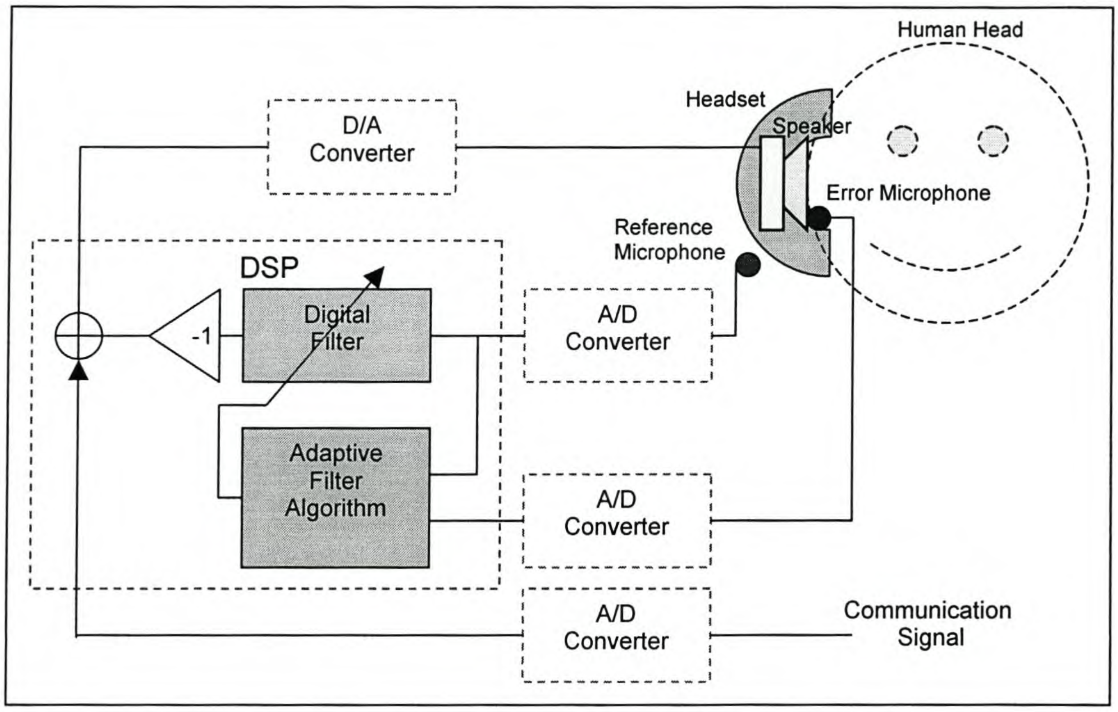


Figure 1.1 Operation of the digital feedforward method

An adaptive filter algorithm is used to estimate the transfer of the headset. A DSP processor performs all the digital filtering and adaptive filter calculations. The complete topology block diagram is shown in Figure 1.1.

1.2. Scope

The following objectives were set for this thesis.

- A realistic simulation environment must be created to simulate the operation of the adaptive feedforward control method.

- A study of different adaptive filter algorithms must be made in an attempt to establish the optimal adaptive filter algorithm for this application.

- A proto type system must be constructed to test the optimal adaptive filter solution for broadband noise.

- Simulated results must be compared to the proto type system measurements.

1.3. System specifications

The current SAAF active noise reduction capability was found to be 8dB attenuation over a 500Hz bandwidth [5]. This capability was accomplished using an analog feedback method and had good omni-directional noise cancellation properties.

The desired active noise reduction capability required by the SAAF was given as at least 8dB attenuation over a 3kHz bandwidth.

In a literature study preceding this thesis, a number of key problems were identified that could complicate achieving the desired feedforward noise cancellation solution. They are outlined here.

- Limited success has been found employing the feedforward method for broadband noise attenuation [2], [6], [7].
- Feedforward noise cancellation has been found to be very directional with respect to the noise source [7].

1.4. Outline

This thesis was broken down into 9 chapters of which this chapter is one.

Chapter 2 describes the creation of a simulation environment to evaluate different adaptive filters. Each component of the simulation is mentioned and the criteria to which different adaptive filters will be evaluated are set in this chapter.

Chapter 3 explains the operation of the LMS algorithm and investigates its performance according to the ANR simulation of chapter 2.

Chapter 4 explains the operation of the RLS algorithm and investigates its performance according to the ANR simulation of chapter 2.

Chapter 5 explains the operation of the Kaczmarz Projection algorithm and investigates its performance according to the ANR simulation of chapter 2.

Chapter 6 compares the simulated results found for the LMS, RLS and Kaczmarz Projection algorithm in an attempt to find the optimal adaptive filter algorithm.

Chapter 7 describes the construction of the proto type system, the acoustic and electrical problems found and solutions to these stumbling blocks.

Chapter 8 explains the measurement setup and attenuation measurements made on the proto type headset headpiece.

Chapter 9 discusses further work that might be applicable to future research.

Chapter 10 concludes this thesis with the contributions of this thesis and a summary.

Chapter 2

2 Active noise reduction simulation model

This chapter investigated the creation of a simulation model to simulate the adaptive feedforward noise cancellation topology for headsets. Different adaptive filter algorithms will be implemented in this simulation for the purpose of finding an optimal adaptive filter solution.

2.1. Basic system configuration

According to [2], [8], [9], and [10] the basic feedforward noise cancellation configuration suited for ANR is similar to each other. The basic configuration is shown in the block diagram of Figure 2.1.

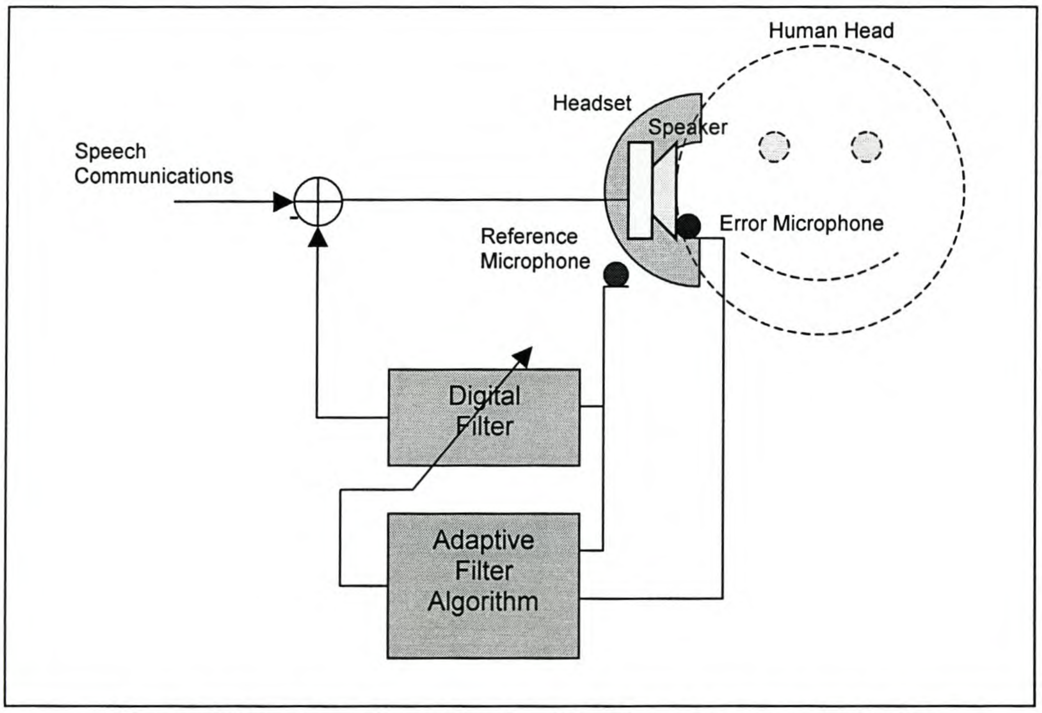


Figure 2.1 Basic configuration for Adaptive active noise reduction

The function of the adaptive filter is to estimate the real time transfer characteristics of the headset from the reference and error microphone signals.

The estimated headset transfer characteristics are revealed in the form of filter coefficients for a digital filter. This filter simulates the behaviour of the acoustic transfer of the headset.

The adaptive filter algorithm has the property that it is updated on-line once every sample of the analogue to digital converters sample rate. This property enables the headset filter to adapt according to possible transfer changes of the headset.

The acoustic headset transfer is expected to change as the user moves his/her head. The seal of the headset earpiece is essential to the actual transfer of the headset since the amount of external noise that seeps in to the inside of the headset through the seal constitutes the transfer of the headset. Since this seal can move when the user moves his head, the headset transfer might change as external noise seeps in to the earpiece of the headset in different amounts.

An adaptively estimated digital filter is thus used to filter the noise from the reference microphone. The filtered output resembles the noise inside the headset earpiece and is subtracted from the speech signal, causing the transmition of 'anti-noise' plus speech by the headset loudspeaker inside the headset cavity. The transmitted 'anti-noise' cancels the noise inside the headset earpiece. A block diagram of this process is shown in Figure 2.1.

2.1.1. Effect of speech being included in the error microphone signal

The error microphone senses the ensuing unwanted noise inside the headset cavity. It also senses the transmitted communications speech signal produced by the headset loudspeaker. This error microphone signal is used with the reference microphone signal for the estimation of the headset transfer filter.

The speech signal included in the error signal might seem unwanted but the speech will have a limited effect on the estimation of the filter parameters since there is expected to be little correlation between the speech signal and the reference noise signal, thus

$$E[\{Reference\ Signal(n)\}\{Speech\ Signal(n)\}] \approx 0 \qquad 2.1$$

But a strong correlation between the error and reference signal exists so that,

$$E[\{Reference\ Signal(n)\}\{Error\ Signal(n)\}] = \boldsymbol{Q}(n) \qquad 2.2$$

$\mathbf{Q}(n)$ represents the time dependant cross correlation between the error and reference signals.

It is this cross correlation feature that enables adaptive calculation of the headset transfer characteristics according to the Normal equations or Weiner-Hopf equation [8], [11].

The Normal equations and Weiner-Hopf equation is very similar and can be expressed as shown in equation 2.3. Equation 2.3 shows this equation were $\mathbf{w}(n)$ is the headset transfer filter tap weights, $\mathbf{Q}(n)$ the time dependant cross correlation matrix and $\mathbf{R}(n)$ the correlation matrix of the reference signal with itself.

$$\mathbf{R}(n)=\mathbf{w}(n)\mathbf{Q}(n) \tag{2.3}$$

Different adaptive filter algorithms that solve equation 2.3 for the headset filter tap weights will be investigated in the following chapters. It should be kept in mind that that it is the property of equation 2.1 that allows the inclusion of an external interference signal (Speech communications) and thus makes the adaptive feedforward method viable for aircraft headset noise cancellation.

When constructing a feedforward system it must thus be kept in mind that the reference microphone should be placed outside the influence of the communications speech of the pilot. This will ensure that equation 2.1 holds, thus ensuring no cancellation of an added communication signal.

2.2. <u>Adaptive filter simulation model</u>

The basic active noise reduction configuration was modelled in Matlab to create a platform to evaluate different adaptive filters for optimum performance. A block diagram of the simulation model is shown in Figure 2.2. This block diagram shows how Figure 2.1 was implemented in a simulation environment.

The composition of each block in the simulation will be discussed throughout the rest of this section.

It must be mentioned that all D/A and A/D converters was assumed to have unity transfer and was left out of the simulation. No anti-aliasing filters were included in the simulation since no risk of aliasing existed.

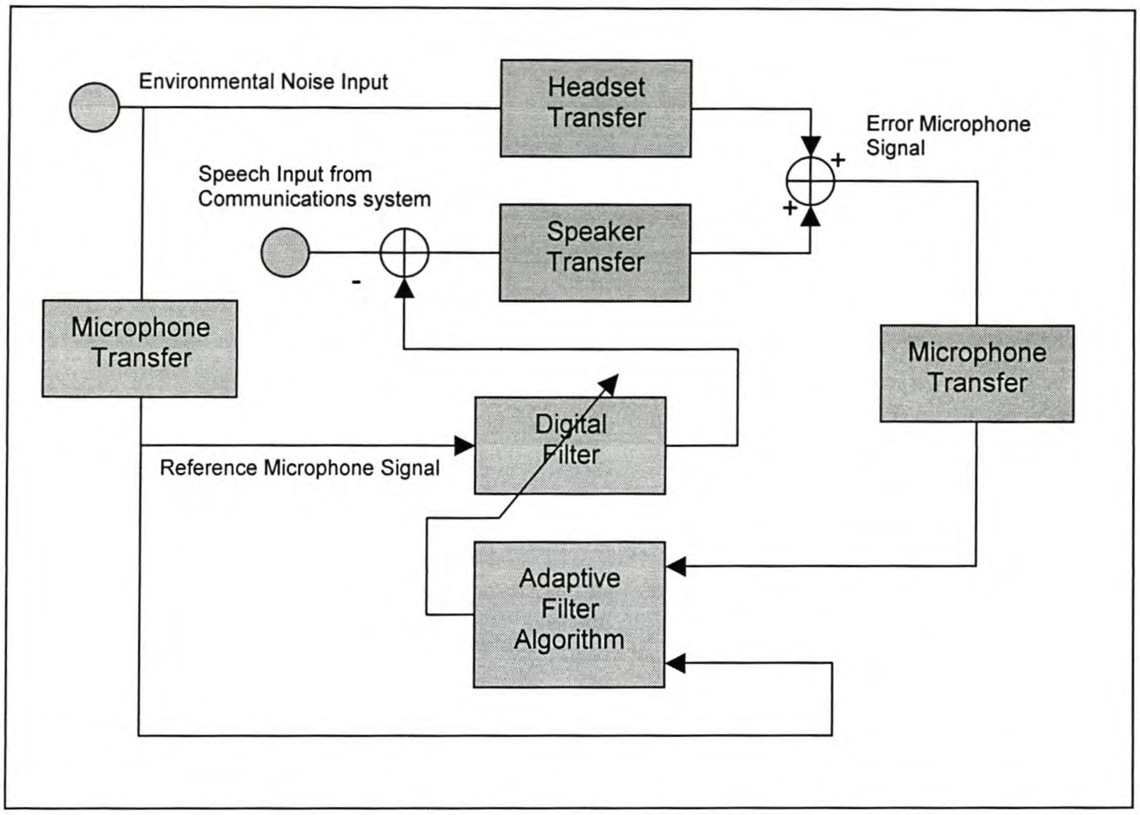

Environmental Noise Input
Headset Transfer
Error Microphone Signal
+
Speech Input from Communications system
+
Speaker Transfer
-
Microphone Transfer
Microphone Transfer
Digital Filter
Reference Microphone Signal
Adaptive Filter Algorithm

Figure 2.2 Block diagram for ANR simulation model

2.2.1. Headset transfer

The headset transfer can vary due to changes of air pressure inside the headset cavity. As mentioned before these changes are due to movement of the cushions that seal the headset to the side of the head of the user. Omni-directional sounds propagate from different angles through the headset to the headset cavity. It is expected that every angle have a slightly different transfer property.

From the above it becomes clear that the headset transfer is expected to be non-stationary when in practical use. To model such a non-stationary transfer function requires the implementation of a real time varying digital filter.

It was decided that the modeling of these non-stationary changes where outside the scope of this thesis. The headset transfer model implemented is thus only applicable for a stationary headset with an orientation where the headset is directed to the noise source.

The stationary headset transfer model was derived as follow.

2.2.1.1. Headset transfer measurement

Two audio recordings were made with a measurement set-up as shown in Figure 2.3. More detail about the test set-up can be found in [5].

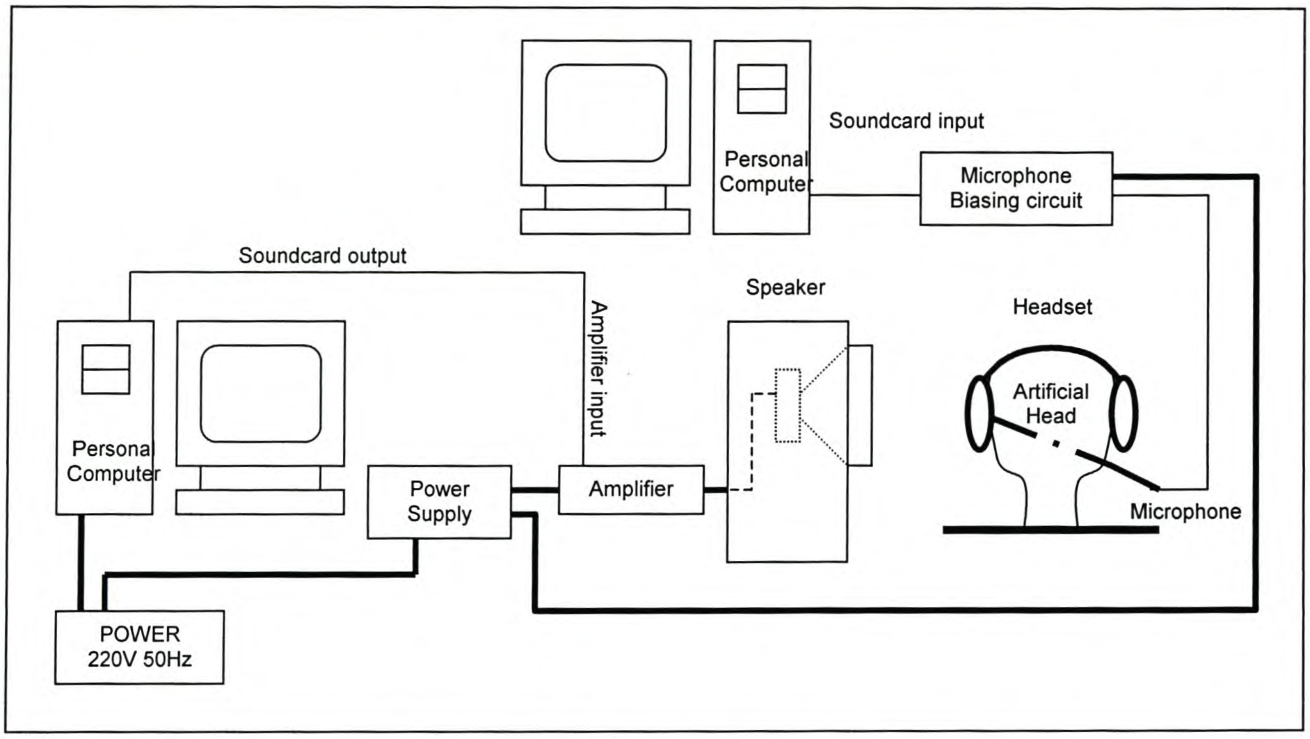


Figure 2.3 Test set-up constructed

A uniform distributed white noise audio wave file was generated with a Matlab routine. This file was transmitted over the loudspeaker of Figure 2.3 via the computer soundcard and an audio amplifier.

The direction of the noise source was kept fixed with relation to the artificial head to ensure that sound only propagated from one direction and no headset movement was allowed during recordings. This was done to ensure that the recordings conform to the previously mentioned headset model specifications

Two recordings were made. The first recording was made with no headset present. This noise recording served as a reference for the headset transfer measurement. The second recording was made with the headset placed on the artificial head over the microphone.

The power density spectrum for the 'transmitted' noise (recording with no headset on artificial head) and 'received' noise (recording with headset on artificial head) was calculated over the recorded frequency range.

Relating the 'received' power density spectrum to the 'transmitted' power density spectrum, as shown in equation 2.4, generated a logarithmic scale of the headset transfer.

$$Transfer = 10\log\left(\frac{Received\ Power\ Density\ Spectrum}{Transmitted\ Power\ Density\ Spectrum}\right) \qquad 2.4$$

The power density spectrum of equation 2.4 was averaged by the use of a Bartlett estimation method [12]. Figure 2.4 shows the resulting measured headset transfer at a 12 kHz sampling rate.

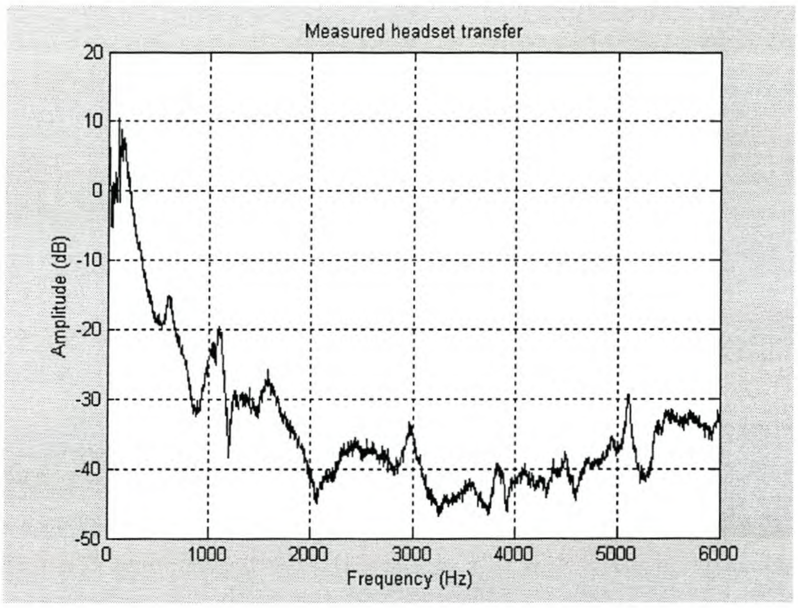


Figure 2.4 Measured headset transfer

2.2.1.2. Headset simulation filter estimation

Different system identification methods exist to determine unknown transfer functions for electronic systems.

The Least Squares (LS) method [13], [14] is one of the most commonly used transfer function estimation techniques and was used to determine different z-transforms for the previously mentioned audio recordings.

The Least Squares method is most understandably explained in the words of Gauss who stated that:

"According to this principle, the unknown parameters of a mathematical model should be chosen in such a way that the sum of the squares of the differences between the actually observed and computed values, multiplied by numbers that measure the degree of precision, is a minimum." [13]. The least squares method is directly derived from equation 2.3, as shown in equation 2.5.

$$\mathbf{w}(n)=\left[\mathbf{R}(n)\right]^{-1}\mathbf{Q}(n) \tag{2.5}$$

It must be noted that the LS method requires an input filter measurement and an output filter measurement to compute the related filter tap-weights. These measurements must be completed before any calculations can be made. The least square method does thus have some limitations. They are outlined as follow:

- The least squares method cannot estimate system parameters for non-stationary systems since measurements must be completed before any filter tap-weight calculations can be preformed.
- The least squares method cannot estimate system parameters in real-time since tap-weights can only be derived from completed measurements.
- For an auto regressive mean average (ARMA) filter, the estimation runs the risk of being unstable.
- If the order of the system to be estimated is unknown, it must be chosen by trial and error.

The previously recorded audio recordings, mentioned in section 2.2.1.1, were taken to derive a stationary headset transfer function. The LS limitation on stationaraty will thus not play any role in the estimation of a transfer function for the headset.

Real time issues were of no concern since the audio recordings were analysed after recording and could even allow non-causal analysis.

Establishing filter stability would not be a problem since the estimated filter frequency response could be evaluated according to the measured headset transfer found in section 2.2.1.1.

The only issues under consideration for the least squares estimation would be the choice of the system order.

In 1962 Shaw and Thiessen [15] showed that the passive attenuation of a headset could be modelled as a second order function consisting of the shell mass (M), cushion damping (R), stiffness of air in the ear peace cavity (K_v) and stiffness of air in the cushion (K_c). This function is shown in equation 2.6.

$$Attenuation = \frac{\frac{K_v}{M}}{s^2 + s\frac{R}{M} + \frac{K_v + K_c}{M}} \qquad 2.6$$

According to equation 2.6 it can be expected that the theoretical headset transfer will consist of at least 2 poles.

The LS estimation method was implemented in Matlab. To confirm that the headset attenuation preformed as expected by Shaw and Thiessen [15], [16], a 2 pole and no zero filter was estimated from the recorded waves. The resulting transfer function was compared to the measured headset response. This LS estimated filter response is shown in Figure 2.5.

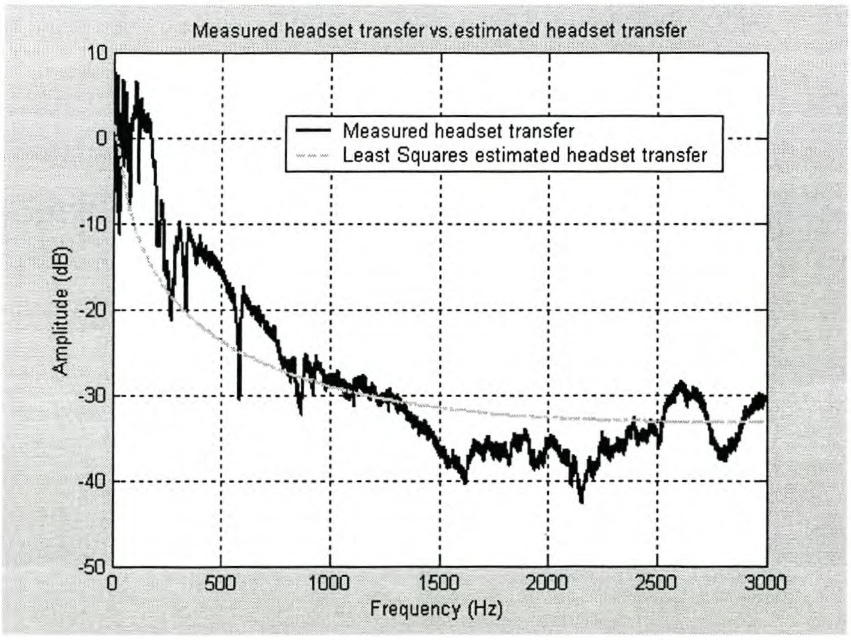


Figure 2.5 Two pole, no zero LS estimated headset transfer

The resulting z-transform filter was found to be,

$$Attenuation = \frac{-0.03724}{z^2 - 0.841z - 0.1278} \qquad 2.7$$

To improve the simulation model to reflect better to the measured response, it was decided to investigate higher order estimated filters. The choice of higher order filters provided the estimated filters to resemble the measured response more accurately. To describe why higher order filters could improve the transfer representation, it should first be shown how a filter output is constructed.

When an ARMA filter is used, as is used in this case, each filter output is constructed from past input and output values of the filter in question. This is shown below.

Z-transform filter $$\frac{Out(z)}{In(z)} = \frac{z+a}{z+b}$$

Time domain representation $$Out(n) = In(n) + aIn(n-1) - bOut(n)$$

Poles thus resemble constants that weigh past output values to contribute to a new filter output and zeros resemble constants that weigh past input values that contribute to a new filter output.

The best pole/zero relation was found for a filter relation where the headset transfer filter consisted of more zeros (past input values) but a limited amount of poles (fed back output values).

The amount of zeros chosen could be high since the headset transfer is related to time delayed noise the seeps in through the earpiece seal. A limited amount of poles could be chosen before the estimated filter became unstable. This instability is due to the feedback of to many past output values. The limited amount of poles to be chosen thus constitutes that only a limited amount of poles exists for the headset transfer.

The LS estimated filter that best resembled the measured headset transfer was an ARMA filter with 3 poles and 40 zeros. The resulting transfer is shown in Figure 2.6.

The headset transfer model established will allow the feedforward simulation model to simulate the headset attenuation of external noise to the inside of the headset cavity on a sample for sample basis. This is essential since the adaptive filter algorithm will adapt dynamically on a sample for sample basis.

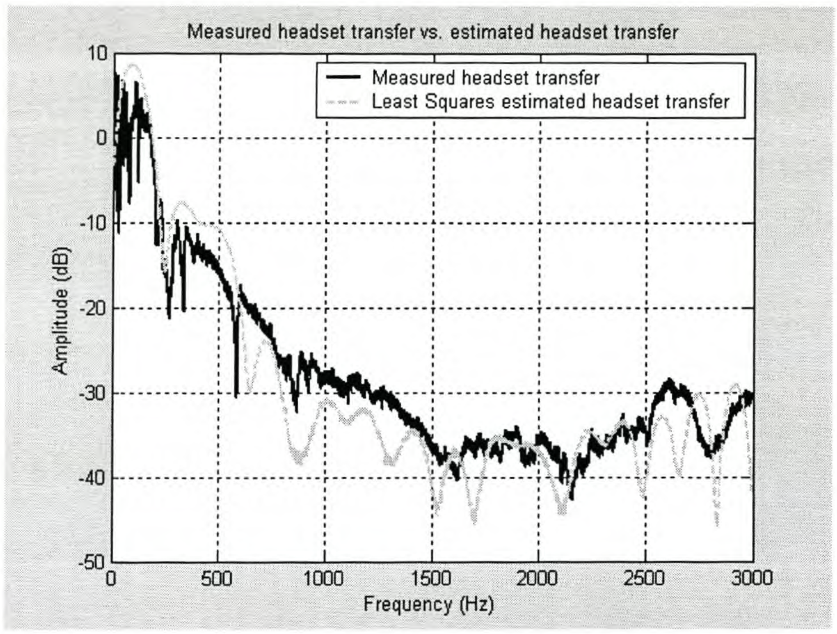


Figure 2.6 Three pole, forty zero LS estimated headset transfer

2.2.2. Loudspeaker transfer

A digital filter could also model the loudspeaker characteristics. The transfer characteristics for military specification headphone loudspeakers were taken from the specifications given in [17]. The specification were found to be

Type:	Dynamic moving coil
Diaphragm Material:	Water resistant mylar
Impedance:	30 Ω
Frequency Response:	20 to 20 000 Hz

The sample rate of the ANR system would be chosen is such a way to provide a operational bandwidth much lower than 20kHz and it was for this reason that the loudspeaker filter was chosen as a high pass filter.

The loudspeaker filter was chosen as a first order Butterworth high pass filter. The filter frequency response is shown in Figure 2.7. The filter transfer function is shown in equation 2.8.

$$F_{speaker} = \frac{0.975(z-1)}{z-0.949} \qquad 2.8$$

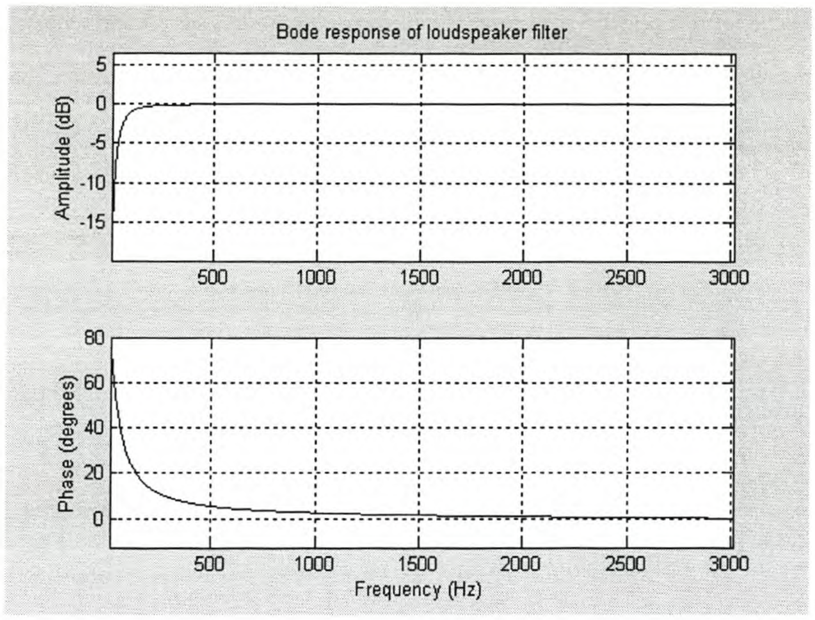


Figure 2.7 Frequency response of loudspeaker high pass filter

The filter was implemented as a causal difference equation to enable real time simulation.

2.2.3. Microphone transfer

A digital filter could similarly model the sense microphone behaviour. The transfer characteristics for military specification microphones were once again taken from the specifications given in [17]. The specifications taken from [17] was,

Type:	Electret, with integral FET pre-amplifier
Output Impedance:	2500 Ω
Frequency Response:	20 to 15 000 Hz
Features:	Rugged construction to withstand a wide range of temperatures and humidity conditions

As before the microphone response was also approximated to a high pass filter since the system operational bandwidth was expected to be lower than the low pass microphone cut-off.

A first order Butterworth high pass filter configuration was chosen to simulate the microphone response. The filter transfer function is similar to that of the loudspeaker and is shown once again in equation 2.9.

$$F_{speaker} = \frac{0.975(z-1)}{z-0.949} \qquad 2.9$$

The frequency response of the microphone simulation transfer function is shown in Figure 2.8.

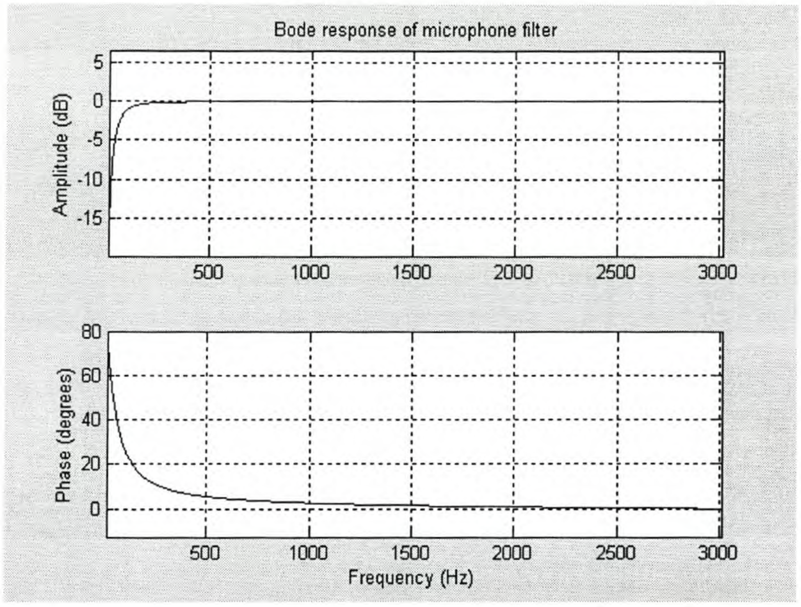


Figure 2.8 Frequency response of microphone filter

The filter was also implemented as a causal difference equation to enable real time simulation.

2.2.4. Adaptive filter

Any adaptive filter algorithm could be implemented in the ANR simulation model but only a number of adaptive filter characteristics were found applicable to noise cancellation in this topology.

Filters are very versatile and can be used in a number of applications. The applicable filter characteristics for filtering aircraft noise from a headset can be outlined as follow.

- The filter must be adaptive since the acoustic headset transfer is different for every user of a headset. This is due to different head and ear dimensions of users. The filter must also be adaptive to changes that occur in the headset transfer due to changes on the earpiece seal when a user moves his head.

- To ensure that all adaptive filters to be estimated are stable, the adaptive filter topology will only consider estimating finite-duration impulse response (FIR) filters since they are always guaranteed to be stable.

- The transfer of the headset can be approximated as being linear. The adaptive filters to be used must thus be able to estimate linear transfer functions.

- Since most adaptive filters are based on second order statistics (Wide sense stationary), it must be possible to acquire second order statistics, such as the mean, correlation and cross-correlation of the outside versus inside noise to the headset.

The adaptive filters that adhere to the above specifications and that will be investigated are,

The method of Steepest Descent (LMS algorithm)
Recursive Least Squares Algorithm (RLS algorithm)
Kaczmarz's Projection Algorithm (Normalised LMS algorithm)
Kalman filter

It was found that the Kalman filter reduces to the RLS algorithm in a simplified form, it was thus deemed unnecessary to investigate both the Kalman filter and RLS algorithm. Only the following adaptive filters were investigated.

The method of Steepest Descent (LMS algorithm)
Recursive Least Squares Algorithm (RLS algorithm)
Kaczmarz's Projection Algorithm (Normalised LMS algorithm)

2.2.5. Sampling rate

The choice of an adequate simulation sampling rate was made according to the bandwidth performance required for the ANR system and the physical set-up of the ANR system.

The bandwidth specified for ANR operation was set at 3kHz. This bandwidth was chosen since the military speech band is 300 – 3000Hz and would thus be included in the noise reduction bandwidth.

ANR of this bandwidth would improve the audibility of the communications speech in a headset fitted with this system.

From previous literature, [2], [7] it must however be noted that the bandwidth chosen here is exceptionally broadband with respect to ANR systems currently in use.

Keeping the Nyquist criterion in mind, the minimum sampling rate must be twice the bandwidth needed. Thus the minimum simulation sampling rate was set at 6kHz.

Sample rate microphone positioning implications

When the ANR system is practically implemented the reference and error microphone will never be able to sense the same sound wave at precisely the same time. This is due to the fact that the microphones will always be a distance (ℓ) apart. This can be illustrated as shown in Figure 2.9.

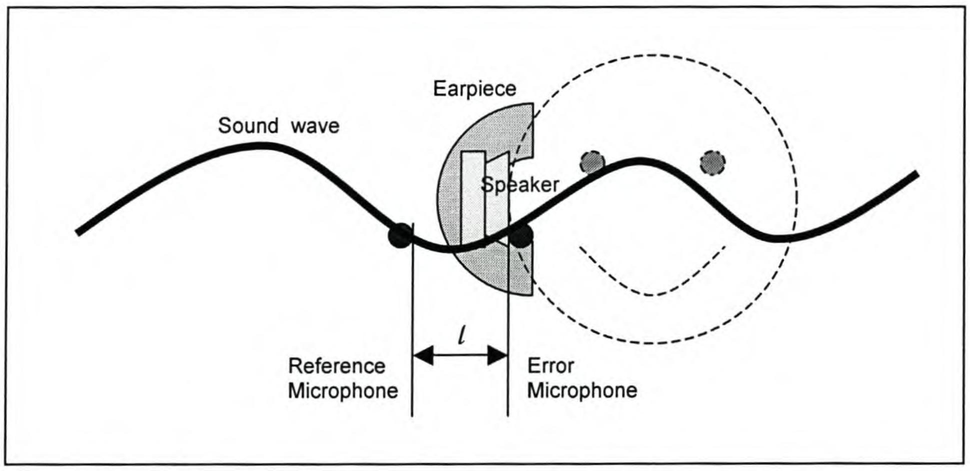


Figure 2.9 Microphone placement with relation to the system sample rate

For optimal adaptive filter operation, the error and reference signals used by the adaptive filter must be of the same time instant on the same propagating sound wave. To solve this acoustic delay problem, the sample rate of the system can be adapted to reflect the distance between the microphones.

This distance between the microphones is related to the speed of sound and the specific sample rate as shown in equation 2.10.

$$c_{sound} = \frac{\ell}{T_s} = f_s \ell \qquad 2.10$$

c is the speed of sound (330 m/s), f_s the system sampling rate and ℓ is the distance between the microphones. The distance between the microphones was measured to be 30mm.

Thus f_s computes to: $$f_s = \frac{c_{sound}}{\ell} = \frac{330}{0.03} = 11kHz$$

If the reference microphone samples are delayed for one sample period before being related with the current sampled error microphone sample, no acoustic time laps will be experienced for the adaptive filter calculations. This sampled delay compensation will be referred to as the algorithm update delay. More about this delay in section 8.2.2.

Since simulation for high sample rates are very time consuming, it was decided to run simulations at 6kHz. This implies that the distance between the microphones of the simulated system is approximately 55mm. This distance is related to the 6kHz sample rate by equation 2.10.

It was thus found that the acoustic time delay for sound to travel from one microphone to the other can be electrically compensated for by adjusting the sampling rate and an number of sample delays on the first microphone signal to bring it in phase with the second microphone signal. This compensation is only applicable to updating the adaptive filter algorithm and not the actual filtering for the reference noise to resemble the inside noise.

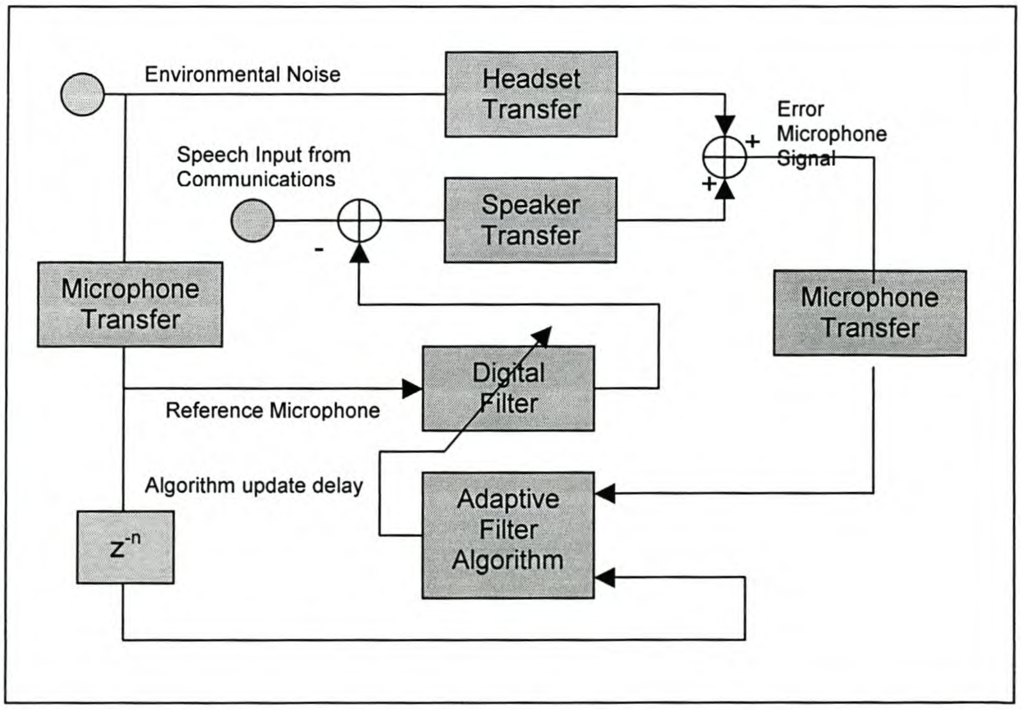


Figure 2.10 Time delayed reference signal adaptive filter updating to compensate for acoustic delay between microphones

Solving this phenomenon was found essential to ensure broadband noise cancellation. Figure 2.10 shows the adapted updating topology.

2.3. Z-transform model of simulation

To understand the simulation operation of the ANR system better, the block diagram of Figure 2.2 was transformed to a time domain block diagram, Figure 2.11. This time domain block diagram was then transformed to a z-parameter block diagram. This block diagram is shown in Figure 2.12.

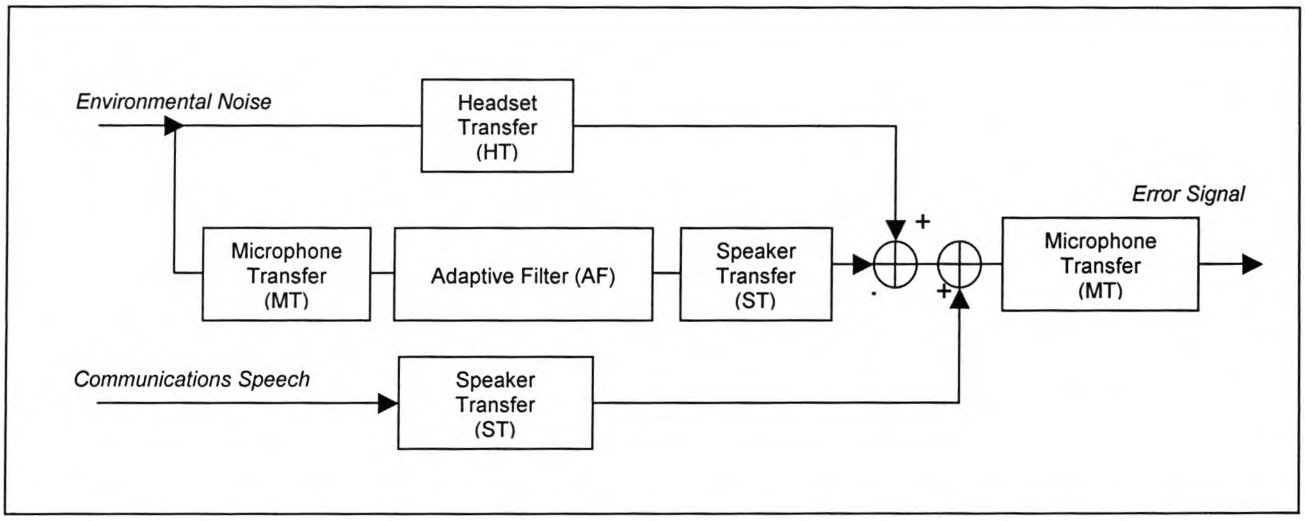


Figure 2.11 Time domain of ANR simulation operation

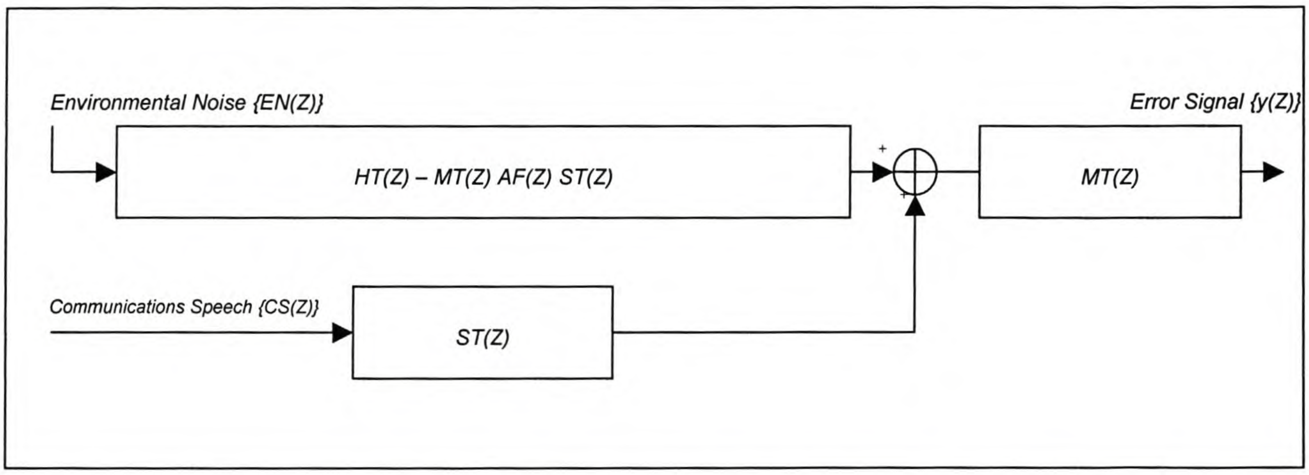

Environmental Noise {EN(Z)}
Error Signal {y(Z)}
HT(Z) – MT(Z) AF(Z) ST(Z)
+
+
MT(Z)
Communications Speech {CS(Z)}
ST(Z)

Figure 2.12 Z-domain of ANR simulation operation

It must be noted that the use of the adaptive filter error signal path was removed since the adaptive filter update calculations cannot be modelled as a transfer function. The adaptive filter is assumed to be a stationary filter for every sampling period and that the filter coefficients are updated before every filter sample period.

The system transfer can thus be described as shown in equation 2.11.

$$MT(Z)\left[EN(Z)HT(Z)-EN(Z)MT(Z)AF(Z)ST(Z)+CS(Z)ST(Z)\right]=Y(Z) \quad 2.11$$

To understand the basic operation of the system, equation 2.11 was simplified with the following assumptions.

$$ST(Z)=MT(Z)=1$$

These assumptions holds if the loudspeaker and microphones are ideal and have unity transfer. For simplicity it is also assumed that the system operates without any communications speech, thus $CS(Z)$ = 0. Then equation 2.11 changes to equation 2.12.

$$EN(Z)HT(Z)-EN(Z)AF(Z)=Y(Z) \quad 2.12$$

In the time domain, the relation of equation 2.12 can be seen as follows,

$$x(n)-\hat{x}(n)=y(n) \quad 2.13$$

Equation 2.13 can intuitively be explained as headset filtered environmental noise ($x(n)$) that is cancelled by subtracting anti-noise created by digitally filtering the reference environmental noise ($\hat{x}(n)$).

$y(n)$ is the error output after noise cancelling. The adaptive filter uses the $y(n)$ output as a error indication for the adaptive filter and so updates the filter coefficients to minimise $y(n)$.

2.4. Evaluation of adaptive filter algorithms using the simulation model

The simulation model was implemented in Matlab. The program code for a basic simulation can be seen in Appendix C. The model components were individually designed and tested as shown in section 2.2.

Al tested components were combined in the simulation. The purpose of the simulation was to gather adequate information to evaluate the different adaptive filter algorithms.

The performance of each adaptive filter algorithm will be evaluated to a set of criteria that will ensure the adaptive filter is optimally set-up. This is essential to establish the optimal solution between the different adaptive filter possibilities.

The criteria for evaluating adaptive filters for optimality is outlined as follow.

- <u>Algorithm parameters</u>

  Almost every adaptive filter algorithms has individual characteristic parameters that can be adjusted to optimise the performance of the adaptive filter.

  These parameters are unique to each algorithm and their influence on the operation of the adaptive filter will be explained in the chapters concerned with each particular filter algorithm.

- <u>Added communications speech</u>

  Communications speech interferes with the feed forward error signal to the adaptive filter. This communication signal was also added to the simulation environment as shown in Figure 2.2.

  This was done to ascertain the effects this will have on the adaptive filter performance.

- <u>Filter topology and tap size of adaptive filter</u>

  The ANR simulation was limited to estimating FIR filters of a variable tap size. These tap sizes were varied to find the optimal tap size for the estimation filter.

- <u>Bandwidth attenuation performance (No communication vs. communication)</u>

  This parameter investigates the attenuation achieved by the system over the operational bandwidth of the system. The system performance for when a communications signal is present versus when no communications signal is present was also investigated using this measurement.

- <u>Convergence considerations</u>

The convergence of each algorithm will be evaluated by summing all the tap weight values of the estimated filters into one variable. This variable will be known as the convergence variable. The significance of the convergence variable is that it resembles the step response of the filter.

For example:

If the FIR adaptive filter difference equation is given as

$$y(n) = Ax(n) + Bx(n-1) + Cx(n-2) = \begin{bmatrix} x(n) & x(n-1) & x(n-2) \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix}$$

The convergence variable for time n will be the sum of the filter tap weights.

$$\mathit{Convergence\ variable(n)} = A + B + C$$

This is the same as taking the unit step response for the FIR difference equation as can be shown for the calculation of the 5th sample output of the chosen FIR filter. For a unit step response $x(n\text{-}k)=1$ for all $n>1$ and $k>1$ except where $k=n$.

$$y(5) = Ax(5) + Bx(4) + Cx(3) = A + B + C$$

Changes in the tap weight vector will thus be directly reflected in one single variable over time. This single variable will be used as a guide to evaluate the convergence of an adaptive filter algorithm.

The influence of an added communication signal on the convergence of the adaptive filter can also be investigated using this variable.

- <u>Microphone and Loudspeaker transfer influences</u>

The inclusion of loudspeaker and microphone transfer models in the ANR simulation will cause phase differences between the headset attenuated aircraft noise and the adaptive filter generated anti-noise.

The effects of adding these filters will also be investigated

2.5. Active noise reduction simulation model conclusion

In conclusion this chapter gave the outline for the ANR configuration that will be used to evaluate different adaptive filters in simulation. This basic configuration will enable an equal standard to weigh different adaptive filter methods as an ANR solution.

It is now possible to start an investigation in relation to the possible adaptive filters for this ANR system. The following chapters are dedicated to investigate different adaptive filters and their performance in the simulated ANR environment.

Chapter 3

3 Method of Steepest Descent (LMS algorithm)

Chapter 3 is part of the investigation on adaptive filters as a solution to active noise reduction in aircraft headsets.

The adaptive filter method of steepest decent will be addressed in this chapter and the performance of this method will be established in the simulated environment according to specifications set out in chapter 2.

3.1. Background

The basic idea of steepest descent is based on the assumption that a continuously differentiable cost function, $J(\mathbf{w})$, with optimal solution, $\mathbf{w}_o$, exists. $J(\mathbf{w}_o)$ will be optimal if it is the smallest attainable value for $J(\mathbf{w})$. This cost function must thus comply with the inequality of equation 3.1, for all $\mathbf{w}$.

$$J(\mathbf{w}_o) \leq J(\mathbf{w}) \qquad 3.1$$

If a initial $\mathbf{w}(0)$ is chosen, a sequence of $\mathbf{w}(n)$ values must be found to improve $J(\mathbf{w}(n))$ with every iteration. This can be illustrated with equation 3.2.

$$J(\mathbf{w}(n+1)) < J(\mathbf{w}(n)) \qquad 3.2$$

The steepest descent algorithm is based on the methodology that $\mathbf{w}(n+1)$ can be updated along the gradient of the cost function, $J(\mathbf{w}(n))$, towards it's optimum value for as long as equation 3.1 and 3.2 holds.

With this in mind the method of steepest descent can be formalised by equation 3.3. This will ensure that a more optimised cost function is found for every iteration of the algorithm.

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{\mu}{2}\left[\nabla \mathbf{J}(n)\right] \qquad 3.3$$

μ controls the size of the incremental correction applied to $\mathbf{w}(n)$ and is referred to as the step size or weighing constant.

An adaptation of the steepest descent method gives rise to the least-mean square algorithm and is considered a standard at which all adaptive filters can be benchmarked. This is due to the simplicity of the algorithm. [8], [11].

3.2. Algorithm outline and operation

To derive the least mean squares algorithm from the steepest descent method, a cost function must be found that complies with the restrictions of the steepest descent method.

A quadratic cost function, as shown in equation 3.4, is chosen since it is continuously differentiable and complies with equation 3.1 and 3.2.

$$J(\mathbf{w},t) = E\left[d(n) - \mathbf{w}^T(n)\mathbf{u}(n)\right]^2 \qquad 3.4$$

$d(n)$ is the desired output of the adaptive filter. $\mathbf{u}(n)$ is a observation matrix comprising of past inputs to the adaptive filter as far back as the order of the filter to be estimated and $\mathbf{w}(n)$ is the filter tap weights to be estimated.

$$\mathbf{u}(n) = [x(n) \quad x(n-1) \quad x(n-2) \quad ..... \quad x(n-order)]$$
$$\mathbf{w}(n) = [w_1(n) \quad w_2(n) \quad w_3(n) \quad ..... \quad w_{order}(n)]$$

The optimal solution to equation 3.4 will be the solution with the smallest value, thus the solution with least squares, hence the name least mean squares algorithm. If equation 3.4 is expanded into its canonical form it can be seen as shown in equation 3.5.

$$J(n) = \sigma_d^2 - \mathbf{w}^T(n)\mathbf{Q}(n) - \mathbf{Q}^T(n)\mathbf{w}(n) + \mathbf{w}^T(n)\mathbf{R}(n)\mathbf{w}(n) \qquad 3.5$$

Where σ_d^2 is the variance of a desired signal $d(n)$.
$\mathbf{Q}(n)$ is the cross-correlation vector between $\mathbf{u}(n)$ and $d(n)$, E[$\mathbf{u}(n)d(n)$], and
$\mathbf{R}(n)$ is the correlation matrix of the input observation vector $\mathbf{u}(n)$, E[$\mathbf{u}(n)\mathbf{u}^T(n)$].

The least mean squares algorithm is based on the principal of calculating the gradient of the cost function and recursively optimising the tap weight vector $\mathbf{w}(n)$ to ensure the cost function converges to a minimum value. The surface of the cost function can be described as the error-performance surface.

Since the cost function is quadratic, the error-performance surface can be visualised as a bowl-shaped surface with the minimum point at the bottom of the bowl.

The steepest decent algorithm will thus aim to calculate the slope or gradient of the cost function and adapting the filter tap weights will be done so that the cost function gradient converges to the bottom of the error-performance surface.

The slope of the cost function is calculated as the derivative of equation 3.5. The result is shown in equation 3.6.

$$\frac{\partial J(n)}{\partial \mathbf{w}(n)} = \nabla \mathbf{J}(n) = -2\mathbf{Q}(n) + 2\mathbf{R}(n)\mathbf{w}(n) \qquad 3.6$$

From equation 3.6 it can be seen that the optimum solution for the cost function (bottom of the bowl) will be the Weiner-Hopf equation as shown in equation 3.7.

$$\mathbf{R}(n)\mathbf{w}_o = \mathbf{Q}(n) \qquad 3.7$$

Where $\mathbf{w}_o$ can be seen as the optimum tap weight vector or Weiner solution.

If equation 3.7 is substituted into equation 3.5, the minimum cost function solution is found as shown in 3.8.

$$J_{\min} = \sigma_d^2 - \mathbf{Q}^T \mathbf{w}_o \qquad 3.8$$

This minimum ($J_{\min}$) can be seen as the lowest point on the quadratic cost function. The least mean square algorithm will tend to approach this minimum value.

The error-performance surface gradient of equation 3.6 can thus be substituted into the steepest descent tap update algorithm of equation 3.3. The tap weights are thus adapted along the negative direction of the error-performance surface slope, towards $J_{\min}$, by equation 3.9.

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) - \frac{\mu}{2}[\nabla \mathbf{J}(n)] \\ \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu[\mathbf{Q}(n) - \mathbf{R}(n)\mathbf{w}(n)]\end{aligned} \qquad 3.9$$

In order to ensure stability for the least mean square algorithm, μ must be bigger than zero but smaller than 2 over the largest eigenvalue (λ_{Max}) of the correlation matrix $\mathbf{R}$. The derivation of this boundary condition can be found in [8], [11].

To compute $\mathbf{Q}(n)$-$\mathbf{R}(n)\mathbf{w}(n)$ however is not so simple if $\mathbf{Q}(n)$ and $\mathbf{R}(n)$ is not known. [8], [11] gives another approach to $\mathbf{Q}(n)$-$\mathbf{R}(n)\mathbf{w}(n)$ as shown in the derivation of 3.10. In this derivation it is accepted that all signals are only real and no imaginary parts are present.

$$\begin{aligned}\mathbf{Q}(n) - \mathbf{R}(n)\mathbf{w}(n) &= E[\mathbf{u}(n)d(n)] - E[\mathbf{u}(n)\mathbf{u}^T(n)]\mathbf{w}(n) \\ &= E[\mathbf{u}(n)d(n) - \mathbf{u}(n)\mathbf{u}^T(n)\mathbf{w}(n)] \\ &= E[\mathbf{u}(n)(d(n) - \mathbf{u}^T(n)\mathbf{w}(n))] \\ &= E[\mathbf{u}(n)(d(n) - \mathbf{w}^T(n)\mathbf{u}(n))] \\ &= E[\mathbf{u}(n)e(n)]\end{aligned} \qquad 3.10$$

The estimation error $e(n)$ is very easily calculated from the desired reference signal and the estimated filter output, as shown in equation 3.11. The least mean square algorithm now come down to implementing equation 3.11 and equation 3.12.

$$e(n) = d(n) - \mathbf{w}^T(n)\mathbf{u}(n) \qquad 3.11$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu E[\mathbf{u}(n)e(n)] \qquad 3.12$$

This version of the least mean square algorithm is however not very well suited for non-stationary processes since the tap-weights are estimated along a deterministic trajectory of the error-performance surface.

If this least mean square algorithm is allowed to determine the tap-weights with some element of converging random motion around an error-performance surface, it will be easier to estimate non-stationary processes.

This converging random motion is possible if two types of convergence can be achieved, namely convergence in the mean, which implies

$$E\left[\mathbf{w}(n)\right] \rightarrow \mathbf{w}_o \quad as \quad n \rightarrow \infty \qquad 3.13$$

And convergence in the mean square, which implies

$$J(n) \rightarrow J(\infty) \quad as \quad n \rightarrow \infty \qquad 3.14$$

Where $\mathbf{w}_o$ and $J(\infty)$ implies that the Weiner-Hopf solution has been found.

Both these convergences are possible with a random motion convergence if $\mathbf{Q}(n)$-$\mathbf{R}(n)\mathbf{w}(n)$ is approximated by $\mathbf{u}(n)e(n)$. This simplifies the previous algorithm to give the final LMS algorithm, as shown in equation 3.15 and 3.16.

$$e(n) = d(n) - \mathbf{w}^T(n)\mathbf{u}(n) \qquad 3.15$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)e(n) \qquad 3.16$$

This LMS algorithm derivation was derived in accordance to [8] and [11].

3.3. <u>Algorithm performance</u>

The algorithm performance of the LMS algorithm was established according to the specifications set in section 2.4 of chapter 2 and the simulation model of chapter 2.

3.3.1. Algorithm parameters

From equation 3.16, it can be seen that the only variable parameter in the LMS algorithm will be the step size variable (μ).

Optimising the algorithm parameter (μ) for the LMS algorithm in the headset topology was addressed in three steps.

- Step one was to establish the boundary $2/\lambda_{Max}$.

- Step two was to establish the optimum step size parameter for when only noise is present in the ANR system.

- Step three was to establish the optimum step size parameter for when noise and a communication signal are present in the ANR system.

<u>Step one</u>

The LMS algorithm was simulated for a stationary headset transfer, 6 kHz sampling rate and no added communications speech to determine the behaviour and optimal value for the step size variable.

From [8], [11] it is known that the step size variable may not exceed the boundaries of $0<\mu<2/\lambda_{Max}$. This first simulation test was done to establish where the $2/\lambda_{Max}$ boundary is situated.

Figure 3.1 shows a plot of the cost function values for different step sizes (μ). The cost function values were derived from the square of the simulated error microphone signal. The simulated error signal data for second 2 to 3 of a 3 second simulation was used to calculate the average cost function values of Figure 3.1.

From Figure 3.1 it can be seen that the $2/\lambda_{Max}$ step size boundary can be approximated to be at 0.009. It can be seen how the LMS performance deteriorates exponentially towards this boundary.

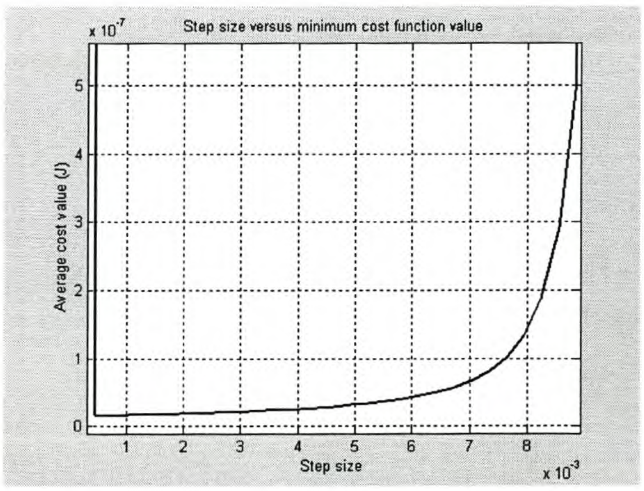


Figure 3.1 Step size versus cost function value

Figure 3.1, for small step size values, show a deterioration of performance since the cost function values increase drastically for step sizes lower than 0.00045. This deterioration is due to the convergence time boundary of 3 seconds set for the simulation. If the simulation were allowed an infinite convergence time, the algorithm would be allowed to converge to an optimal solution with even lower cost function values than shown in Figure 3.1.

The convergence time constraint can also be explained by Figure 3.2. Figure 3.2 shows the convergence variable mentioned in section 2.4 of chapter 2 plotted over the simulation run time.

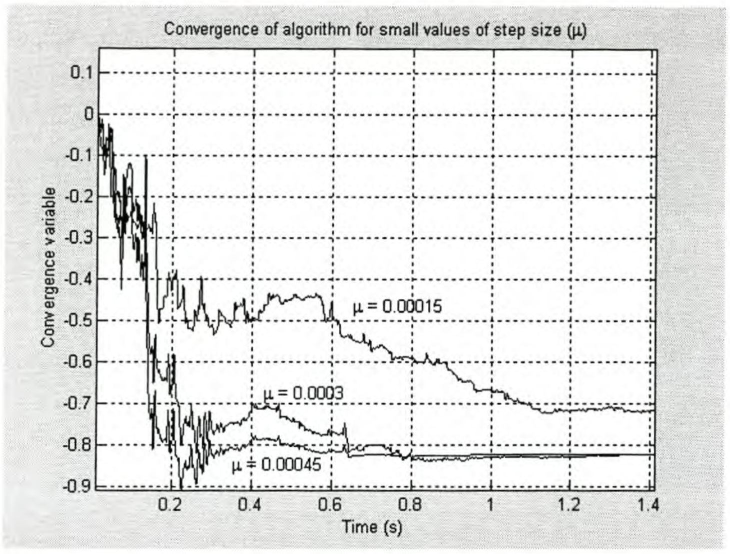


Figure 3.2 Slow convergence for small step size values

Figure 3.2 shows how small step size values take longer to converge to the optimal estimated filter. If the convergence time is limited, the algorithm will not have converged to the optimal filter thus resulting in a big minimum squared error or cost function (J) value. This thus explains the sharp increase in cost function values experienced for step size values smaller than 0.00045.

From Figure 3.1 it can thus be seen that step size values smaller than 0.00045 constitutes LMS algorithm convergence times longer than 2 seconds.

It is thus recommended that the bad cost function performance for step size values smaller than 0.00045 be ignored with the understanding that step size value smaller than 0.00045 converge slower than was required by the LMS algorithm in simulation.

Step two

The experimental simulation run for Figure 3.1 was done with only a noise signal present. Figure 3.1 is thus applicable to step two as well.

It is clear that the optimal step size value for the LMS algorithm, when only noise and no speech signal is considered, will be the lowest value on the curve of Figure 3.1.

If a closer look is taken to Figure 3.1 it can be seen that the optimal step size value can be chosen at 0.0005. This will ensure an algorithm convergence of less than 2 seconds with the best estimated fit to the transfer of the given headset. A zoomed version of Figure 3.1 is shown in Figure 3.3.

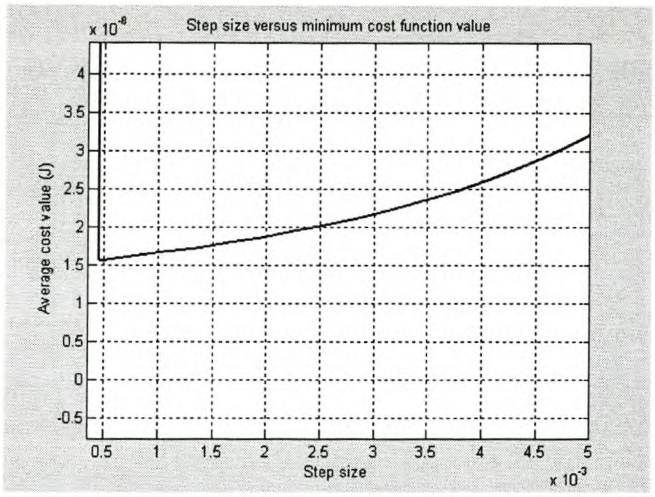


Figure 3.3 Zoom step size versus cost function value

Step three

When speech is added to the system, the minimum cost function value attainable rises considerably, as can be seen in the cost function values of Figure 3.4.

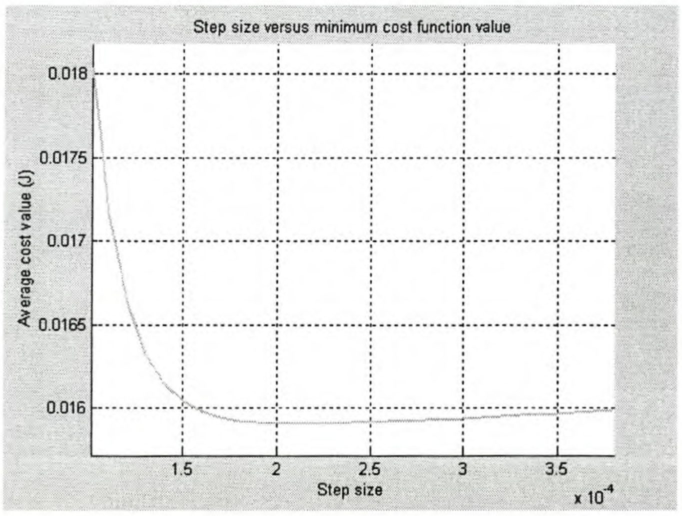


Figure 3.4 Step size versus cost function values with a communications signal

The optimum step size with an added communication signal was found to be 0.00021. The optimum step size value thus decreased due to a added communication signal.

This implies that an added communications signal allows faster convergence at the cost of finding a less accurate estimated filter. This is evident from the rise in the minimum cost function value. The value rose from 1.6×10^{-4} to approximately 1.6×10^{-2}.

3.3.2. Filter topology and tap size of adaptive filter

As stated before, only FIR filters will be estimated to ensure that all estimated filters are stable. It must be noted that it would be impossible for the implemented FIR adaptive filter algorithm to exactly estimate a filter to resemble the ARMA headset transfer since a FIR filter can not exactly resemble an ARMA filter. The minimum squared error can thus never be zero.

The effects of varying the filter order was investigated according to the average cost function value of every specific adaptive filter with specified filter order.

The simulations were preformed for a stationary headset, sample rate of 6kHz and step size chosen at 0.0005. The simulations were preformed twice, once with communications speech and once without communications speech.

It is shown in Figure 3.5 that an increase of the tap weight order improves the adaptive filter performance since the average cost function value approaches a minimum squared error value for high filter orders.

The minimum squared error approaches a small value close to zero when no communication signal is present but a larger constant minimum squared error is found for when a communications signal is present. This implies that the inclusion of a communications signal makes it more difficult for the LMS filter to estimate the given headset transfer.

A limiting factor in the choice of the number of taps can be the processing ability of the DSP chip to be used to implement the LMS algorithm.

The limiting factor of the DSP chip for algorithm calculation does however not have such far reaching effects with the LMS algorithm as apposed to the RLS algorithm that requires the calculation of a inverse correlation matrix update, $\mathbf{P}(n)$, but more about this in chapter 4.

The LMS algorithm was found the least calculation intensive of all considered algorithms. This allows the LMS algorithm to estimate higher orders of tap weights than other adaptive filters.

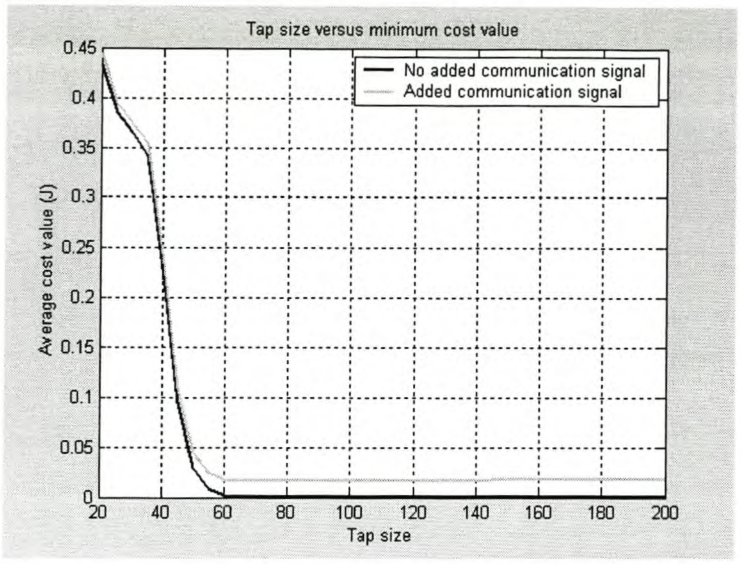


Figure 3.5 Average cost function value versus tap order of the adaptive filter

A reasonable tap size that will be adequate for DSP implementation and ensure maximum attenuation is chosen at 100 tap weights.

3.3.3. Bandwidth attenuation performance (No communication vs. communication)

This section investigated the LMS bandwidth attenuation properties for when a communications signal is present versus when no communications signal is present.

The frequency spectrum attenuation was also compared to the energy attenuation of the noise signals. This attenuation was calculated according to equation 3.17. The total attenuation of signal energy due to ANR was calculated from simulation-generated signals. $Y(n)$ is the residue noise inside the ear piece cavity after noise reduction and $X(n)$ is the noise inside the ear piece cavity with no reduction.

$$Attenuation = 10\log\left(\frac{\frac{1}{N}\sum_{n=1}^{N}X^2(n)}{\frac{1}{M}\sum_{n=1}^{M}Y^2(n)}\right) \qquad 3.17$$

The bandwidth attenuation performance for the two simulations is shown in Figure 3.6.

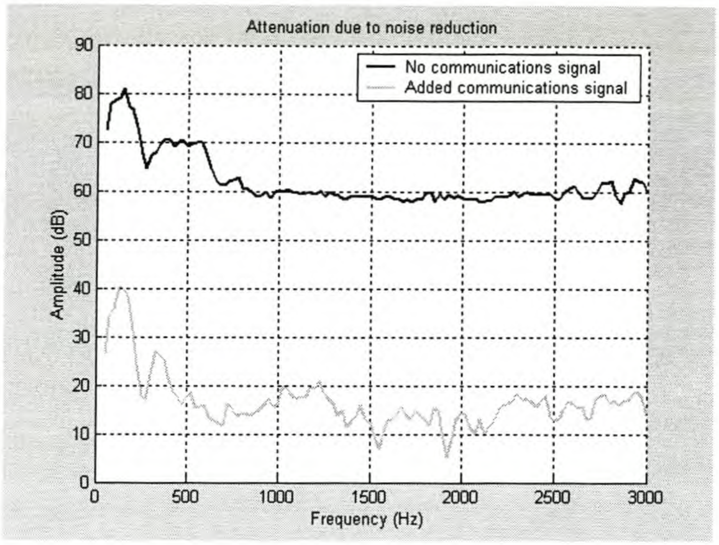


Figure 3.6 Bandwidth attenuation performance (Communication vs. No Communication)

From Figure 3.6 it was found that added communications signals drastically reduce the attenuation attained by the system. The total energy attenuation for the two cases was found to be 75.434dB for no communications signal and 28.467dB for an added communications signal.

3.3.4. Convergence considerations

The convergence of the LMS algorithm was evaluated according to the convergence variable as described in section 2.4 of chapter 2.

Considerations that were investigated were the convergence performance of the LMS algorithm for a system with communications signal as apposed to one without a communications signal.

The effect of very small step size values (μ) was investigated in section 3.3.1 and will not be addressed again in this section.

Convergence differences for a system with communications signal as apposed to one without a communications signal

The convergence variable behaviour for the two cases is shown in Figure 3.7. It was found that the convergence variable for case one (no communications signal) converges to a single optimal filter solution of mean value -0.6567 and no variance. This mean value resembles the filter step response output.

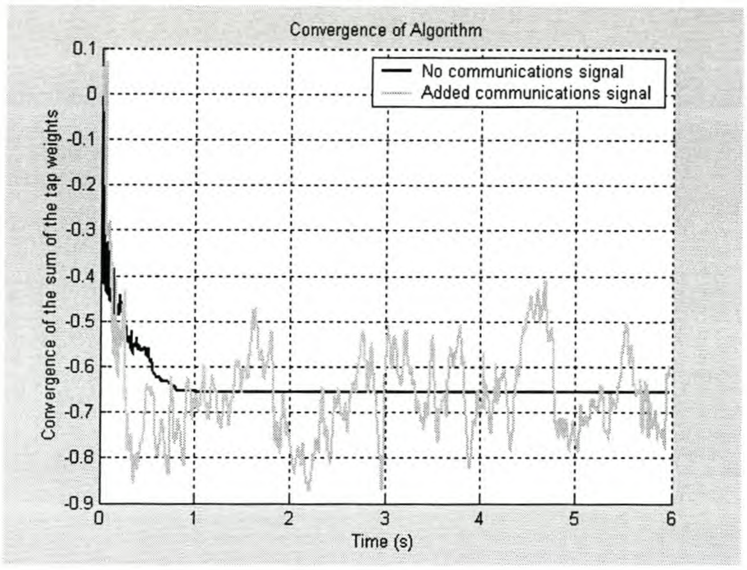


Figure 3.7 Convergence variable for stationary headset transfer estimation

The convergence variable for case two (with communications signal) constantly adapts to try and find a better solution. The convergence variable for case two was found to have a variance of 0.00884 around a mean value of -0.6638.

Both cases thus tend to produce a similar mean step response but case two has a variance around the mean step response. Both cases reached a close proximity of their mean value within 1 second of the start of the simulation. The transient filter estimation responses for both filters were completed within 1 second.

3.3.5. Loudspeaker and microphone filter influences

All previous simulations were conducted with the assumption that the transfer of the loudspeaker and microphones used in the ANR system are unity filters. In reality this is not true.

The transfer of the loudspeaker and microphones were modeled as described in section 2.2.2 and 2.2.3. This section investigates the LMS algorithm performance effects that occur due to the inclusion of these filters.

Simulations with loudspeaker and microphone transfers were run with added communications speech. These simulations were considered as the worst-case scenario for adaptive filter implementation for ANR in headsets.

Again the optimal step size for these simulations had to be found to ensure the best algorithm performance. Figure 3.8 shows the step size versus cost function relation.

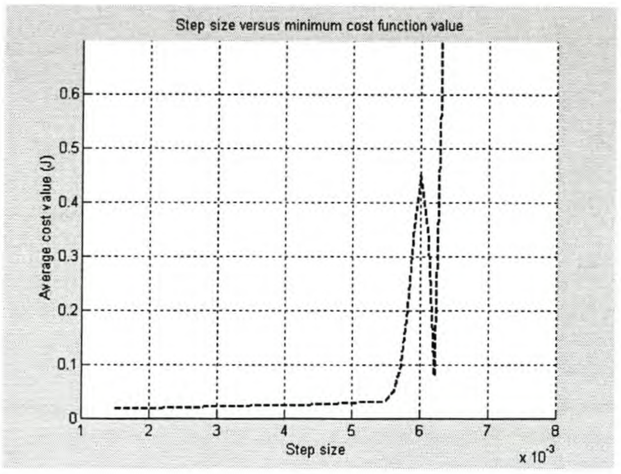


Figure 3.8 Step size versus cost function when a loudspeaker and microphone is included

The algorithm convergence time allowed was extended to 5 seconds for these simulations. This was done since the loudspeaker and microphone transfer inclusion complicates the filter to be estimated. A more complicated filter requires a smaller step size to estimate a practical filter. This smaller step size slows the convergence of the algorithms extensively and therefore a longer convergence time was required.

The $2/\lambda_{Max}$ boundary was found to be 0.006 and the optimal step size was chosen halfway between 0 and the boundary, at 0.003. The following simulation results were found in this respect.

It was found (Figure 3.9) that the LMS algorithm convergence time increased to approximately 3 seconds. A visible difference between the convergence variable mean occurred. If we recall that without loudspeaker and microphones the convergence variable mean was -0.6638 whilst with loudspeaker and microphones it was found to be 6.5572.

The difference in the convergence variable mean shows that the new estimated filter is different from the filter estimated without loudspeaker and microphone transfer. This is evident since the estimation filter convergence step response converges to a different mean value.

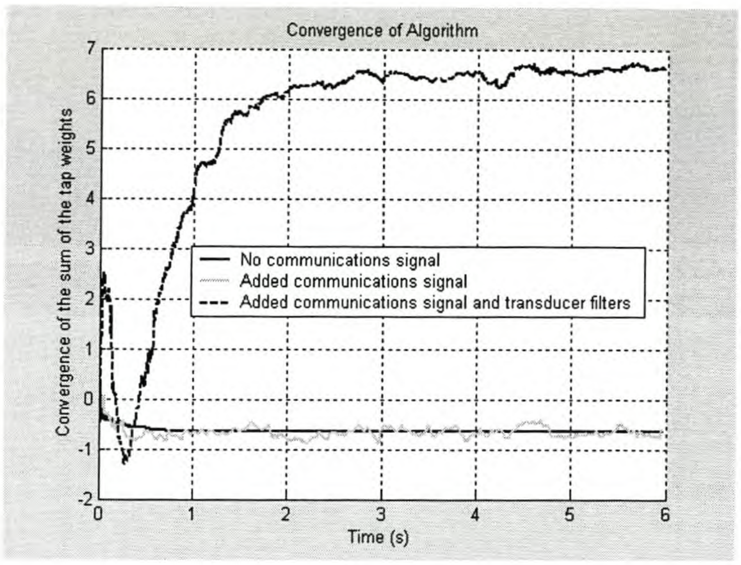


Figure 3.9 Convergence variable with loudspeaker and microphone transfer

It is predicted that the new filter estimated by the LMS algorithm has attempted to add inverse filters for the loudspeaker and microphones to the previously estimated headset transfer. This is done by the LMS algorithm in an effort to limit the effects of the transducers on the system. The inclusion of transducer transfers complicates the transfer estimation of the LMS algorithm. This can be seen from the increased convergence variable variance.

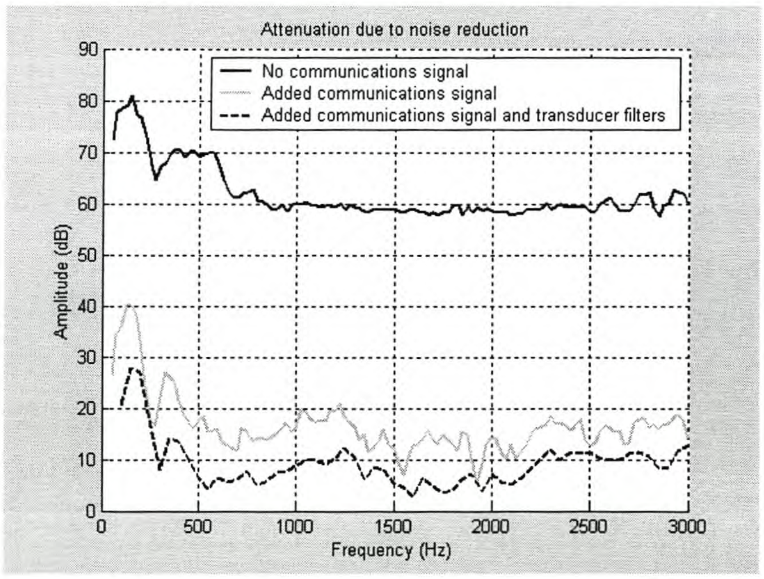


Figure 3.10 Bandwidth attenuation performance (With loudspeaker/microphone filters)

The total attenuation of signal energy after convergence of the algorithm in the new simulation deteriorated from 28.467dB to 16.646dB, the deteriorated bandwidth attenuation performance is shown in Figure 3.10.

In an attempt to find a better solution, these filter effects were included in the LMS algorithm derivation of section 3.1. The microphone filters were ignored at first.

This changed the LMS cost function of equation 3.4 as shown in equation 3.18. A graphical representation of the block diagram can be seen in Figure 3.11.

$$J(\mathbf{w},t) = E\left[d(n) - \sum_{i=0}^{M-1} s_i \mathbf{w}^T(n-i)\mathbf{u}(n-i)\right]^2$$
$$J(\mathbf{w},t) = E\left[d(n) - \sum_{i=0}^{M-1} s_i y(n-i)\right]^2 \qquad 3.18$$

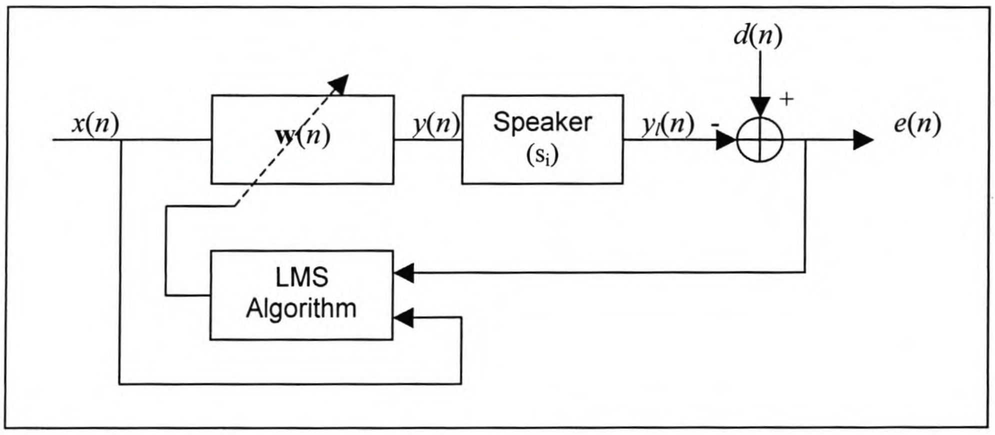


Figure 3.11 Inclusion of loudspeaker filter into model

s_i is the loudspeaker filter tap weights. The derivative of the new cost function is calculated from equation 3.18. The result is shown in equation 3.19.

$$\begin{aligned}
\frac{\partial J(n)}{\partial \mathbf{w}(n)} = \nabla \mathbf{J}(n) &= -2E\left[\left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)d(n)\right] + 2E\left[\left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)\left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)^T\right]\mathbf{w}(n) \\
&= -2E\left[\left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)d(n) - \left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)\left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)^T \mathbf{w}(n)\right] \\
&= -2E\left[\left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)\left[d(n) - \left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)^T \mathbf{w}(n)\right]\right] \\
&= -2E\left[\left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)\left[d(n) - \mathbf{w}^T(n)\left(\sum_{i=0}^{M-1} s_i \mathbf{u}(n-i)\right)\right]\right]
\end{aligned} \qquad 3.19$$

Substituting this result in the steepest descent algorithm of equation 3.3 gives an improved update algorithm known as the filtered-x LMS algorithm [19].

$$\mathbf{w}(n+1)=\mathbf{w}(n)-\frac{\mu}{2}[\nabla\mathbf{J}(n)]$$

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\mu E\left[\left(\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right)\left[d(n)-\mathbf{w}^T(n)\left(\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right)\right]\right] \qquad 3.20$$

Convergence in the mean, implying

$E\left[\mathbf{w}(n)\right]\rightarrow\mathbf{w}_o \quad as \quad n\rightarrow\infty$

and convergence in the mean square, implying

$J(n)\rightarrow J(\infty) \quad as \quad n\rightarrow\infty$

is once again assumed. Equation 3.20 simplifies to equation 3.21.

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\mu\left(\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right)\left[d(n)-\mathbf{w}^T(n)\left(\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right)\right] \qquad 3.21$$

Equation 3.21 implies that filtering the adaptive filter input with a filter that has the same filter characteristics as the loudspeaker filter could improve the headset filter estimation.

The inclusion of the filtered-x filter changes the block diagram of Figure 3.11 as shown in Figure 3.12.

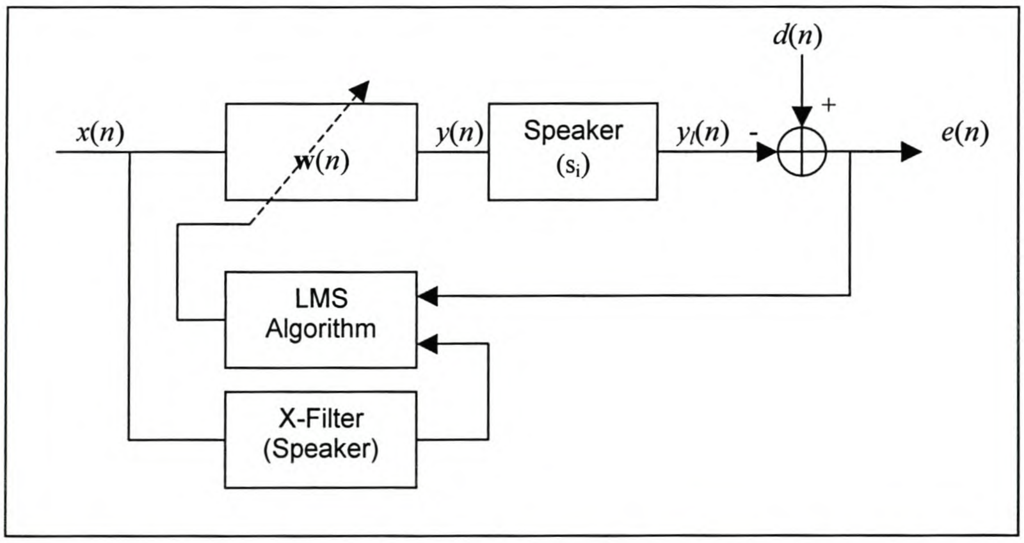


Figure 3.12 X-filter to compensate for the loudspeaker transfer in the feed forward error

Inclusion of the microphone filters shows that the microphone filters are already in similar filtered-x configuration with the adaptive filter if it is assumed that both microphones have the same transfer properties. It is thus unnecessary to develop an x-filter for the microphones.

Using the new algorithm entailed re-establishing the optimal step size. The step size experiments were repeated and the results are shown in Figure 3.13.

It can be seen that the inclusion of the x-filter restores the original $2/\lambda_{Max}$ boundary as found in step one of section 3.2.1. This implies that the filtered-x algorithm restores the operation step size range to the same range for when the loudspeaker and microphone is not included. The cost function values are also somewhat closer to zero, and this shows that this algorithm should give improved attenuation.

Another experimental ANR simulation was run to compare the x-filtered performance with previous experiments. The experiment was preformed at 6kHz, with a communications signal, loudspeaker and microphone filters and an x-filter that has the same filter characteristics as the loudspeaker. The step size was chosen at 0.003. The following results were generated.

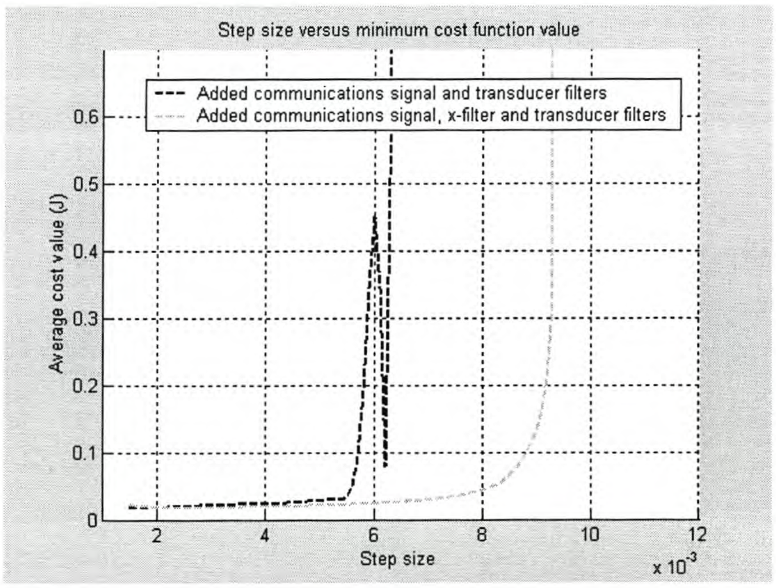


Figure 3.13 Step size versus optimal cost function value for the filtered-x LMS

The un-filtered-x convergence variable mean was 6.5572 while the filtered-x case gave a convergence variable mean of 6.9449. Once again it can be seen that the convergence variable for the filtered-x case approaches the new filter needed due to the inclusion of the loudspeaker and microphone.

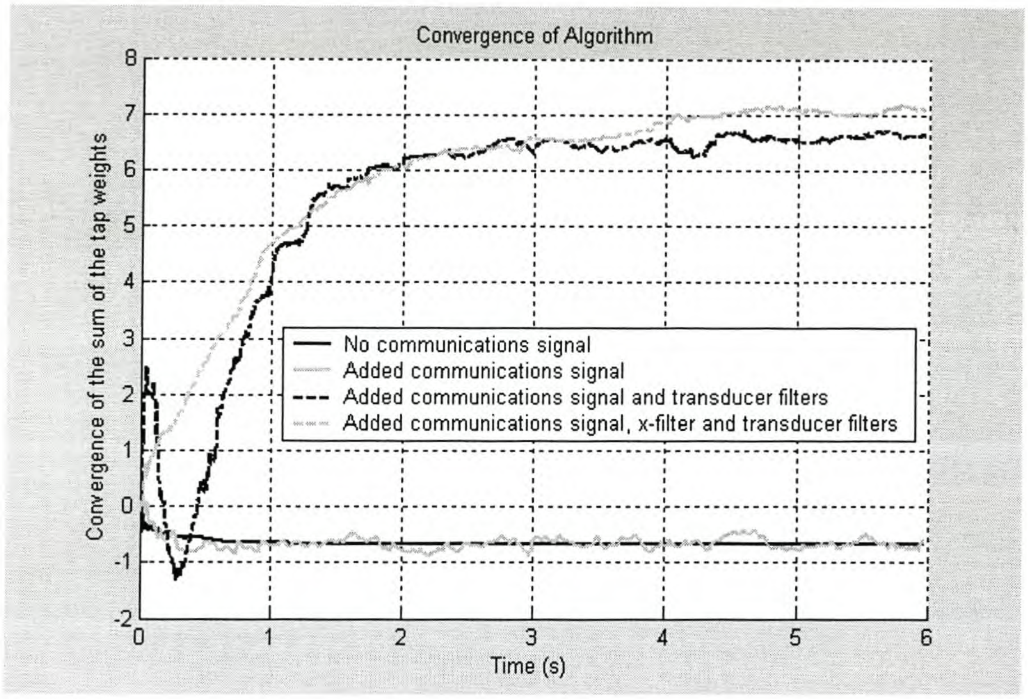


Figure 3.14 Convergence variable with x-filter

The convergence variable variance was found to reduce from 0.0119 to 0.00552. From Figure 3.14 it can also be seen that the convergence is smoother and thus less random. The bandwidth attenuation performance for the filtered-x case is shown in Figure 3.15.

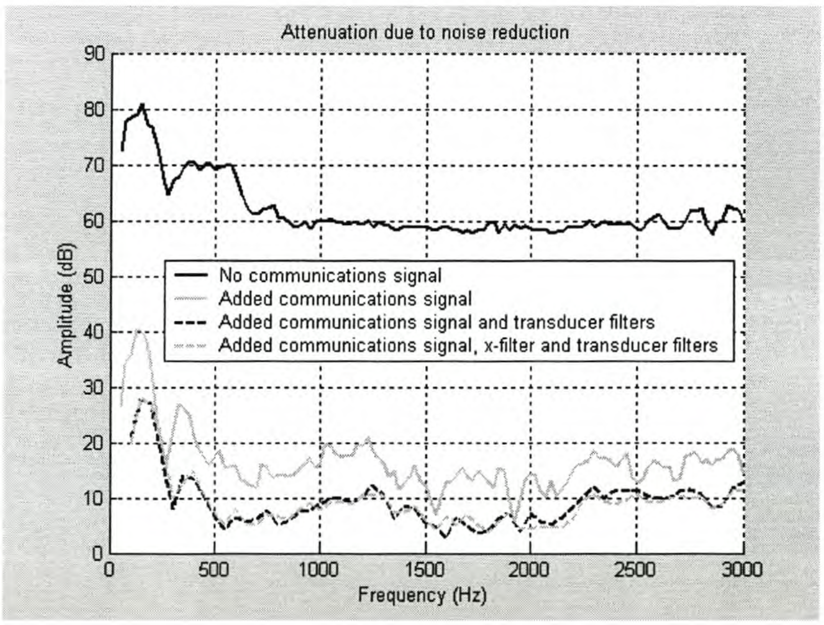


Figure 3.15 Filtered-x bandwidth attenuation relation

From Figure 3.15 it is not clearly shown that the filtered-x performance is better than when no x-filter is used. The filtered-x attenuation of signal energy however shows that the experiment with an x-filter gives a 1.553dB improvement since the total signal attenuation improved from 16.646dB to 18.199dB.

From the above results it can be seen that the filtered-x simulation improves the ANR performance of the simulation but does not cancel the effect of microphone and loudspeaker transfer on the system.

3.4. <u>LMS algorithm conclusion</u>

The working of the LMS algorithm was explained in this section and the performance of the algorithm was evaluated in a simulation environment.

From the results found in this chapter it can be concluded that the filtered-x LMS algorithm will be satisfactory to perform ANR in headsets. The maximum simulated attenuation for this algorithm was found to be 18.199dB.

It should however be kept in mind that in reality it is hard to estimate an exact transfer for the loudspeaker for the purposes of a filtered-x implementation since this transfer is not stationary [4]. It could be more practical to implement the LMS algorithm without the filtered-x configuration.

The data extracted from this chapter will be compared with that of other adaptive filter algorithms in chapter 6. This will enable the choice of an optimal adaptive filter for ANR.

Chapter 4

4 Recursive Least-Squares Estimation Algorithm

This chapter investigates the use of the standard recursive least-squares estimation algorithm to implement ANR in aircraft used headsets.

The RLS algorithm is based on least squares estimation. The adaptive filter is fairly complex to derive but is known for its leniency in getting a working adaptive filter fairly easily. This adaptive filter implementation has many similarities to the implementation of Kalman filters and more specifically the dynamic autoregressive Kalman filter.

This chapter will explain the RLS algorithm derivation and performance for an ANR environment.

4.1. Algorithm outline and operation

First a cost function must be defined for the RLS algorithm. Since The RLS algorithm is based on least squares estimation, the cost function must minimise the squared error of the system. This is done as shown in equation 4.1.

$$J(\mathbf{w},t) = \sum_{n=1}^{i} \lambda^{i-n} \left| d(n) - \mathbf{w}^T(n)\mathbf{u}(n) \right|^2 \qquad 4.1$$

Where $d(n)$ is the desired filter output

$\mathbf{u}(n)$ is the filter input observation matrix

$\mathbf{w}(n)$ is the estimated filter tap-weights computed by the RLS algorithm

λ^{i-n} is a weighing or forgetting factor. The purpose of the forgetting factor is to exponentially weigh data from the distant past lower than current data. This is required to improve the non-stationary transfer estimation ability of the algorithm since ignoring distant past data, ignores old changes in the transfer estimation and makes the algorithm more susceptible to current transfer changes.

The optimum tap-weight solution for equation 4.1 is obtained from the least squares theory and is known as the normal equations [8], [11]. Equation 4.2 shows the definition of the normal equations.

$$\mathbf{R}(n)\mathbf{w}_o(n) = \mathbf{Q}(n) \qquad 4.2$$

$\mathbf{Q}(n)$ represents the time dependant cross correlation matrix between the desired system output, $d(n)$, and observation matrix $\mathbf{u}(n)$ with exponential forgetting as shown in equation 4.3. No complex signals will be used in the noise reduction applications, thus all derivations and equations will only be relevant for real signals.

$$\mathbf{Q}(n) = \sum_{n=i}^{i} \lambda^{i-n}\mathbf{u}(i)d(i) \qquad 4.3$$

$\mathbf{R}(n)$ represents the correlation matrix of the input observations, $\mathbf{u}(n)$ with exponential forgetting and is shown in equation 4.4.

$$\mathbf{R}(n) = \sum_{n=i}^{i} \lambda^{i-n}\mathbf{u}(i)\mathbf{u}^T(i) \qquad 4.4$$

Equation 4.3 and equation 4.4 can be adapted to be recursively updateable as shown in equation 4.5 and equation 4.6.

$$\begin{aligned}\mathbf{Q}(n) &= \sum_{n=i}^{i} \lambda^{i-n}\mathbf{u}(i)d(i) \\ &= \lambda\left[\sum_{n=i}^{i-1} \lambda^{i-1-n}\mathbf{u}(i)d(i)\right] + \mathbf{u}(n)d(n) \\ &= \lambda\mathbf{Q}(n-1) + \mathbf{u}(n)d(n)\end{aligned} \qquad 4.5$$

$$\begin{aligned}\mathbf{R}(n) &= \sum_{n=i}^{i} \lambda^{i-n}\mathbf{u}(i)\mathbf{u}^T(i) \\ &= \lambda\left[\sum_{n=i}^{i-1} \lambda^{i-1-n}\mathbf{u}(i)\mathbf{u}^T(i)\right] + \mathbf{u}(n)\mathbf{u}^T(n) \\ &= \lambda\mathbf{R}(n-1) + \mathbf{u}(n)\mathbf{u}^T(n)\end{aligned} \qquad 4.6$$

To calculate recursively updated tap-weights for equation 4.2, it is required to calculate a recursive inverse correlation matrix ($\mathbf{R}(n)^{-1}$). This can be achieved with the use of the matrix inversion lemma of matrix algebra. The inversion lemma is also known as Woodbury's identity [20].

The matrix inversion lemma states that the inverse of a matrix can be found if it is written in the following form

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^{T} \qquad 4.7$$

Where $\mathbf{A}$ and $\mathbf{B}$ must both be positive definite M-by-M matrices

$\mathbf{D}$ is another positive definite N-by-N matrix

$\mathbf{C}$ is an M-by-N matrix

The inverse matrix can then be calculated as shown in equation 4.8.

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^{T}\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^{T}\mathbf{B} \qquad 4.8$$

The RLS algorithm can now be formulated if the following substitution is made.

$\mathbf{A} = \mathbf{R}(n)$ $\qquad$ $\mathbf{B}^{-1} = \lambda\,\mathbf{R}(n\text{-}1)$

$\mathbf{C} = \mathbf{u}(n)$ $\qquad$ $\mathbf{D} \;= 1$

Thus from the matrix inversion lemma the recursive inverse correlation matrix can be formulated as

$$\mathbf{R}^{-1}(n) = \lambda^{-1}\mathbf{R}^{-1}(n-1) - \frac{\lambda^{-2}\mathbf{R}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^{T}(n)\mathbf{R}^{-1}(n-1)}{1 + \lambda^{-1}\mathbf{u}^{T}(n)\mathbf{R}^{-1}(n-1)\mathbf{u}(n)} \qquad 4.9$$

For simplicity and convenience equation 4.9 is redefined by substituting

$$\mathbf{P}(n) = \mathbf{R}^{-1}(n)$$

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1 + \lambda^{-1}\mathbf{u}^{T}(n)\mathbf{P}(n-1)\mathbf{u}(n)}$$

This gives equation 4.10.

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^T(n)\mathbf{P}(n-1) \qquad 4.10$$

$\mathbf{k}(n)$ is referred to as the gain vector. If the gain vector is rearranged, the following relation is found.

$$\begin{aligned}\mathbf{k}(n) &= \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{u}(n)} \\ &= \lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n) - \mathbf{k}(n)\lambda^{-1}\mathbf{u}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{u}(n) \\ &= \left[\lambda^{-1}\mathbf{P}(n-1) - \mathbf{k}(n)\lambda^{-1}\mathbf{u}^T(n)\mathbf{R}^{-1}(n-1)\right]\mathbf{u}(n) \qquad 4.11 \\ &= \mathbf{P}(n)\mathbf{u}(n) \\ &= \mathbf{R}^{-1}(n)\mathbf{u}(n)\end{aligned}$$

The significance of this relation will become clear as the time update for the tap-weight vector is derived from equation 4.5, 4.6, 4.10 and 4.11.

$$\begin{aligned}\mathbf{w}(n) &= \mathbf{R}^{-1}(n)\mathbf{Q}(n) \\ &= \mathbf{P}(n)\mathbf{Q}(n) \\ &= \lambda\mathbf{P}(n)\mathbf{Q}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d(n) \\ &= \mathbf{P}(n-1)\mathbf{Q}(n-1) - \mathbf{k}(n)\mathbf{u}^T(n)\mathbf{P}(n-1)\mathbf{Q}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d(n) \\ &= \mathbf{R}^{-1}(n-1)\mathbf{Q}(n-1) - \mathbf{k}(n)\mathbf{u}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{Q}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d(n)\end{aligned}$$

Now substituting

$\mathbf{w}(n-1) = \mathbf{R}^{-1}(n-1)\mathbf{Q}(n-1)$

it is found that

$$\begin{aligned}\mathbf{w}(n) &= \mathbf{w}(n-1) - \mathbf{k}(n)\mathbf{u}^T(n)\mathbf{w}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d(n) \\ &= \mathbf{w}(n-1) - \mathbf{k}(n)\mathbf{u}^T(n)\mathbf{w}(n-1) + \mathbf{k}(n)d(n) \\ &= \mathbf{w}(n-1) + \mathbf{k}(n)\left[d(n) - \mathbf{u}^T(n)\mathbf{w}(n-1)\right] \qquad 4.12 \\ &= \mathbf{w}(n-1) + \mathbf{k}(n)\left[d(n) - \mathbf{w}^T(n-1)\mathbf{u}(n)\right]\end{aligned}$$

The total RLS algorithm can thus be summarised into four equations that must be preformed in the given sequence to estimate filter tap-weights for a adaptive filter with desired signal $d(n)$ and input history $\mathbf{u}(n)$. The recursive equations are given in 4.13.

$$
\begin{aligned}
\mathbf{k}(n) &= \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{u}(n)} \\
e(n) &= d(n)-\mathbf{w}^T(n-1)\mathbf{u}(n) \\
\mathbf{w}(n) &= \mathbf{w}(n-1)+\mathbf{k}(n)e(n) \\
\mathbf{P}(n) &= \lambda^{-1}\mathbf{P}(n-1)-\lambda^{-1}\mathbf{k}(n)\mathbf{u}^T(n)\mathbf{P}(n-1)
\end{aligned}
\qquad 4.13
$$

If these equations are compared with that of the dynamic autoregressive Kalman filter, an exact comparison is found. The dynamic autoregressive Kalman filter does however not allow exponential forgetting and is thus more of a least squares estimation.

The performance of the RLS algorithm was investigated and the results are shown in the following section of this chapter.

4.2. <u>Algorithm performance</u>

The algorithm performance was established according to the specifications set in section 2.4 of chapter 2 and the simulation were implemented as specified in chapter 2.

4.2.1. Algorithm parameters

The only variable parameter concerned with the RLS algorithm is the forgetting factor (λ). The forgetting factor in essence weighs past values of the least squares error to give present values more weight than past values. This is helpful to improve the RLS algorithm performance for non-stationary transfer estimation.

From [8], a forgetting factor value close to one ensures better performance results since a λ smaller than one changes the operation of the RLS algorithm and causes weight vector noise, weight vector lag and a increase in the total value of the average excess mean-squared error. Thus for λ much smaller than one the losses are to overwhelming and no performance gain can be achieved, but for values very close to one, some performance improvements can be found for non-stationary transfer estimations.

Some simulations were implemented to determine the characteristics and optimal value for the forgetting factor in the estimation of the headset transfer function.

Forgetting factor values from 0.8 to 1 were simulated in the ANR simulation for the chosen stationary headset transfer. The simulations were repeated twice, once for no added communications signal and once with a communications signal. The results are shown in Figure 4.1.

From Figure 4.1 it can be seen that the squared error decreases as λ grows closer to 1 and goes to a minimum for λ=1. The minimum error is also expected at λ=1 according to [8], [11].

This is expected since the best estimate for a non-stationary process will be an estimate without any weight vector noise, weight vector lag and minimum average mean-squared error.

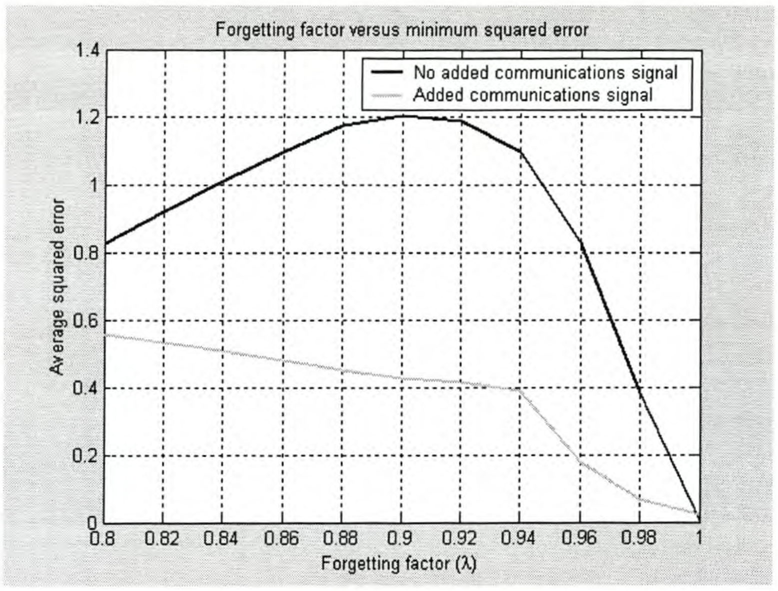


Figure 4.1 ANR attenuation vs. forget factor for estimation of a stationary process

As mentioned before and according to [8], [11] a λ<1 can improve the performance of the RLS adaptive filter when a slowly varying non-stationary process is estimated. This analysis is however only sensible if it is clear what is meant by a slow varying process in relation to a chosen forgetting factor.

We know that the forgetting factor weighs past error measurements as given in equation 4.1. Thus equation 4.1 can be rewritten as equation 4.14.

$$J(\mathbf{w},t)=\sum_{n=1}^{i}\lambda^{i-n}\left|y(n)-\mathbf{w}^{T}(n)\mathbf{u}(n)\right|^{2}$$
$$J(\mathbf{w},t)=\lambda^{i-1}\left|e(1)\right|^{2}+\lambda^{i-2}\left|e(2)\right|^{2}+\ldots.+\lambda^{1}\left|e(i-1)\right|^{2}+\lambda^{0}\left|e(i)\right|^{2} \qquad 4.14$$

If we assume that squared error contributions ($|e(i-n)|^2$) multiplied by λ^{i-n}= 0.001 are neglectable to the current cost function, then we can calculate how far in the past relevant squared errors are kept for updating the current adaptive filter tap weights.

Table 4.1 was constructed to illustrate how far in the past the error signal samples are resident in the algorithm to update the current filter tap weights. The error sample resident time is related to a chosen forgetting factor and history limit set by choosing a minimum multiplication cutoff for equation 4.14. The past error multiplication limit was set at λ^N=0.001. This implies that it is assumed that past squared error samples multiplied by values smaller than this limit gives little or no contribution to the cost function of equation 4.14.

λ Value	**Weighting close to 0.001**	**Memory Time (s)** (number of samples/Sample rate)
$\lambda = 1.00$	$\lambda^{\infty} = 1.00000$	$\infty/6000 = \infty$
$\lambda = 0.99$	$\lambda^{680} = 0.00108$	680/6000 = 0.1133s
$\lambda = 0.98$	$\lambda^{340} = 0.00104$	340/6000 = 0.0566s
$\lambda = 0.97$	$\lambda^{225} = 0.00106$	225/6000 = 0.0375s

Table 4.1 RLS Forgetting Factor Memory Time Table

From Table 4.1 it can be seen that a non-stationary process that time varies slower than once every 0.1133 seconds will seem stationary for any forgetting factor smaller than 0.99, given the related error multiplication limit.

The non-stationaraty of headset transfer was expected to vary slower than once every 0.1133 seconds. Although the simulation does not support non-stationary headset transfer, the above findings show that investigating forgetting factors for non-stationary headset transfer would be unnecessary for the RLS algorithm if non-stationary headset transfer were modeled in the simulation model of chapter 2.

Since the headset transfer was seen as a stationary process, the optimal forgetting factor (λ) was always chosen at λ=1 from [8], [11] and Figure 4.1.

4.2.2. Filter topology and tap size of adaptive filter

To ensure stable filters, all filters to be estimated were chosen as all zero finite impulse response (FIR) filters. Different filter orders were tested in a simulation environment with communication speech signal and with out communications speech. The optimal λ, (λ=1) was used.

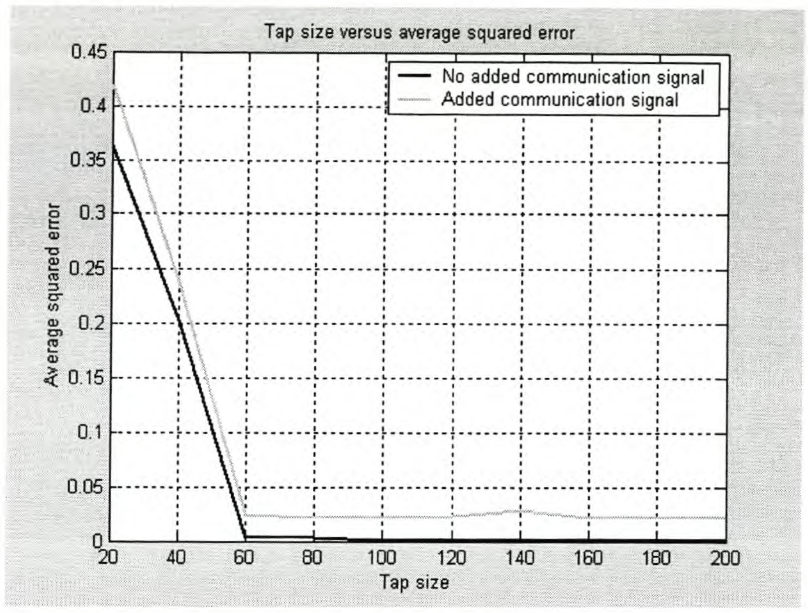


Figure 4.2 Average squared error vs. number of RLS taps

Figure 4.2 shows the results for filter order versus average squared error found in the simulations. It is clear that any filter order above 60 would estimate the simulated headset transfer equally well.

The choice of the number of taps can however be limited by the processing ability of the DSP chip that will be used to implement the RLS algorithm. A higher tap order will enlarges the size of the inverse correlation matrix, $\mathbf{P}(n)$, thus exponentially increasing the amount of calculations needed to calculate $\mathbf{P}(n)$.

To ensure good filter estimates in the simulated environment, the tap size for the RLS algorithm was chosen to be 100 taps.

4.2.3. Bandwidth attenuation performance (No communication vs. communication)

This section investigated the RLS algorithm bandwidth attenuation properties for when a communications signal is present versus when no communications signal is present. The bandwidth attenuation will once again be compared with the total signal energy attenuation calculated by equation 3.17 of chapter 3.

The attenuation versus frequency bandwidth results attained from these simulations is shown in Figure 4.3.

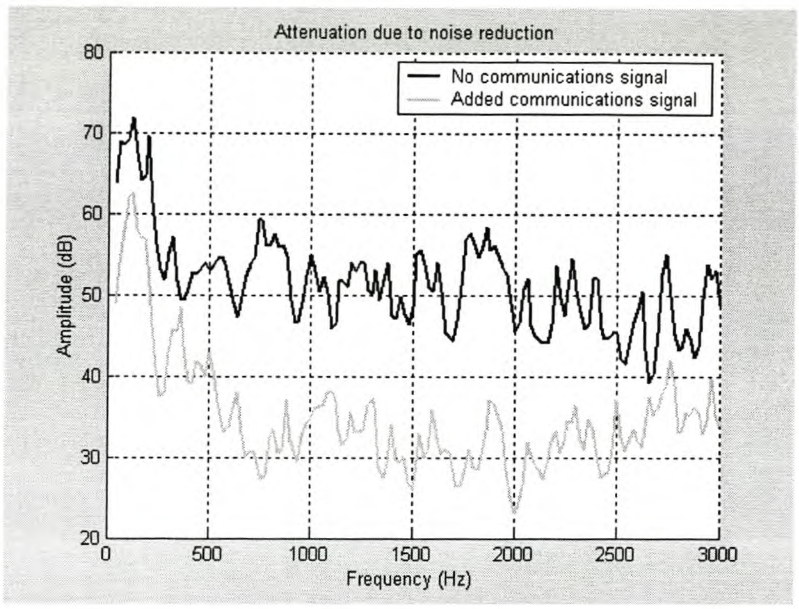


Figure 4.3 Frequency bandwidth versus attenuation

From Figure 4.3 it is seen that the simulation with no communications signal ensures a much better attenuation of noise than when a communication signal is included.

The signal energy attenuation (calculated from equation 3.17) was found to be 62.517dB for the no communications case. This value reduced to 48.994dB when a communications signal is added.

4.2.4. Convergence considerations

The convergence of the RLS algorithm was also evaluated according to the convergence variable as described in section 2.4 of chapter 2.

The convergence performance for the RLS algorithm was investigated for a system with communications signal and system without a communications signal. Figure 4.4 shows the convergence variable behaviour found.

From Figure 4.4 it can be seen that the RLS algorithm has a fast initial convergence and then slowly keeps converging to find an optimal solution. The convergence variable mean was found to be -0.6560 for no added communication signal, and -0.6551 for an added communications signal. From the above it can thus be seen that the same filter is being estimated in both cases.

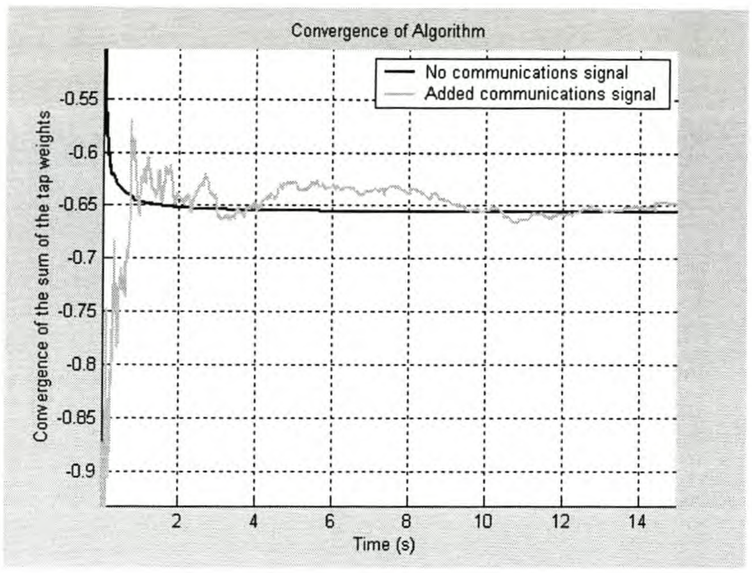


Figure 4.4 RLS algorithm convergence

The convergence variable variance increased for added communications signal. This value rose from 0.00066541x10^{-5} to 2.1323x10^{-5}. This shows that this algorithm also finds it more difficult to estimate the optimal filter when a communications signal is present.

4.2.5. Speaker and microphone effects

The inclusion of a loudspeaker and microphones to the RLS algorithm simulation will be investigated in this section.

The optimal forgetting factor value for this configuration was investigated first since the inclusion of a loudspeaker and microphone filter was found to hinder the normal operation of an adaptive filter in this configuration.

Forgetting factors from 0.8 to 1 was simulated with a convergence limit of 0.5 seconds. This means that the RLS tap weights were not analysed over the first 0.5 seconds since this time was given to the adaptive filter to converge to the optimal tap weights. This convergence limit might seem as a short time but it is adequate to establish what forgetting factor gives the smallest squared error or cost function value. The resulting forgetting factor versus average squared error plot is shown in Figure 4.5.

From Figure 4.5 it can be seen that the optimal forgetting factor value will once again be $\lambda=1$. This is expected since the speaker and microphone transfers for the simulation is also stationary. A simulation was run with an added communications signal and speaker/microphone transfer included.

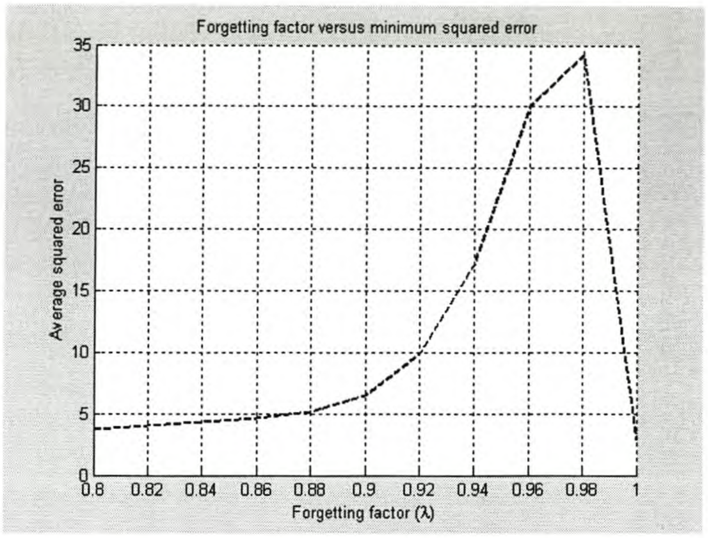


Figure 4.5 Forgetting factor versus average squared error

The following convergence, bandwidth versus attenuation and signal power attenuation results were found in relation with previous results.

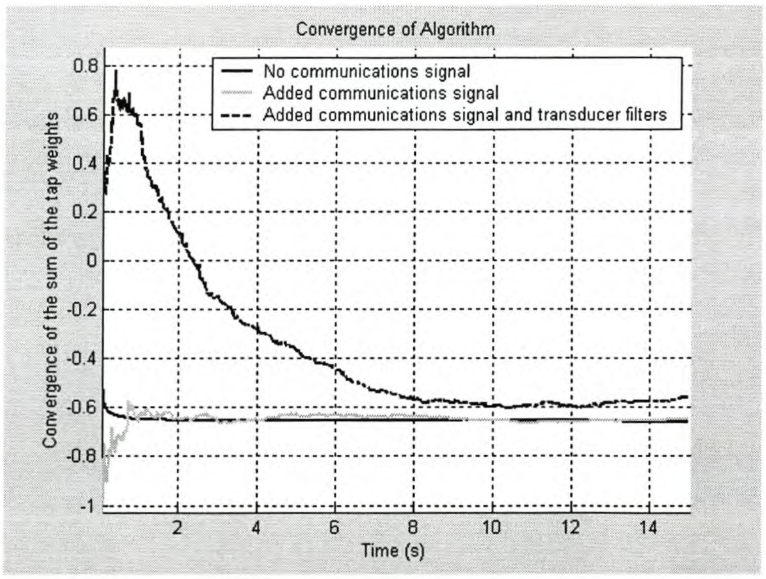


Figure 4.6 Convergence variable for added speaker and microphone transfer

Figure 4.6 show that the added speaker and microphones hinder the convergence of the RLS algorithm to an undesirable extent. The bandwidth versus attenuation response for this case is shown in Figure 4.7.

A very reduced attenuation of noise can be seen from Figure 4.7. The attenuation of signal energy was found to be 6.0324dB. This is drastically deteriorated from the previous found 48.994dB for no speaker and microphones included.

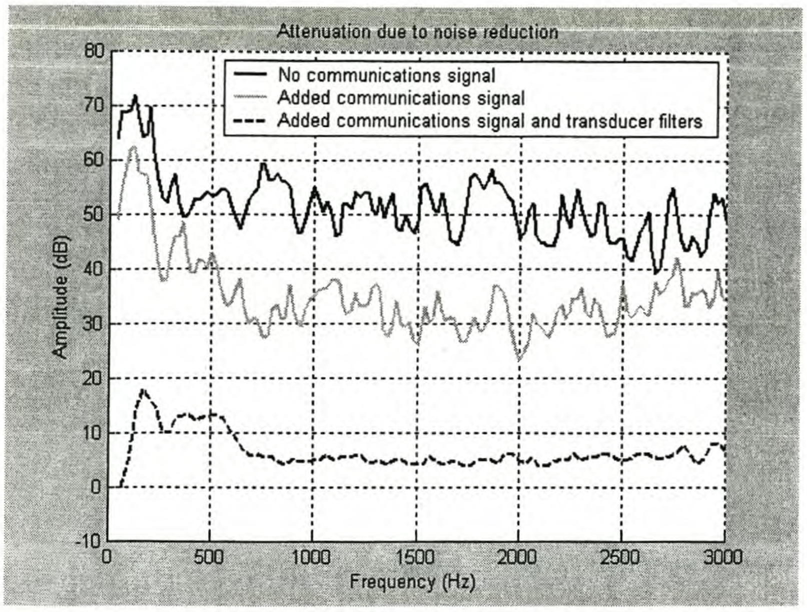


Figure 4.7 Bandwidth versus attenuation response with added speaker and microphones

An investigation into finding a RLS algorithm that makes provision for these transducer transfers was made but few implementable algorithm solutions were available. Research in this area is being done according to [21].

A possible solution is given in [4] were the RLS algorithm is implemented in a different topology. It was decided that investigating this solution would be contrary to the aim of finding an optimal algorithm for the topology of Figure 2.2 of chapter 2. This possible topology may be investigated in further work.

4.3. <u>RLS algorithm conclusion</u>

The working of the RLS algorithm was explained in this section and the performance of the algorithm was evaluated according to a standard simulation environment as depicted in chapter 2.

It was found that the RLS algorithm has a very good disturbance signal rejection capability (communications signal) but cannot adapt well if transducer transfers disrupt the error and reference signals.

The data extracted from this chapter will be compared with that of other adaptive filter algorithms in chapter 6.

Chapter 5

5 Kaczmarz Projection (KP) Algorithm

This chapter investigates the use of the Kaczmarz Projection Algorithm as an estimation filter to implement ANR in aircraft used headsets.

The Kaczmarz projection algorithm resembles the LMS algorithm, but is derived with the aim to normalise the step size parameter of the LMS algorithm. This is done so that the filter estimation preformed by this algorithm is independent from the statistics of the reference noise to the algorithm.

The chapter will also investigate the Kaczmarz Projection Algorithm derivation and performance

5.1. Algorithm outline and operation

The Kaczmarz projection algorithm is also known as the normalised least mean square algorithm and aims to improve the LMS algorithm by making the step size parameter of the LMS algorithm adaptive.

This step size can be optimised by minimising its square Euclidean norm [11], [12]. This can be represented as shown in equation 5.1. It can be seen that equation 5.1 relates to the weight updating equation of the LMS algorithm and is concerned with optimising the step difference between tap weight updates.

$$\delta\mathbf{w}(n+1)=\mathbf{w}(n+1)-\mathbf{w}(n) \qquad 5.1$$

Equation 5.1 must be subject to the constraint that an reference input signal filtered by the estimated filter will give the desired signal output. The desired output in this case is an exact representation of the noise inside the earmuff. This relation is shown by equation 5.2. This constraint will ensure implementation of this adaptive filter.

$$\mathbf{w}^T(n+1)\mathbf{u}(n)=d(n) \qquad 5.2$$

To solve this optimisation problem, the method of Lagrange multipliers is used. The method of Lagrange multipliers will not be explained in this section since it is of little significance to the topic of this thesis. A detailed explanation can be found in Appendix C of [11].

Since only real signals are used in this adaptive filter application, all derivations were made relevant to real signals only.

According to the Lagrange multiplier method, the cost function for this problem can be given as shown in equation 5.3. [11]

$$J(\mathbf{w},t)=\frac{1}{2}\left[\mathbf{w}(n+1)-\mathbf{w}(n)\right]^T\left[\mathbf{w}(n+1)-\mathbf{w}(n)\right]+\overline{\alpha}\left[d(n)-\mathbf{w}^T(n+1)\mathbf{u}(n)\right] \quad 5.3$$

The Lagrangian multiplier ($\overline{\alpha}$) is given to handle the constraint of equation 5.2. The optimal solution of equation 5.3 can be found from the following derivation.

Firstly the derivative of the cost function of equation 5.3 is found with respect to $\mathbf{w}(n+1)$. This will give equation 5.4.

$$\frac{\partial J(n)}{\partial \mathbf{w}(n+1)}=\left[\mathbf{w}(n+1)-\mathbf{w}(n)\right]-\overline{\alpha}\mathbf{u}(n) \quad 5.4$$

Setting the derivative equal to zero will thus produce the optimal solution for $\mathbf{w}(n+1)$.

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\overline{\alpha}\mathbf{u}(n) \quad 5.5$$

The solution of the Lagrangian multiplier ($\overline{\alpha}$) can be found by substituting equation 5.5 into the Lagrangian constraint of equation 5.2 as shown in the derivation of 5.6.

$$\begin{aligned} d(n)&=\mathbf{w}^T(n+1)\mathbf{u}(n)\\ &=\left[\mathbf{w}(n)+\overline{\alpha}\mathbf{u}(n)\right]^T\mathbf{u}(n)\\ &=\mathbf{w}^T(n)\mathbf{u}(n)+\overline{\alpha}\mathbf{u}^T(n)\mathbf{u}(n)\\ &=\mathbf{w}^T(n)\mathbf{u}(n)+\overline{\alpha}\left\|\mathbf{u}(n)\right\|^2 \end{aligned} \quad 5.6$$

Now

$$\bar{\alpha} = \frac{d(n) - \mathbf{w}^T(n)\mathbf{u}(n)}{\|\mathbf{u}(n)\|^2}$$
$$\bar{\alpha} = \frac{e(n)}{\|\mathbf{u}(n)\|^2} \qquad 5.7$$

Were $e(n)$ is the error between the desired and estimated signals, as shown in equation 5.8

$$e(n) = d(n) - \mathbf{w}^T(n)\mathbf{u}(n) \qquad 5.8$$

The final tap weight update equation for the Kaczmarz projection algorithm or normalised LMS algorithm can thus be given as shown in equation 5.9.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e(n) \qquad 5.9$$

The algorithm will need some adjustments to make it more practical. By adding an extra parameter (γ), the step length of the adjustment parameter can be adjusted. It can be proven from [11] and [12] that γ is bounded by $0<\gamma<2$.

By adding yet another parameter (α), we can ensure that no potential problems occur when $\mathbf{u}(n) = 0$. We choose $\alpha > 0$ and reasonably small. The resulting equation is shown in equation 5.10.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\gamma}{\alpha + \|\mathbf{u}(n)\|^2}\mathbf{u}(n)e(n) \qquad 5.10$$

The form given in equation 5.10 resembles the LMS algorithm of chapter 3. The only difference is that the LMS weighting constant (μ) now becomes adaptive in the Kaczmarz algorithm. It is for this reason that it is expected that the Kaczmarz algorithm will have distinct characteristics of it's own as an adaptive filter solution.

5.2. Algorithm performance

The Kaczmarz Projection algorithm was evaluated to the same criteria as given in section 2.4 of chapter 2.

5.2.1. Algorithm parameters

Just like the LMS algorithm the KP algorithm only has one variable namely the step length parameter (γ). The optimal choice of the γ-parameter was investigated in simulation.

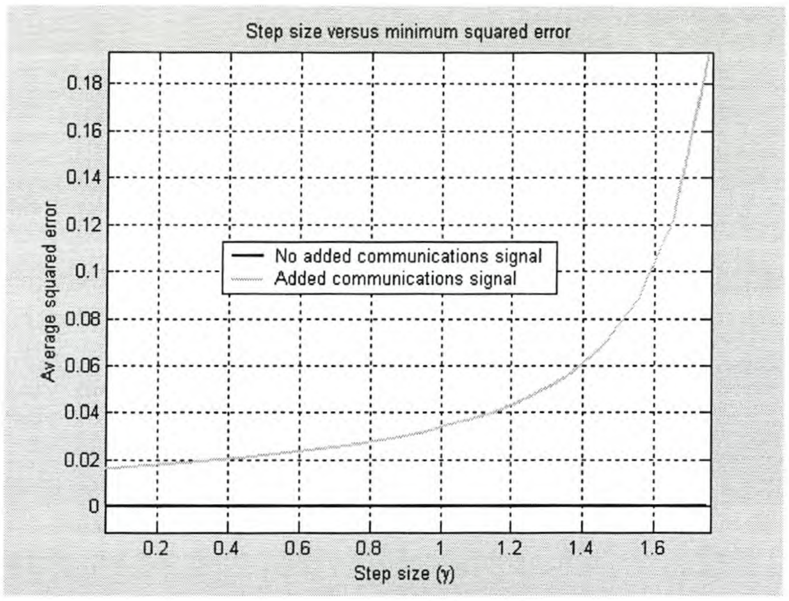


Figure 5.1 Step size versus average squared error (With and without Communications Signal)

The step size versus average squared error for a simulation with no communications signal was found and compared to a simulation with communications signal. The results are shown in Figure 5.1.

No slow convergence limit issues were found for this algorithm as was found in the LMS algorithm. This is attributed to the fact that the starting step size values are chosen reasonably large. The average squared error was calculated for the second to third seconds of simulation. This implies that the algorithm converged within 2 seconds.

There is a significant difference between the magnitude average squared error for a added communications signal as apposed to no communications signal but both cases followed the same average squared error versus step size behavior. The optimal step size value was chosen small. The step size was chosen at $\gamma = 0.1$.

5.2.2. Filter topology and tap size of adaptive filter

Once again it was decided to estimate stable all zero FIR filters of different orders. Different filter orders were tested in simulation with no communication signal and with communication signal. These simulations were implemented for the chosen γ, (γ = 0.1) and stationary headset.

From Figure 5.2 it can be seen that the minimum average squared error for when a communication signal is present and when no communication is present can be found for a filter order chosen bigger than 60. The adaptive filter order was chosen at 100 to ensure best estimation, the same as for the LMS algorithm.

This number of taps will ensure that the requirements for the DSP chip are not to labour intensive to perform to many calculations

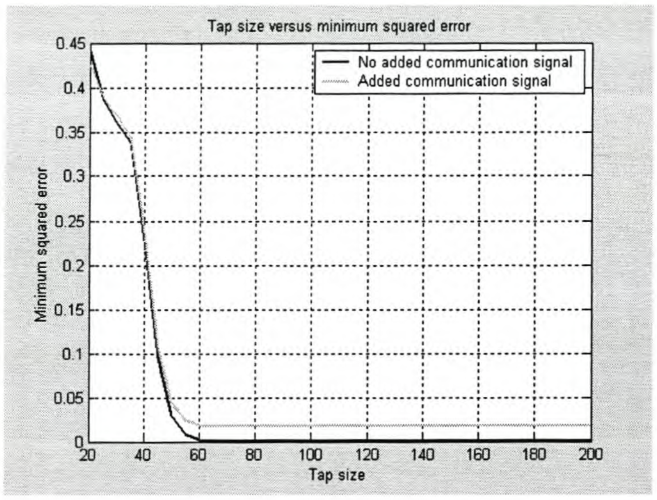


Figure 5.2 Average squared error vs. number of taps to be estimated

5.2.3. Bandwidth attenuation performance (No communication vs. communication)

This section investigated the KP algorithm bandwidth attenuation properties for when a communications signal is present versus when no communications signal is present. The bandwidth attenuation will once again be compared with the attenuated signal energy calculated from equation 3.17 of chapter 3.

The attenuation versus frequency bandwidth results attained from these simulations are shown in Figure 5.3.

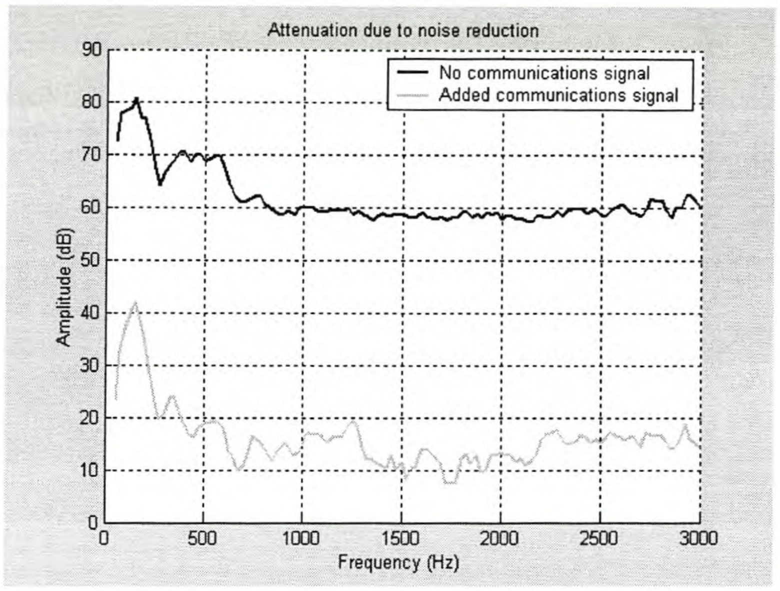


Figure 5.3 Attenuation versus bandwidth

The simulated attenuation of signal energy was calculated to be 75.1770 dB for the no communications signal case and 29.2705dB for the added communications signal case.

Once again it is clear that the addition of a communications signal greatly reduces the attenuation attainable by the KP algorithm. The LMS algorithm showed similar characteristics. These similarities can be attributed to the similar operation of the KP and LMS algorithms.

5.2.4. Convergence considerations

The convergence of the KP algorithm was evaluated according to the convergence variable as described in section 2.4 of chapter 2.

Considerations investigated were the convergence performance of the KP algorithm for a system with communications signal as apposed to one without a communications signal. The convergence variable behaviour found for the Kaczmarz projection algorithm can be seen in Figure 5.4.

It can be seen that for the added communications signal case the convergence variable has a bigger variance than for the no communications signal case. These variances were calculated to respectively be 0.0244 (communications) and 0.0042 (No communications).

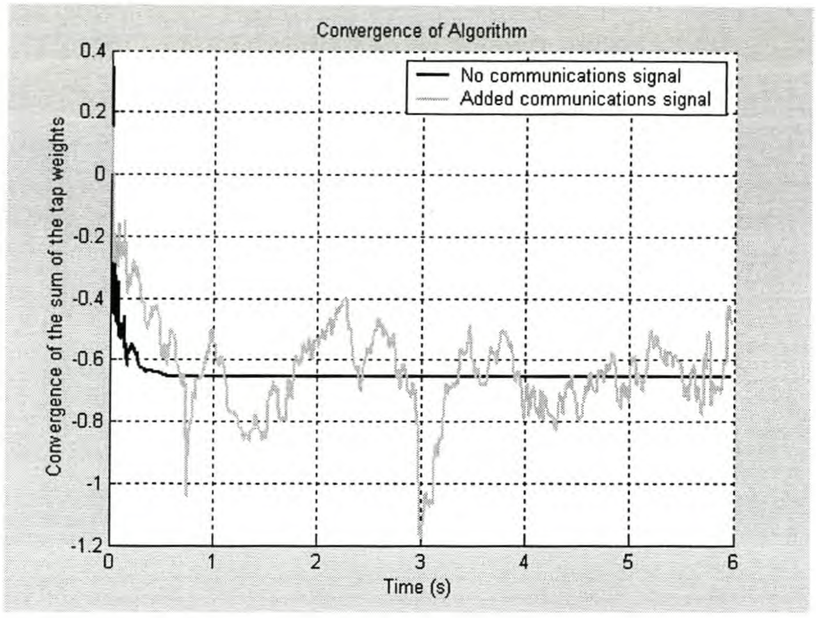


Figure 5.4 Adaptive filter convergence

For both cases the convergence variable settles to the approximate mean value of both convergence variables within 0.7 seconds. The convergence variable mean values were found to be -0.6335 (communications) and -0.6457 (No communications).

This implies that both cases are trying to find the same optimal filter since both cases tend to produce a similar step response output. The increase in variance of the convergence variable of the two cases can be seen as how much harder it is for the adaptive filter to adapt and find the optimal filter tap weights.

5.2.5. Loudspeaker and microphone effects

The influence of loudspeaker and microphone behavior once again needed some investigation for this algorithm. A simulation was run to see what effects typical loudspeaker and microphone transfer would have to the attenuation capabilities of the KP algorithm. These simulations were run with added communications speech.

To ensure the optimal operation of the KP algorithm with loudspeaker and microphone transfer, the optimal step size for this environment was found as shown in Figure 5.5. The algorithm convergence time allowed was extended to 5 seconds for these optimal step size simulations. This was done since the loudspeaker and microphone transfer inclusion slowed the convergence of the algorithms due to smaller step size values required to find a useful filter estimation.

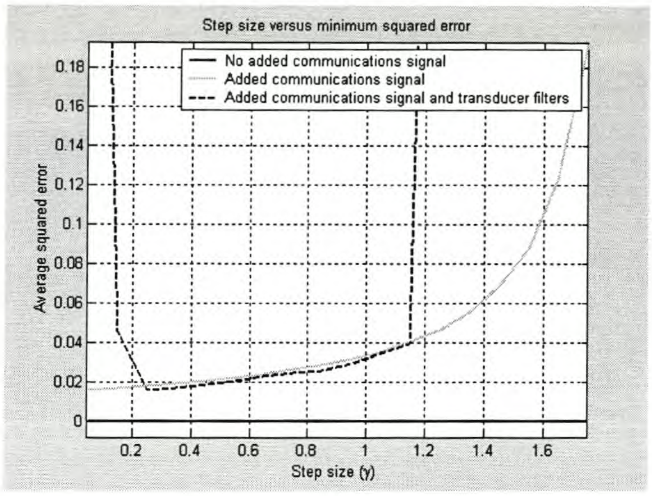


Figure 5.5 Optimal step size parameters with added loudspeaker and microphone transfer

From Figure 5.5 it can be seen that the $\gamma < 2$ boundary is corrupted by the loudspeaker and microphone filter addition. The optimal step size was chosen at half of the new boundary, $\gamma = 0.65$. This decision was made independently of the optimal step size values found for the case with no transducers.

From Figure 5.5 it becomes evident that the new adaptive filter configuration has changed the operation of the adaptive filter. Figure 5.6 shows the convergence variable and attenuation of signal energy found for the $\gamma = 0.65$ step size choice.

Figure 5.6 shows the adaptive filter's attempt to estimate a different filter as before since the convergence variable mean changes from -0.6335 to 6.9464. The loudspeaker and microphones thus change the required filter. It is also clear that the new case takes longer to converge as before.

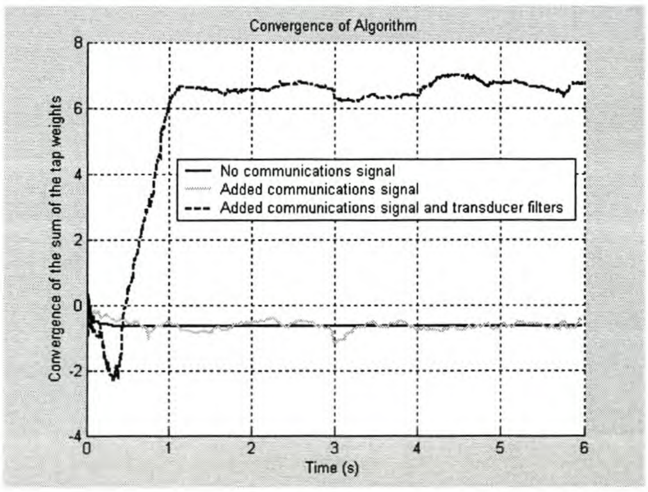


Figure 5.6 Convergence variable when a loudspeaker and microphones is added

Figure 5.7 shows the attenuation versus bandwidth performance of the added loudspeaker and microphones case in relation with previous simulations.

From Figure 5.7 the new case shows very deteriorated attenuation levels as apposed to previous simulations. The attenuation of signal energy deteriorated from 29.2705dB (No loudspeaker/microphones) to 17.1452dB for the loudspeaker and microphones case.

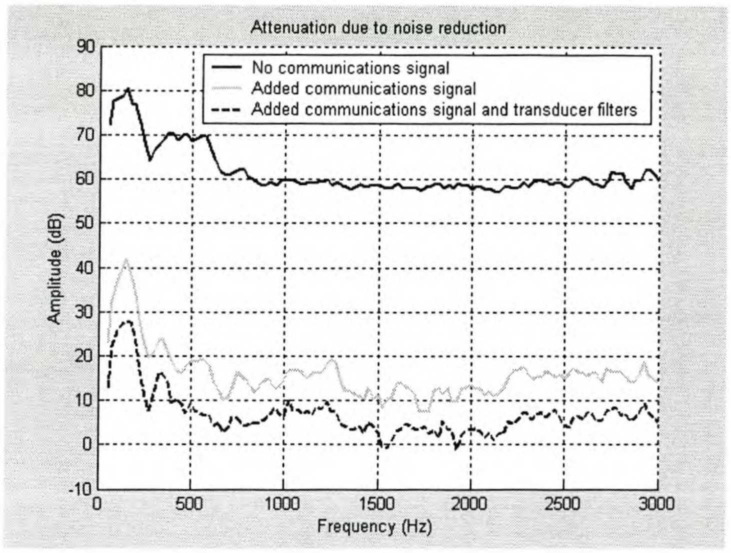


Figure 5.7 Attenuation versus bandwidth performance with loudspeaker and microphone filters

A method to improve the loudspeaker/microphone-filtered case was investigated. The filtered-x method used in chapter 3 was attempted for the KP algorithm. The filtered-x KP algorithm was derived in a similar fashion as the filtered-x LMS algorithm.

Consider that the system operation is altered by the insertion of a loudspeaker as shown in Figure 5.8.

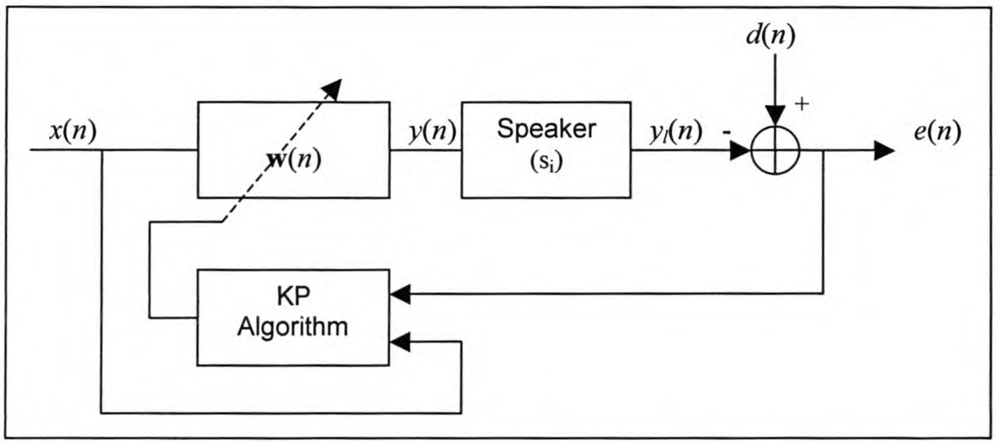


Figure 5.8 Inclusion of loudspeaker filter into model

This inclusion of a disturbance transfer must be included in the algorithm cost function as shown in equation 5.11.

$$J(\mathbf{w},t)=\frac{1}{2}\left[\mathbf{w}(n+1)-\mathbf{w}(n)\right]^T\left[\mathbf{w}(n+1)-\mathbf{w}(n)\right]+\bar{\alpha}\left[d(n)-\sum_{i=0}^{M-1}s_i\mathbf{w}^T(n+1-i)\mathbf{u}(n-i)\right] \tag{5.11}$$

The cost function derivative is found as shown in equation 5.12.

$$\frac{\partial J(n)}{\partial \mathbf{w}(n+1)}=\left[\mathbf{w}(n+1)-\mathbf{w}(n)\right]-\bar{\alpha}\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i) \tag{5.12}$$

Setting the derivative equal to zero will produce the optimal solution for $\mathbf{w}(n+1)$.

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\bar{\alpha}\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i) \tag{5.13}$$

Now to calculate the solution of the Lagrangian multiplier ($\bar{\alpha}$) we can substitute equation 5.13 into the Lagrangian constraint of equation 5.2 with added interference filter as shown below.

$$\begin{aligned}
d(n)&=\sum_{i=0}^{M-1}s_i\mathbf{w}^T(n+1-i)\mathbf{u}(n-i)\\
&=\sum_{i=0}^{M-1}s_i\left[\mathbf{w}(n-i)+\bar{\alpha}\sum_{j=0}^{M-1}s_j\mathbf{u}(n-j-i)\right]^T\mathbf{u}(n-i)\\
&=\sum_{i=0}^{M-1}s_i\mathbf{w}^T(n-i)\mathbf{u}(n-i)+\bar{\alpha}\left[\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right]^T\left[\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right]\\
&=\sum_{i=0}^{M-1}s_i\mathbf{w}^T(n-i)\mathbf{u}(n-i)+\bar{\alpha}\left\|\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right\|^2
\end{aligned}$$

Now

$$\bar{\alpha}=\frac{d(n)-\sum_{i=0}^{M-1}s_i\mathbf{w}^T(n-i)\mathbf{u}(n-i)}{\left\|\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right\|^2}=\frac{e(n)}{\left\|\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right\|^2}$$

By adding the step size (γ) and stability parameter (α), the filtered-x KP algorithm can be formulated as shown in equation 5.14.

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\frac{\gamma}{\alpha+\left\|\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right\|^2}\left(\sum_{i=0}^{M-1}s_i\mathbf{u}(n-i)\right)\left[d(n)-\sum_{i=0}^{M-1}s_i\mathbf{w}^T(n-i)\mathbf{u}(n-i)\right] \quad 5.14$$

Once again the microphone filters can be ignored since they already are in a filtered-x configuration with the adaptive filter. The new configuration resembles the LMS filtered-x topology as shown in Figure 3.12.

To ensure the optimal operation of this algorithm, the step size versus average squared error was once again established as shown in Figure 5.9.

It can be seen that the γ <2 boundary condition is restored by the filtered-x algorithm to resemble the original step size performance as shown in Figure 5.1. The optimal step size was still chosen smaller than half the step-size boundary since it is not possible to restore the KP algorithm performance totally and smaller step values tends to converge to more exact filter tap weights. γ was again chosen at 0.65.

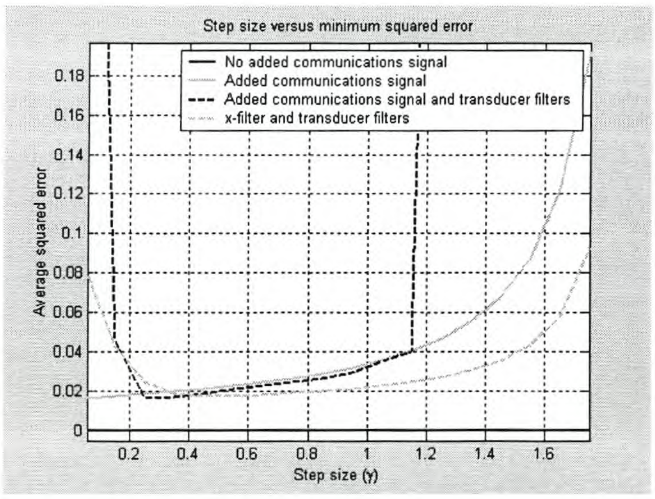


Figure 5.9 Optimal step size parameters with filtered-x loudspeaker transfer and microphones

The filtered-x KP algorithm was evaluated with added communications speech. From Figure 5.10 it was found that the filtered-x estimated filter was of the same form as the filter estimated without x-filtering since the filtered-x convergence variable mean (6.8552) resembled the convergence variable mean (6.9464) for the no-filtered-x case.

The convergence variable variance was found to have dropped from 0.0284 (No x-filtering) to 0.0069 (filtered-x). The filtered-x convergence variable preformed smoother than before and thus shows that the new filter estimation is less random.

From Figure 5.11 it can be seen that the attenuation versus bandwidth performance improved slightly due to the addition of the x-filter. This improvement is also slightly evident from the attenuation of signal energy that rose from 17.1452dB to 17.5603dB.

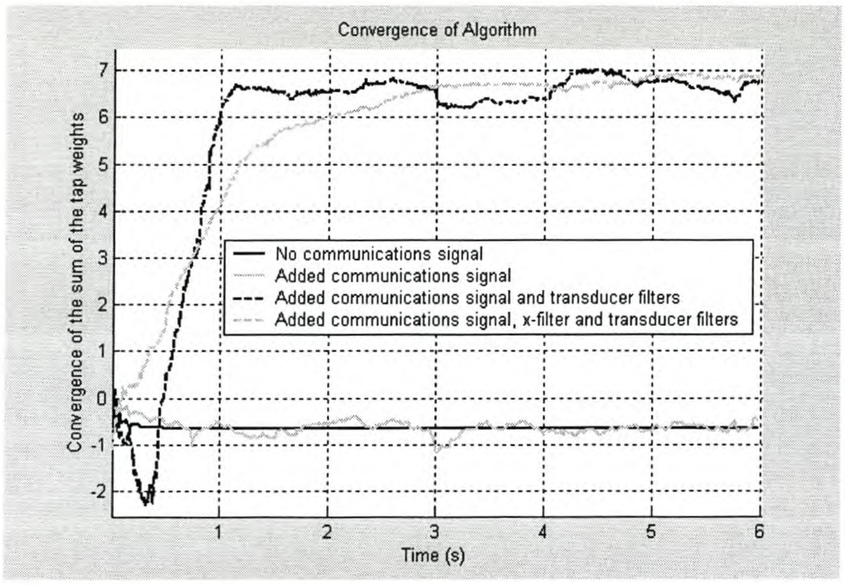


Figure 5.10 Convergence variable for filtered-x KP algorithm

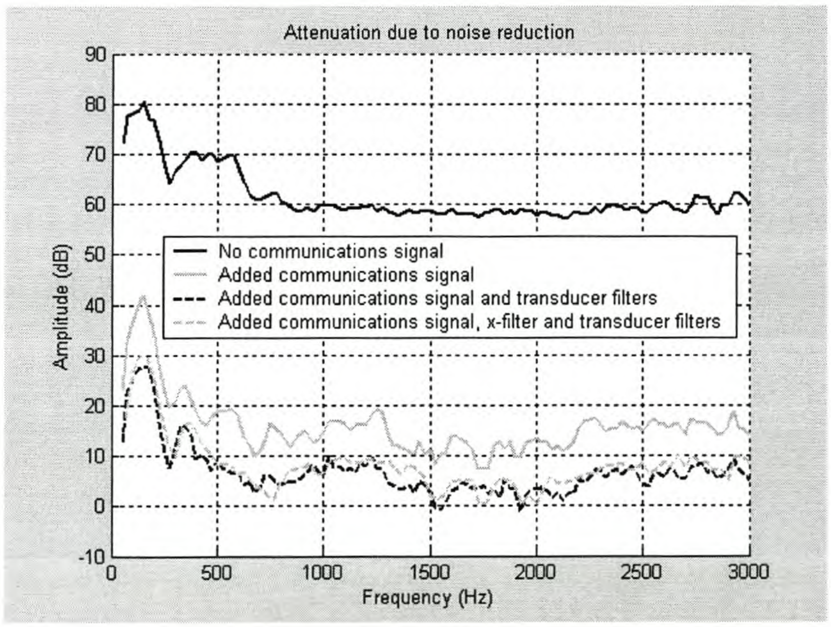


Figure 5.11 Attenuation versus bandwidth performance for filtered-x KP algorithm

The results thus show that using the filtered-x KP algorithm restores the original operational properties of the algorithm and slightly improves the ANR performance.

5.3. Kaczmarz projection algorithm conclusion

The working and performance of the Kaczmarz projection algorithm was investigated in this section.

It was found that the KP algorithm has got characteristics different from the LMS algorithm and that these characteristics are beneficiary to ANR implementation.

The data extracted from this chapter will be compared with that of other adaptive filter algorithms in chapter 6.

Chapter 6

6 Adaptive filter summary

Chapter 6 will compare the different adaptive filters that were investigated in chapter 3, 4 and 5.

A summary of each adaptive filter analysed will be given. The summary for each filter is outlined in the following sections. Each section is awarded a score relative to the findings of the different adaptive filter algorithms.

This will be done to establish the characteristics of every adaptive filter and to get an implementable adaptive filter to provide the best noise reduction performance for ANR in this specific topology.

6.1. Least Mean Square algorithm summary

Optimal implementation

Since the $\mu < 2/\lambda_{Max}$ boundary is unknown and varying according to the statistics of the noise environment, it is difficult to choose a optimal step size (μ) value.

Choosing a very small step size value is not a guarantee for good algorithm performance since this could prolong the convergence time of the algorithm. The LMS optimal implementation set-up is therefore classified as being 'complex'.

Calculation complexity

The filter tap weight update algorithm for the LMS algorithm is very simple and not calculation intensive at all. The calculation complexity is therefore classified as being 'simple'.

Estimation filter order

Since the calculation complexity of the LMS algorithm is simple, this allows the algorithm to push its effort in calculation of higher order filters for DSP implementations. The LMS algorithm can thus be classified as being a 'high order' estimation filter.

<u>Transfer estimation</u>

The transfer estimation ability of the LMS algorithm could be seen in the attenuation levels attained by the LMS algorithm when no added interferences where simulated.

For this case the LMS algorithm had to estimate a 40 zeros, 3 poles ARMA filter with a 100 zeros FIR filter. The LMS algorithm attained 75.434dB attenuation. This classifies the LMS algorithm as having a 'very good' transfer estimation ability.

<u>Interference signal rejection</u>

The ability of the LMS algorithm to keep performing well while an uncorrelated interference signal such as communications speech is present on the feedback error signal of the algorithm was found to be 'average'.

This could be seen in the drop of attenuation when a communications signal was added to the ANR LMS simulation. Attenuation levels dropped from 75.434 dB to 28.467dB.

<u>Interference filter rejection</u>

If an interference filter filters the reference input and error feedback signals of the LMS algorithm, it was found that the LMS performance dropped from 28.467dB to 16.646dB dB for specific loudspeaker and microphone filters.

The LMS algorithm does however allow the adaptation of the filtered-x LMS algorithm. Inclusion of an x-filter, improves the LMS performance from 16.646dB to 18.199dB.

The above classifies the interference filter rejection ability of the LMS algorithm as being 'average'.

<u>Filter convergence time</u>

The LMS algorithm convergence time for the best choice of step size was found to be approximately 1 second where no interference filters were present. For the more complex case where interference filters were added the convergence time increased to approximately 3 seconds. This convergence times was classified as being 'good'.

6.2. Recursive Least Squares algorithm summary

Optimal implementation

As stated before the RLS algorithm will be implemented with the assumption that the transfer of the headset is stationary. This simplifies the choice of forgetting factor. The forgetting factor will always be optimal at λ=1.

Since this is the only variable parameter that has to be chosen to implement the RLS algorithm, the optimal implementation is classified as being 'simple'.

Calculation complexity

The filter tap weight update algorithm for the RLS algorithm is much more calculation intensive than the LMS or KP algorithm. This is due to the big matrix multiplication that must take place in the algorithm. The calculation complexity is therefore classified as being 'complex'.

Estimation filter order

Since the calculation complexity of the RLS algorithm is very intensive, this limits the algorithm to only calculating lower order filters for DSP implementation.

The RLS algorithm is thus more often implemented for lower order filters. The RLS algorithm is thus classified as being a 'low order' estimation filter.

Transfer estimation

The transfer estimation ability of the RLS is once again evident in the attenuation levels attained by the RLS algorithm for when no added interferences were simulated.

For this case the RLS algorithm had to estimate a 40 zeros, 3 poles ARMA filter with a 100 zeros FIR filter. The RLS algorithm attained 62.517dB attenuation. This classifies the KP algorithm as having a ' very good' transfer estimation ability.

Interference signal rejection

The ability of the RLS algorithm to keep performing well while an uncorrelated interference signal is present on the feedback error signal of the algorithm was found to be 'very good'.

This could be seen in the reasonable small drop of attenuation when a communications signal is added to the ANR RLS simulation. Attenuation levels dropped from 62.517dB to 48.9938dB.

Interference filter rejection

If an interference filter filters the reference input and error feedback signals of the RLS algorithm, it was found that the RLS performance dropped from 48.9938dB to 6.0324dB for specific loudspeaker and microphone filters. The RLS algorithm does not allow the adaptation of a filtered-x RLS algorithm.

The convergence time for the RLS algorithm with interference filters was found to increase approximately 10 times. The above classifies the interference filter rejection ability of the RLS algorithm as being 'very poor'.

Filter convergence time

The RLS algorithm convergence time for the best choice of step size was found to be approximately 0.9 second where no interference filters were present. With added interference filters the convergence time increased to approximately 10 seconds. This convergence times were classified as being 'good (for no interference filters)'.

6.3. Kaczmarz Projection algorithm summary

Optimal implementation

Since the step size parameter is normalised according to the current input noise, this simplifies the step size choice considerably as apposed to the LMS algorithm. For this algorithm the boundary conditions are always the same for all noise environments. The optimal implementation is therefore classified as being 'average'.

Calculation complexity

The filter tap weight update algorithm for the KP algorithm is somewhat more complex than the LMS algorithm but is still not very calculation intensive. The calculation complexity is therefore classified as being 'simple'.

Estimation filter order

Since the calculation complexity of the KP algorithm is not very intensive, this allows the algorithm to push its effort in calculation of higher order filters for DSP implementations. The KP algorithm can thus also be classified as being a 'high order' estimation filter like the LMS algorithm.

Transfer estimation

The transfer estimation ability of the KP algorithm can also be seen in the attenuation levels attained by the KP algorithm for when no added interferences were simulated.

For this case the KP algorithm had to estimate a 40 zero, 3 poles ARMA filter with a 100 zeros FIR filter. The KP algorithm attained 75.1770dB attenuation. This classifies the KP algorithm also as having a 'very good' transfer estimation ability.

Interference signal rejection

The ability of the KP algorithm to keep performing well while an uncorrelated interference signal is present on the feedback error signal of the algorithm was found to be 'average'.

This could be seen in the drop of attenuation when a communications signal was added to the ANR KP simulation. Attenuation levels dropped from 75.1770 dB to 29.2705dB.

Interference filter rejection

If an interference filter filters the reference input and error feedback signals of the KP algorithm, it was found that the KP performance dropped from 29.2705dB to 17.1452dB for specific loudspeaker and microphone filters.

The KP algorithm does however allow the adaptation of the filtered-x KP algorithm. Inclusion of an x-filter, improves the KP performance from 17.1452dB to 17.5603dB. It was also found that the convergence time of the algorithm increased by approximately 2 times.

The above classifies the interference filter rejection ability of the KP algorithm as being 'good'.

Filter convergence time

The KP algorithm convergence time for the best choice of step size was found to be approximately 0.7 second where no interference filters were present. With added interference filters the convergence time increased to approximately 2 seconds. This convergence times were classified as being 'very good'.

6.4. Adaptive filter summary table

All the above adaptive filter information was combined in one table to simplify the evaluation of the filters with respect to each other.

From Table 6.1 it can be seen that the LMS algorithm has very good performance abilities but is 'complex' to configure for optimal implementation. This makes it unlikely that the LMS algorithm can be implemented as functioning optimally at all time.

The RLS algorithm on the other hand is very simple to implement but is unable to handle interference filters. Unfortunately ANR cannot be implemented without some kind of interference filters.

The adaptive filter performance results thus show that the Kaczmarz projection algorithm will be the optimal adaptive filter to implement an adaptive noise reduction system in headsets. This adaptive filter is simple to implement and shows applicable superior performance results to the LMS and RLS algorithms.

Algorithm	Optimal Implementation	Calculation complexity	Estimation filter order	Transfer estimation	Interference signal rejection	Interference filter rejection	Filter convergence time
Least **M**ean **S**quare	Complex	Simple	High order	Very good	Average	Average	Good
Recursive **L**east **S**quares	Simple	Complex	Low order	Very good	Very good	Very poor	Good (For no interference filters)
Kaczmarz **P**rojection	Average	Simple	High order	Very good	Average	Good	Very good

Table 6.1 Adaptive filter preformance summary

6.5. Filtered-X topology

The filtered-X topology is only used with the KP and LMS algorithms as shown in derivations from chapter 3 and 5.

The filtered-X topology in chapter 3 and 5 was used to try and lesson the effects caused by transducer filters in the system. It was seen that improvements could be achieved with this addition to the KP and LMS algorithms.

Simulations showed the following improvements.

	Normal operation	Filtered-X operation
LMS algorithm	16.646dB	18.199dB
KP algorithm	17.145dB	17.560dB

Table 6.2 Filtered-X attenuation improvement

From Table 6.2 it can be seen that the filtered-X operation gave a significant improvement for the LMS algorithm, but only gave a slight improvement to the KP algorithm.

Since the KP algorithm was chosen as the optimal algorithm for ANR, according to section 6.4, it was decided to exclude the filtered-x configuration for the implementation of an ANR system.

This was decided since the complexity of estimating a x-filter transfer did not justify the improvement in attenuation. The x-filter is known to be non-stationary, [4], which makes estimating an x-filter very cumbersome and unpractical for the purposes of this thesis.

6.6. Optimal adaptive filter conclusion

This chapter concludes the adaptive filter investigation to find an optimal adaptive filter for the given ANR topology. The KP algorithm was chosen for its superior characteristics to the other adaptive filters addressed.

Chapters to follow will elaborate on the DSP implementation of an ANR system that uses the Kaczmarz Projection algorithm as adaptive filter.

Chapter 7

7 Active Noise Reduction assembly

This chapter outlines the construction of the experimental ANR system with emphasis on the constraints found whilst developing the system. The performance of the constructed system is also evaluated in this chapter.

7.1. System components

The ANR system consisted of six components. These components are described in this section. Figure 7.1 shows the basic experimental set-up.

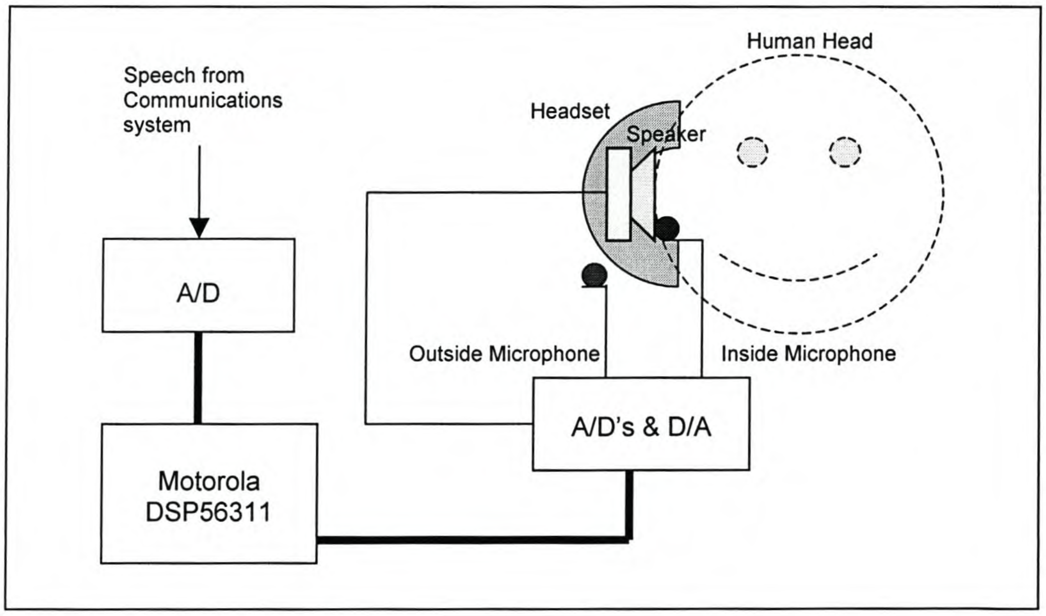


Figure 7.1 Assembly of ANR system

i.) Aircraft Headset

A commercially produced aircraft headset was used to passively dampen environmental noise. This was the same headset as analysed in section 2.2.1.1 for the simulation analysis. The ANR system was added to this headset to suppress noise that could not be minimised by passive methods.

ii.) Outside microphone

An microphone outside the headset obtains the environmental noise outside the headset. The aim of the ANR system is to adaptively filter this noise with the DSP to resemble the noise inside the headset cavity. The data from this microphone also feeds the Kaczmarz projection algorithm to enable adaptive filter updating.

The microphone chosen had the following characteristics

Part No.	430784SP
Sensitivity	17-34mV/Pa

The microphone transfer characteristics were measured according to a reference microphone with a known flat frequency response.

The reference microphone specifications are shown below.

Name and model	Sennheiser e825S
Frequency response	80Hz to 15kHz
Transducer principle	Pressure gradient receiver
Pick-up pattern	Cardioid
Sensitivity	1.5mV/Pa

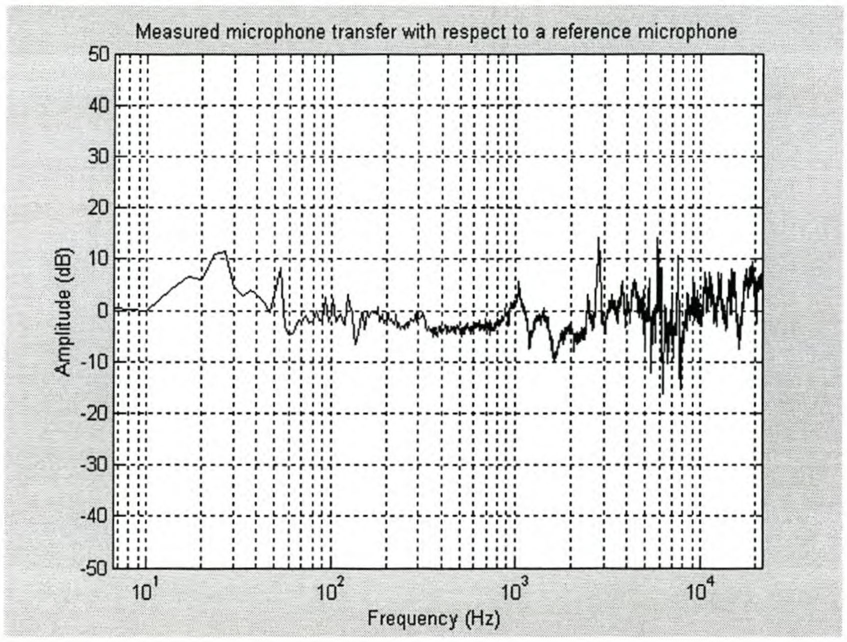


Figure 7.2 Measured microphone transfer with respect to a reference microphone

The 430784SP microphone frequency response was measured by recording the microphone output and the reference microphone output to the same zero mean white noise source. The outputs were transformed to their power density spectrums and a related decibel scale plot was generated as shown in Figure 7.2. The reference microphone was multiplied by a gain to normalise Figure 7.2 around zero decibels.

It was found that the 430784SP microphone had an equally flat response with respect to the reference microphone. This is a good attribute since the whole frequency range concerned will enjoy equal attention.

iii.) Inside microphone

A microphone was placed inside the headset cavity to enable the regulation of the uncancelled residue noise. This signal also feeds the Kaczmarz projection algorithm to enable adaptive filter updating. This microphone had the same characteristics as the outside microphone.

iv.) Correction loudspeaker

The correction loudspeaker produces the anti-noise to cancel the noise inside the headset. This loudspeaker also produces the communications signal for aircraft communication. Simulations showed that the Kaczmarz projection algorithm preformed best for a flat loudspeaker transfer.

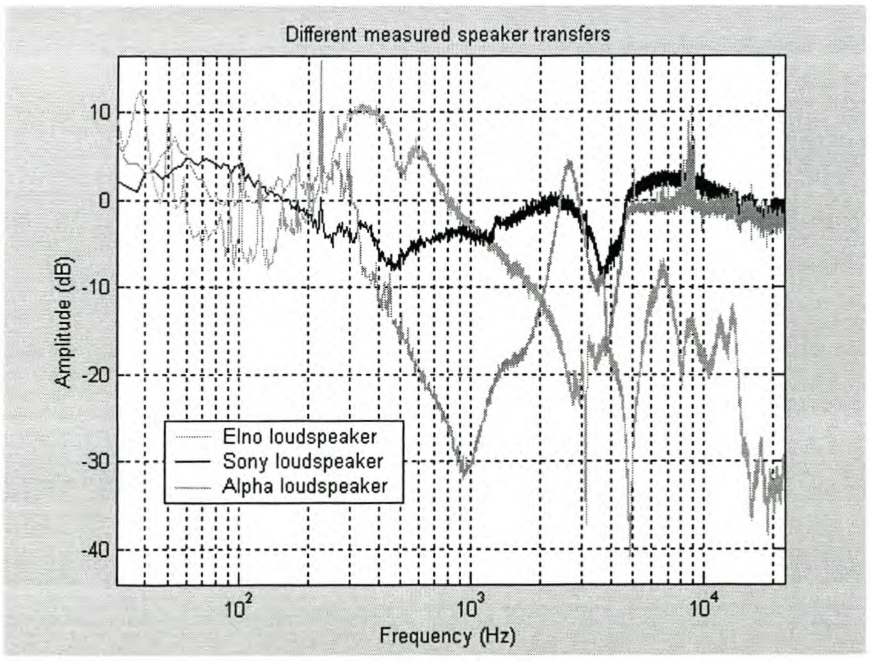


Figure 7.3 Different measured loudspeaker transfers

This can be attributed to the fact that a complex loudspeaker transfer increases the complexity of the corrective transfer that the adaptive filter has to estimate.

A number of loudspeakers were evaluated to find a loudspeaker with a flat response. Figure 7.3 shows the possible loudspeaker choices that were considered.

The loudspeaker transfers were measured by recording a zero mean white noise input to a loudspeaker and relating its power density spectrum to the power density spectrum of the measured output of the loudspeaker. The loudspeaker output was measured with the 430784SP microphone and power density calculations were averaged using a Bartlett estimation method.

The Sony loudspeaker was chosen since it had the flattest response of the loudspeakers that were evaluated. The Sony loudspeaker specifications are listed below.

Name and model	Sony MDR-7506 professional
Frequency response	10Hz to 20kHz
Impedance	nominal 63Ω
Sensitivity	106 dB/W/m

Use of this loudspeaker made the use a filtered-x Kaczmarz projection algorithm even more unnecessary.

v.) A/D and D/A conversion

The signal digital to analog (D/A) and analog to digital (A/D) conversion interface was found to be very critical to the operation of an ANR system.

Different avenues were followed to find a capable conversion solution. This section will only introduce the different A/D and D/A converters used for experimentation purposes, but will not cite the reasons for the use of each A/D and D/A converter. The reasons for using the different converters will be addressed in detail in section 7.2.

Conversion system A

The first conversion system consisted of the use of a stereo audio codec that was interfaced with the DSP56311 chip on the DSP56311EVM evaluation board.

The codec had a stereo channel input and stereo channel output. The D/A and A/D's were both serially interfaced with the enhanced serial interface ports of the DSP56311 chip. These 16 bit converters can handle sampling rates of between 8kHz to 48kHz.

Conversion system B

The second conversion system consisted of the use of individually assembled A/D and D/A converters.

This interface was configured to parallel interface two 16 bit A/D converters and one 14 bit D/A converter to the DSP56311 chip through the DSP56311EVM host port and port A. The devices were addressed one at a time with a chip select signal for each device.

The construction and DSP interface of this conversion system is explained in detail in appendix A of this document.

vi.) DSP56311EVM chip from Motorola

The Motorola DSP56311 was used for real time adaptive filter calculations and digital filtering. All digitised signal information was relayed from the A/D devices to the DSP chip for signal processing. After signal processing was completed the DSP generated an output that was relayed to the D/A device.

DSP characteristics

Resolution	24 bit
Calculations	Two's compliment fixed point
Clock speed	86 MHz

7.2. Acoustic wave propagation time delay constraint

It was found that the biggest constraint for implementing a digital broadband ANR system would be the acoustic time delay for sound to travel from the outside reference microphone to the loudspeaker inside the headset cavity.

7.2.1. Acoustic propagation time delay constraint problem statement

The shortest time delay for a sound wave to propagate from the outside microphone to where cancellation can take place at the loudspeaker inside the headset was calculated as follow.

Speed of sound (c) = 330 m/s

Shortest distance from outside microphone to loudspeaker (d) = 0.03m

$$\text{Propagation time delay} = \frac{\text{Distance (d)}}{\text{Speed of sound (c)}} = \frac{0.03}{330} = 0.00009090s \qquad 7.1$$

A similar time delay was briefly addressed in section 2.2.5. In section 2.2.5 the acoustic time delay between the two microphones were concerned to ensure effective implementation of the adaptive filter algorithms. The propagation delay problem is however different.

A practical problem that has come to light, after implementation attempts, has shown that serial A/D and D/A converters have electronic pipeline and filter delays that exceed the time it takes for sound to travel from the outside microphone to the inside loudspeaker.

This means that no signal from the outside microphone can be digitised (A/D), digitally filtered and reconstructed (D/A) in time to be in phase with respect to the noise that has propagated from the outside of the headset to the inside of the headset cavity. This phenomenon can best be explained by Figure 7.4 and Figure 7.5.

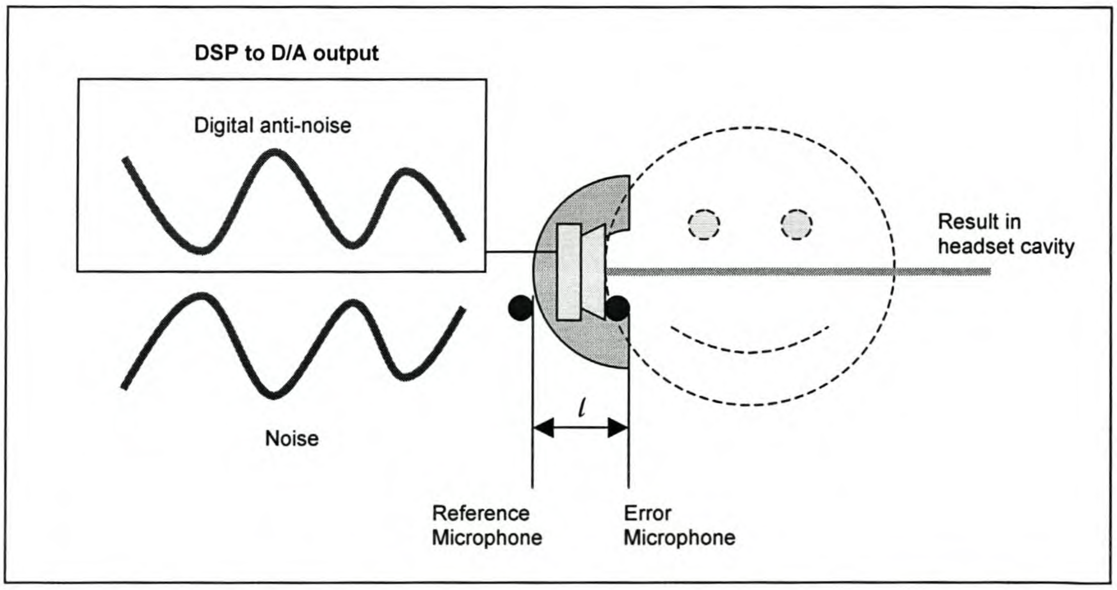


Figure 7.4 ANR operation with no delay from A/D and D/A converters

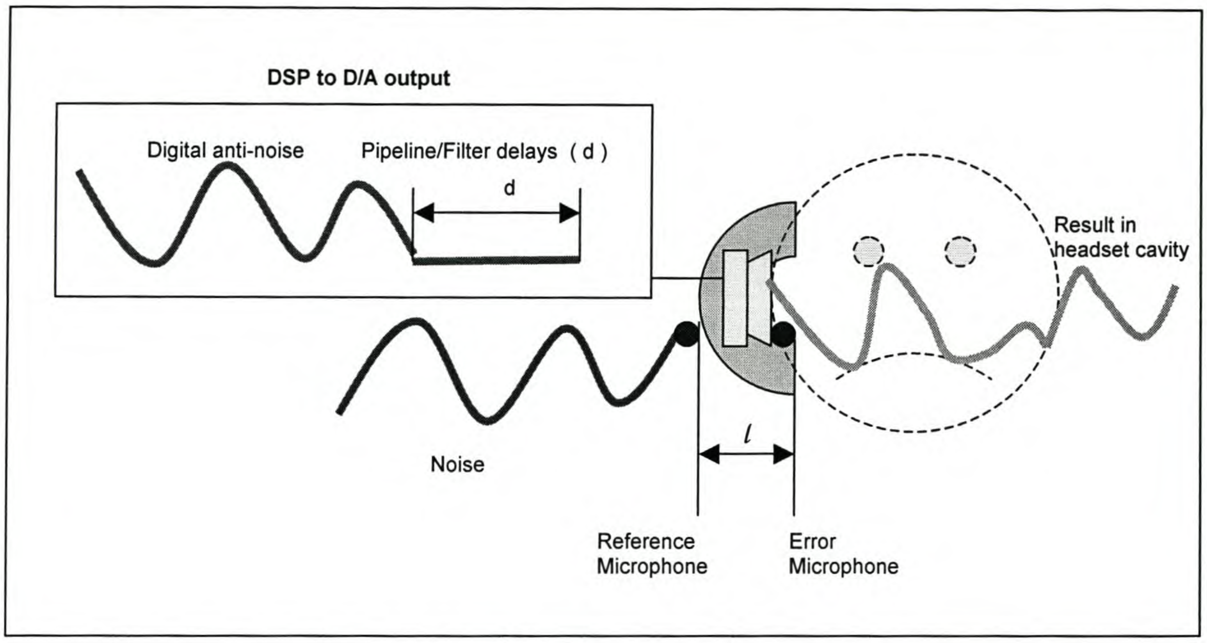


Figure 7.5 ANR operation with delays from A/D and D/A converters

This time delay is mainly due to the decimation and interpolation filters of the serial delta-sigma A/D and D/A converters as described in conversion system A. The serial pipelining of data to the DSP also added some delays.

It was measured that conversion system A had a 22-sample delay before an input to the system gave a related output. The needed sample rate to ensure that a digital output will be in time to cancel propagated noise was calculated as follow.

$$\text{Sample rate} = \frac{(\text{samples delay})}{(\text{Propagation time delay})} = \frac{22}{(0.00009090)} = 242\text{kHz}$$

This sample rate is the minimum sample rate required to ensure that the adaptive filter to be estimated remain causal. If a sample rate lower than this is used, it is required by the adaptive filter to construct a filter that can construct future filter outputs for current filter inputs, thus a non-causal filter.

This sample rate was too high for conversion system A to achieve. Increasing the sample rate of the ANR system also extended the operational bandwidth of the system drastically since delta-sigma converters adapt their anti-aliasing filters dynamically.

The increased sampling rate would also limits the calculation time required by the DSP to perform filtering and filter updating calculations at every conversion sample to such an extent that the adaptive filter calculations could not be preformed in time before the next conversion sample period. It was found imperative that another solution be found.

7.2.2. Exceptions to the propagation time delay constraint

Although the acoustic propagation time delay constraint seems a discouraging phenomenon, there is however a case where slow A/D and D/A conversion does not have a big effect on noise cancellation [2].

When the noise to be cancelled is very narrow band or tonal, the noise can be cancelled very effectively since the adaptive filter can find a filter that phase shifts the tonal anti-noise with the appropriate phase to exactly cancel the noise inside the headset cavity.

An experiment was preformed with the slower conversion system A to show this exception to the propagation delay constraint.

A 400Hz tone was generated as tonal noise. The adaptive filter implemented to cancel the noise signal was a Kaczmarz projection algorithm with no time delay compensation. The sampling rate of the system was set to 8kHz. The frequency bandwidth attenuation for this experiment is shown in Figure 7.6. No disturbance signals were included in this experiment.

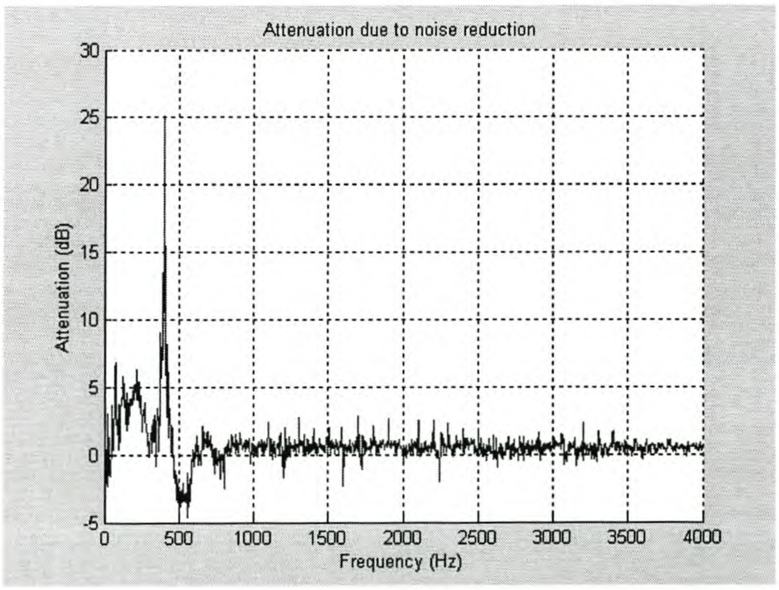


Figure 7.6 Tonal noise reduction with delayed ANR system

It must be kept in mind that the 22 sample A/D and D/A delay is 2.75ms. This is 30.253 times slower than the time it takes for sound to propagate from the outside microphone to the loudspeaker that must cancel the measured noise. The system is thus very non-causal.

It is however just required by the adaptive filter to generate a filter that phase shifts the observed 400Hz signal to be in phase with itself at the output, although the output is delayed 22 samples from the input.

It is clear from Figure 7.6 that the adaptive filter succeeds very well in estimating a phase shifting filter to cancel the tonal noise by 25.1402 dB. Some low frequency noise is also cancelled.

7.2.3. Corrective action for acoustic propagation time delay constraint

The acoustic propagation time delay constraint originates directly from filter and pipeline time delays caused by the A/D and D/A converters.

For an ideal solution, it would thus be the best to limit A/D and D/A time delays as far as possible. This was attempted by implementing new parallel data A/D and D/A converters that have no pipeline delays and fast conversion times (<10µs).

Converters with such capabilities could only be found for converters that employ successive loop approximation conversion with parallel data transfer at high sample rates. The ADS7805 A/D and the AD7538 D/A was implemented in an attempt to limit conversion delays.

The acquisition and conversion delay for the ADS7805 A/D was in the order of 8.25µs. The acquisition and conversion delay for the AD7538 D/A was in the order of 1.177µs.

The shortest time in which calculations and conversions must be completed was calculated before in equation 7.1. This time delay was calculated to be 90.9µs. This means that the system could start to require a non-causal estimated filter from the adaptive filter algorithm, if it takes longer than 90.9µs to acquire, process and output a value from the converters.

The following timing specifications are shown for conversion system B.

A/D conversion and acquisition time	(A/D 1)	8250.0 ns
	(A/D 2)	8250.0 ns

D/A conversion and acquisition time	1177.0 ns
Total time used for conversions and acquisition	17677.0 ns
Minimum DSP processing time required to implement the KP algorithm with a 86MHz DSP clock	13606.8 ns
	+
Total time required for ANR implementation with conversion system B	31283.8 ns

It is thus clear that it can be possible to solve the propagation time delay constraint. All acquisition and processing requires 31.283µs, this is well below the minimum acoustic propagation delay time of 90.9µs. This will ensure that the system works causally at all times.

This time requirement however limits the maximum sampling rate at which the system can function. The maximum sampling rate is 1/31.2838µs = 31.96kHz.

7.3. A/D dc offset problem

7.3.1. A/D dc offset problem statement

From Figure 7.2 and Figure 7.3 it might seem that the microphones and loudspeaker transfer had close to unity transfer for dc values. This is not true since the transducer measurements were made with zero mean Gausian white noise and can thus not show any dc transfer components.

A dc signal on a transducer constitutes a constant acoustic pressure. No transducer can transmit or sense constant acoustic pressure since transducer signals are ac signals or created from ac signals.

It is thus clear that no dc compensation can be accommodated in this noise cancellation system. This alone is not a problem since the microphones do not sense any acoustic dc values and the loudspeaker cannot create constant pressure waves to compensate for acoustic dc signals.

When the transducer signals are converted to digital signals it does however happen that untuned A/D converters sometime add small dc values to converted analog acquisitions. Figure 7.7 show the dc offset values measured for the two channels of conversion system A.

The A/D scales the input voltage to a representative hexadecimal value with a decimal value of between –1 and 1.

Channel 2 is clearly out of tune and shows an average dc offset of approximately 2.2x10^{-4} digital units. But what effect will this uncorrelated dc offset have on the adaptive filter calculations?

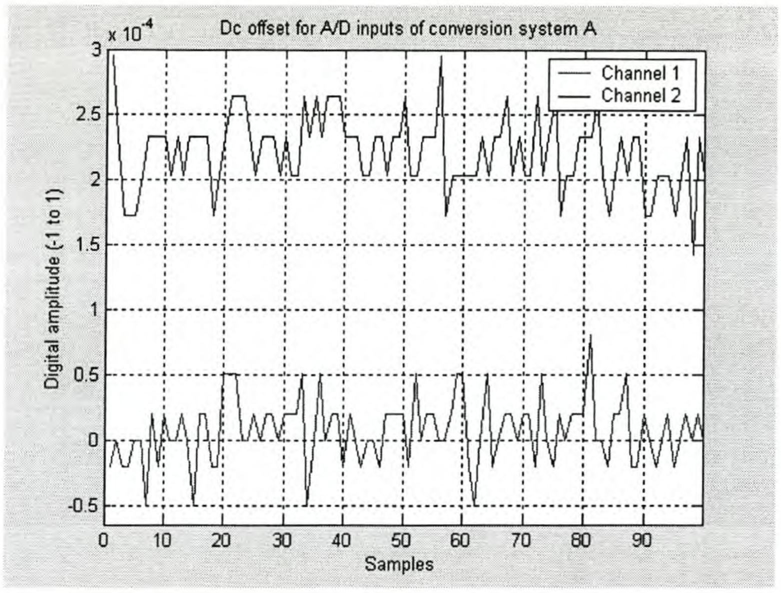


Figure 7.7 Dc offset for analog grounded A/D converters of conversion system A

From experimentation a problem was experienced for calculated adaptive filter coefficients that would reach a certain desired value and would gradually start to diverge from the desired filter coefficients.

This algorithm problem only occurred were a reference and error signal was taken from digitised microphone signals. It was found that the digitised signals had unwanted DC offsets added by the A/D converters.

To understand what happens to the adaptive filter algorithm when uncorrelated dc values are added to the microphone signals, a short derivation can be shown.

Lets consider the simple LMS algorithm to limit the calculation complexity of the example. The behaviour of this calculation will be similar for the Kaczmarz projection algorithm except that the step size value will be normalised to the noise environment. The LMS algorithm for filter coefficient 0 is shown below.

$$w_0(n+1) = w_0(n) + \mu u_0(n)e(n) \tag{7.2}$$

Consider that a and b are uncorrelated dc offset values produced by untuned A/D converters. Equation 7.2 changes as shown below.

$$\begin{aligned} w_0(n+1) &= w_0(n) + \mu\left[u_0(n)+a\right]\left[e(n)+b\right] \\ &= w_0(n) + \mu\left[u_0(n)e(n) + ae(n) + bu_0(n) + ab\right] \\ &= w_0(n) + \mu u_0(n)e(n) + \left[\mu ae(n) + \mu bu_0(n) + \mu ab\right] \end{aligned} \tag{7.3}$$

It can be seen from equation 7.3 that three new terms are added to the algorithm.

It can be argued that the first term, $\mu ae(n)$, will become very small as the algorithm converges to a zero error and is thus negligible. If the algorithm however diverges, this term will help the algorithm diverge even further.

In term two $u_0(n)$ is uncorrelated to b. This means that no estimated filter output can help cancel the effect the value of b has in this term. This term adds an unwanted step to the adaptive algorithm and the term will have a different effect for each filter coefficient calculated depending on the related value of $u_0(n)$ for that sample.

The third term, μab, is another term that cannot be altered by any attempt of the adaptive filter output. This term will add an unchanging constant value to each calculation of the filter coefficient updates. This will cause the filter coefficients to drift from their desired value for any small uncorrelated dc offset.

It is thus essential to eliminate any uncorrelated dc offsets from any signals used by the DSP from the A/D converters.

7.3.2. Corrective action for the A/D dc offset problem

The dc-offset problem is solved by filtering any dc from the A/D signals inside the DSP before any signal processing is preformed. This filtering does however degrade the adaptive filter algorithm performance somewhat since a phase differences is added by the dc filters. This is however a tradeoff that must be taken since the adaptive filter stability is essential to the operation of the system.

The filter designed to block all dc was a single pole single zero filter. The zero was place at z=1 to ensure all dc is removed. The pole was placed close to the zero to ensure a flat transfer over the rest of the frequency bandwidth. The designed filter transfer function is shown in equation 7.4.

$$F(z)=\frac{z-1}{z-0.98} \qquad 7.4$$

A plot of the frequency response of the dc block filter is shown in Figure 7.8.

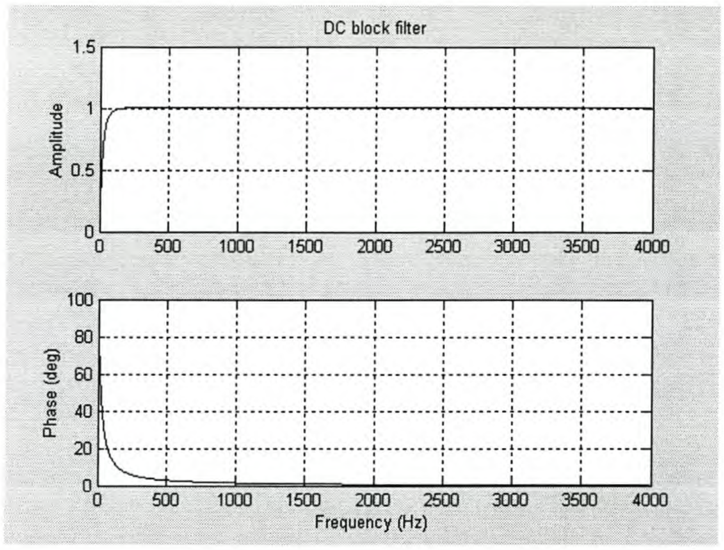


Figure 7.8 DC block filter transfer function

This filter ensured adaptive filter algorithm stability. The A/D DC offset problem only occurred with the use of conversion system A. Conversion system B showed no added DC values to digitised analog values.

7.4. <u>Conclusion to the active noise reduction assembly</u>

This chapter considered the components that make up the experimental ANR assembly, the implementation problems experienced whilst constructing this system and the respective solutions to these problems.

It was found from this chapter that an experimental system could be constructed that operates in fairly the same way as the simulated system of chapter 2. This makes the simulations for finding an optimal adaptive filter configuration (chapter 3,4,5) applicable to this practical system implementation. Chapter 8 will show the experimental set-up and performance measurements for the practical ANR system.

Chapter 8

8 Measurements

This chapter explains the experimental set-up made to perform noise reduction measurements of the implemented system. The optimal configuration of the KP algorithm in this system is also explained and performance measurements are shown.

8.1. Measurement setup

The attenuation measurements were conducted according to the ANSI S3.19 [22] standard.

For this standard the experimental headset was placed on an acoustic test fixture (artificial head). The artificial head dimensions and construction material coincided with ANSI S3.19. The dimensions of the test fixture are shown in appendix B.

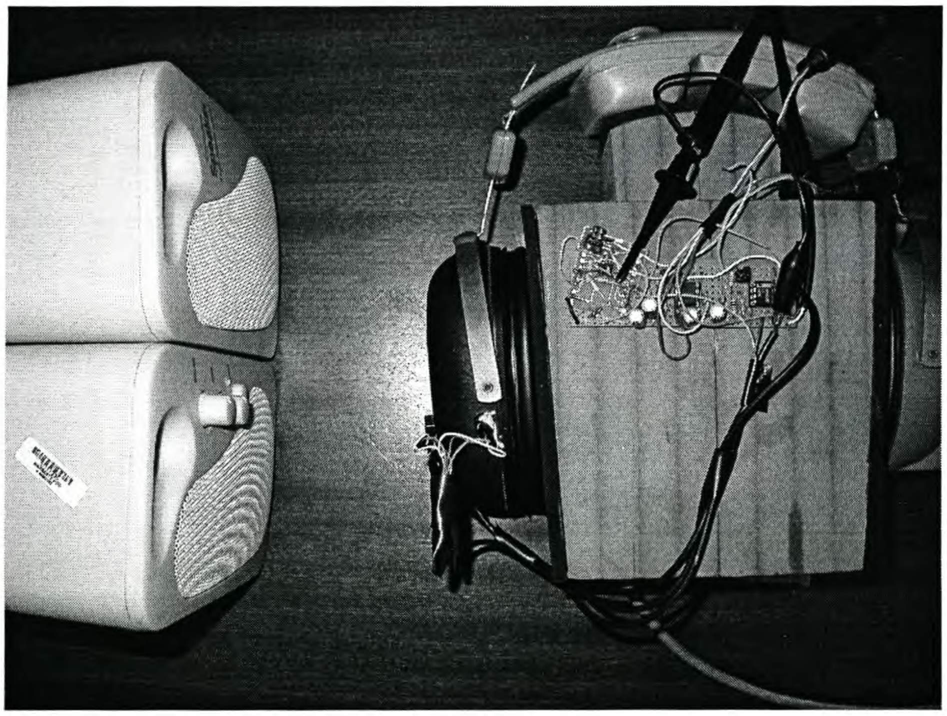


Figure 8.1 Measurement configuration

The test fixture with the headset under test was placed close to 2 PC loudspeakers that would generate 48kHz noise from a PC soundcard. Figure 8.1 shows this set-up.

The measurement set-up was surrounded by anechoic wedges to shield the set-up from external noise and to ensure that only directional noise propagated to the tests fixture. This set-up configuration can be seen in Figure 8.2.

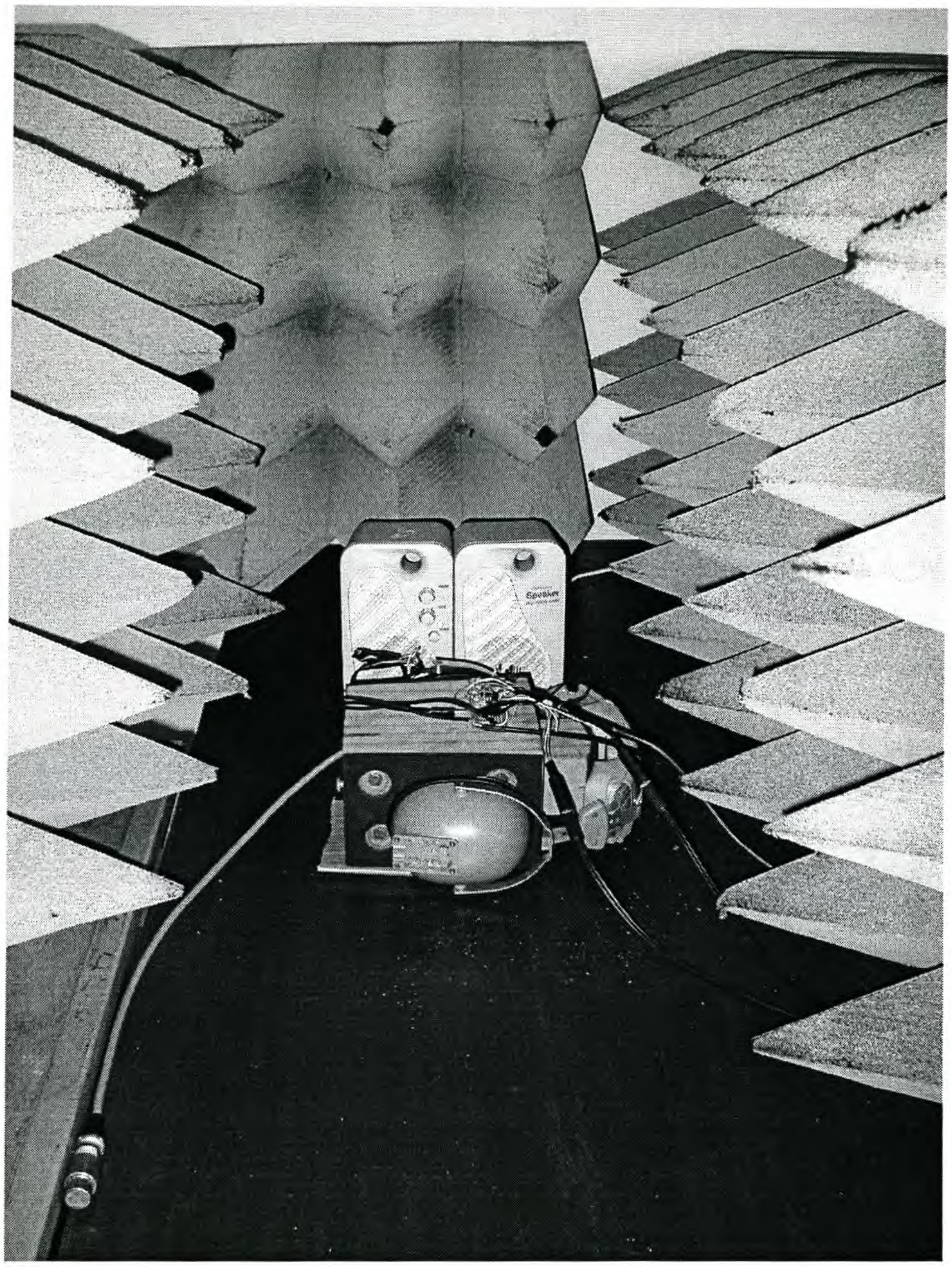


Figure 8.2 External noise shielding and directionality procurement

8.2. Loudspeaker to reference microphone isolation

It is appropriate at this stage to elaborate on the possibility that sound could couple back from the anti-noise loudspeaker to the outside reference microphone. Two measurements were made in this respect with the headset mounted on the artificial head as shown in Figure 8.1.

White noise was transmitted from the loudspeaker inside the headset and the isolation of the headset cavity was measured by relating a recording of the noise at the error microphone to that of noise at the reference microphone.

Different measurement strategies exist to determine system transfer from two measurements like these. One of these strategies was used for previous measurements as mentioned in section 2.2.1.1.

Two basic methods exist. The first method relates the cross correlation of the two signals (input and output) to the correlation of the input signal with itself to estimate the transfer in question. This method is useful since a transfer magnitude and phase plot can be obtained from the two signals. The two signals must however be recorded simultaneously. A simple implementation of the Transfer Function Estimate routine in Matlab can then derive this transfer.

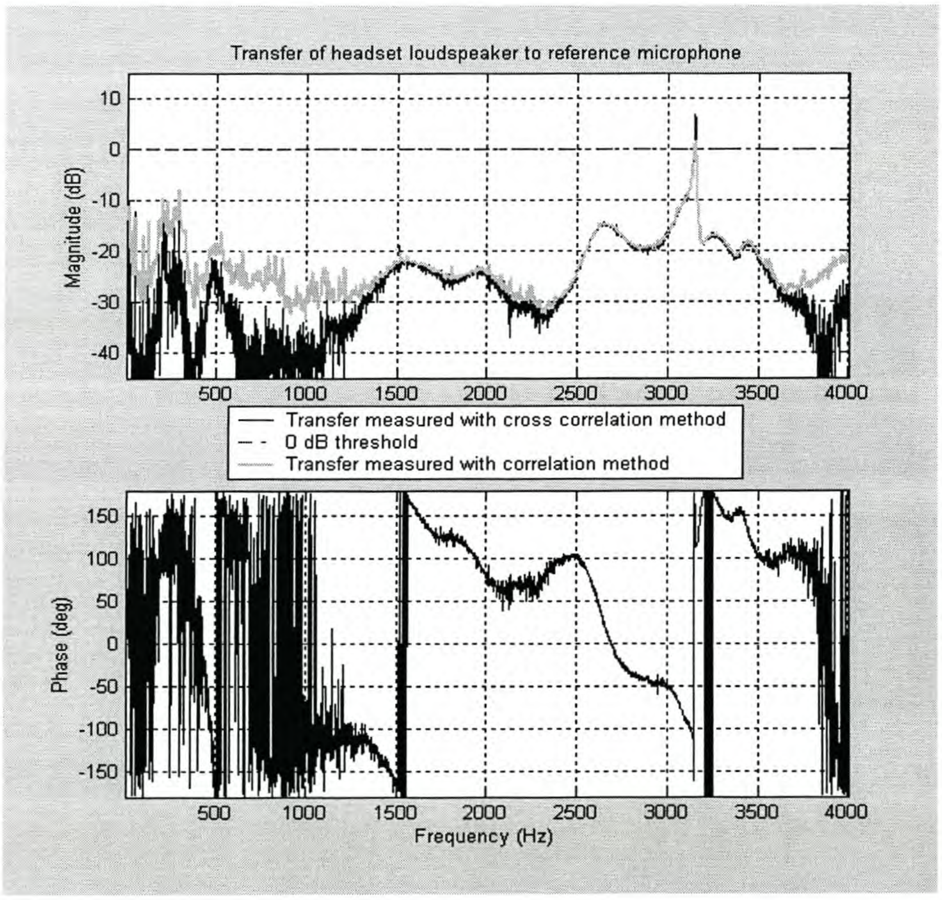


Figure 8.3 Isolation of reference microphone from anti-noise loudspeaker

The second method relates the correlation of the output signal with itself (also known as the power density spectrum) to the correlation of the output signal with itself. The power density spectrum of the output signal is thus related to the power density spectrum of the input signal. This method was used for the measurements of section 2.2.1.1. For this method the signal recordings do not have to be simultaneous.

Both these methods were used to estimate the transfer in question. The spectral isolation of the reference microphone to anti-noise loudspeaker is shown in Figure 8.3.

The two transfer measurement strategies used to generate Figure 8.3 shows that the cross correlation method gives a very rough approximation of the transfer while the correlation method gives a smoother, more defined transfer estimation. Both methods used the same amount of data.

It is for this reason that the correlation method was employed to relate all transfers and attenuation measurements. The phase data is however lost with this method but is of no consequence concerning attenuation measurements and a suitable trade-off to gain better defined magnitude transfer estimates as used in section 2.2.1.

From Figure 8.3 it can be seen that almost the whole spectrum is suppressed by approximately 20dB. There is however a tonal frequency of 3.170kHz that is not suppressed. This tonal frequency might have been generated by a prominent frequency in the ambient noise since these measurements were not conducted in a sound proof room.

It must however be kept in mind that the ANR system will not have to generate large amount of anti-noise in the 3.17kHz range since the passive attenuation of the headset already limits the bulk of noise at this bandwidth. The ANR system is also limited to optimal operation at 2.6kHz by the A/D anti-aliasing filters of the digital system. These two criteria make it highly unlikely that high levels of anti-noise power will be transmitted by the loudspeaker at this bandwidth. Communications speech levels to the loudspeaker are also limited to less than 3kHz, as required by military specifications. Since limited levels of power are transmitted by the loudspeaker at this bandwidth, the risk that bad isolation at 3.17kHz will hinder the performance of the system is very unlikely. It is also stated by [19] that this transfer of the loudspeaker to external headset environment is negligible. This is also evident from the performance measurements of section 8.3.1. The attenuation performance of the system will be shown in Figure 8.9.

8.3. Optimal ANR algorithm configuration

The following strategy was followed to ensure that the optimal algorithm chosen in chapter 6 (Kaczmarz Projection algorithm) was configured optimally for the dimensions of the headset. Three optimisation variables were identified. They were the update step size of the Kaczmarz Projection algorithm, the sampling rate of the system and the algorithm update delay.

The sample rate and algorithm update delay both relate to section 2.2.5. In section 2.2.5 it was stated that the sample rate should be chosen according to the distance between the microphones. This was only applicable for the simulations. It was found to be impractical in the real system since it was unknown if sound propagates directly from the one microphone to the other, or follows a longer route through the earmuff seal.

It is for this reason that sampling rate and algorithm update delay will be handled as separate variables that are related to each other by the path length of sound from the outside to inside microphone.

8.3.1. Optimal algorithm step size

From chapter 5, the step size can be chosen between 0 and 2. It was seen that the step size variable must be chosen according to two considerations.

The first consideration involves the complexity of the transfer function to be estimated. The more complex the system, the smaller steps must be taken to ensure that the bottom of the error performance surface is found. For the experimental system in question, it was expected that a very small step size would be required since the system is very complex due to the addition of transducers and anti-aliasing filters.

The second consideration is the convergence time in which conversion must take place. Large step sizes will ensure fast conversion, whilst small step sizes will take long to converge.

It can be seen that the above-mentioned properties appose each other and an appropriate resolution must be found.

Experiments were preformed for four different sampling rates at seven different step sizes and a number of different algorithm update delays. All experiments allowed a convergence time of 90 seconds.

The experiment results for the four different frequencies can be seen in Figure 8.4.

The results of these experiments show that the system has the same expected behavior as found in the simulations of chapter 5 (Figure 5.6). The Kaczmarz projection algorithm can thus be configured to perform optimally in the same manner as was done in the simulated environment.

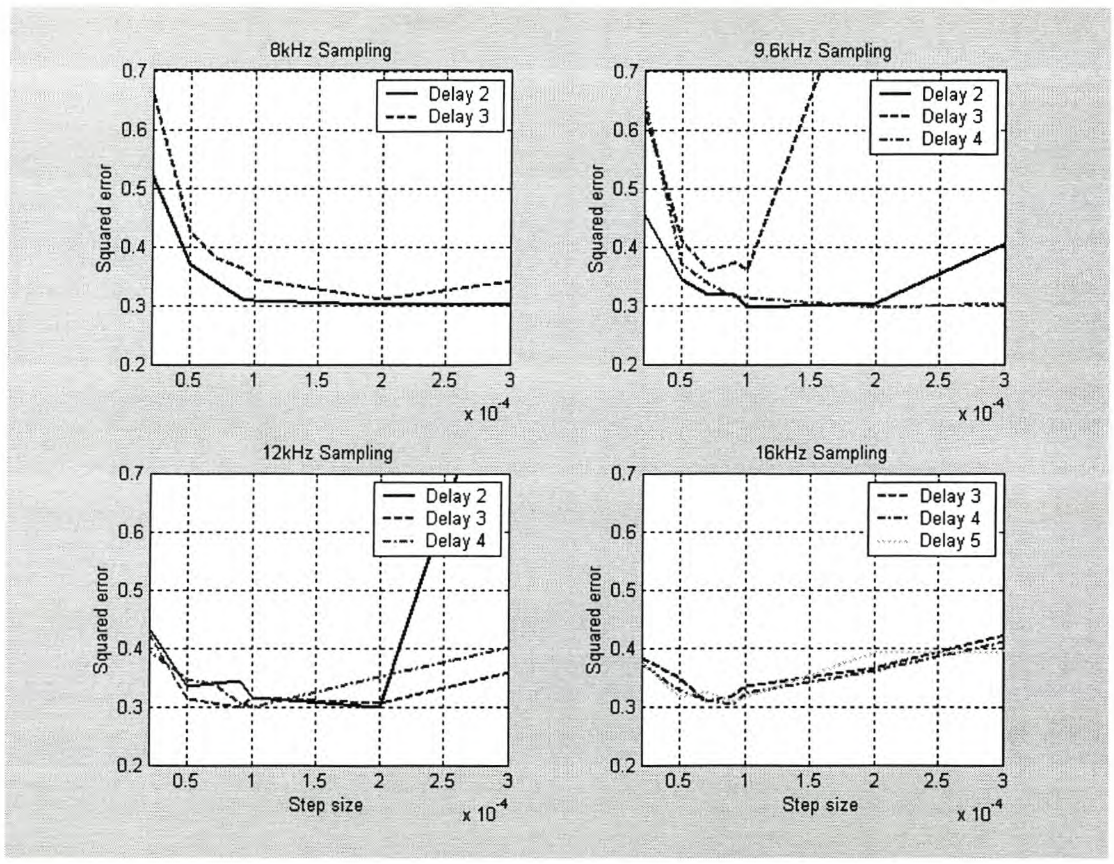


Figure 8.4 Normalised squared error versus step size and algorithm update delay

Once again the degrading squared error for small step sizes is not due to the adaptive filter performance but due to the limitation set on the convergence time of the algorithm.

From Figure 8.4, the optimal step size was chosen at 1×10^{-4}. This step size works well for all sampling rates and ensures the best estimated filter for a convergence time of 90 seconds.

8.3.2. Algorithm update delay

The optimal algorithm coefficient update delay is directly proportional to the distance that sound travels from the outside microphone to the inside microphone of the headset.

To find the optimal update delay for different sampling rates seemed a difficult task but combining the data from different sampling rates measurements made it possible to estimate the distance sound travels between the microphones and thus simplifies choosing a appropriate update delay for any sample rate.

A few experimental measurements were taken at different sample rates and a step size of $1x10^{-4}$, as previously chosen. The algorithm convergence time was once again limited at 90 seconds. The data at different update delays can be seen as shown in Figure 8.5.

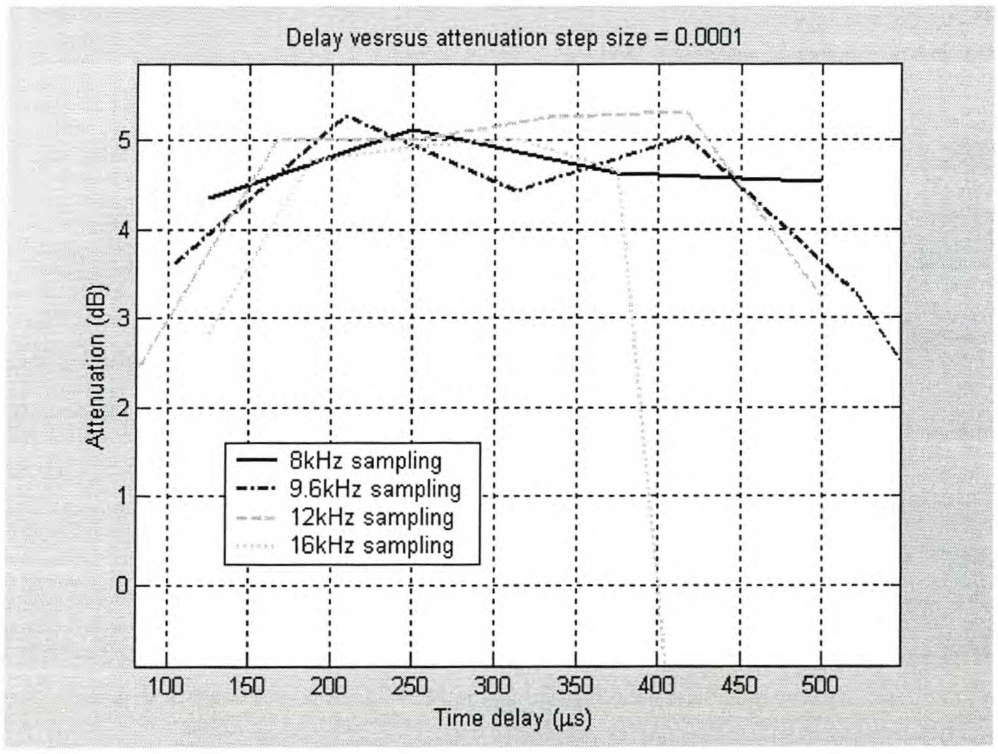


Figure 8.5 Different update delays versus attenuation

From Figure 8.5 it can be seen that the adaptive filter algorithm provides best attenuation for update delays of between 200 to 350µs. This means that the time it takes for sound to travel from the inside microphone to the outside microphone is approximately 200 to 350µs. This would imply that the path followed by sound from the inside to outside microphone varies between 6.6cm to 11.6cm for the shortest path to the longest path.

Any algorithm coefficient update delay that falls within this time frame will ensure optimal algorithm performance. Table 8.1 shows a few sampling rates with appropriate update delays.

Sampling Rate	Update delay	Time delay
8kHz	2	250μs
9.6kHz	3	312μs
12kHz	3	250μs
	4	333μs
16kHz	4	250μs
	5	312μs

Table 8.1 Appropriate update delays for different sampling rates

8.3.3. Sampling rate

Throughout this thesis many considerations were made to choose an appropriate sampling rate. Since only conversion system B allows broadband attenuation, it was decided to perform the system performance measurements using this conversion system. From section 7.1.3 it was established that the maximum possible sample rate for operation with conversion system B is 31.96kHz. No sampling rate above this boundary is thus possible.

Since the required bandwidth of attenuation is 3kHz, the minimum sampling rate cannot be lower than 6kHz, due to the nyquist limitation. Experiments showed that the attenuation due to the adaptive filter is lower at higher sample rates. This phenomenon is shown in Figure 8.6.

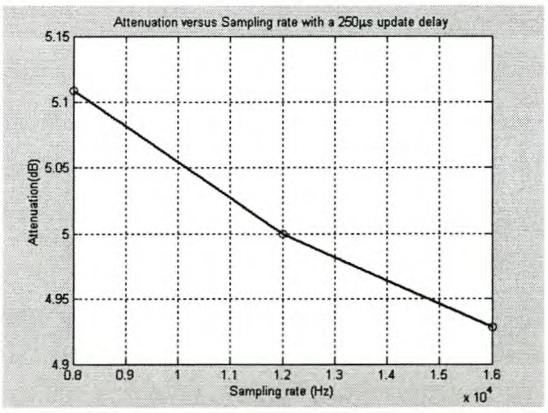


Figure 8.6 Attenuation versus sampling rate with a 250μs update delay and 90 second convergence limitation

This can be attributed to the fact that the adaptive filter order stays the same for all sampling rates. This can be explained by thinking of the z-transform unit circle.

For the same number of poles and zeros, at lower sample rates, poles and zeros inside the unit circle have more influence on the entire frequency bandwidth than for higher sample rates.

An adaptive filter can thus estimate a more accurate filter for low sample rates than high sample rates given that the filter order stays constant for all sampling rates.

It would thus be most appropriate to choose the lowest sample rate with the highest possible order adaptive filter to estimate a headset transfer filter for that bandwidth. The optimal sampling rate was chosen at 8kHz.

8.3.4. Conclusion of optimal ANR algorithm configuration

In conclusion it was found that the optimal algorithm configuration for this headset and operational bandwidth could be summarised as shown in Table 8.2.

Parameter	Value
Algorithm step size	1×10^{-4}
Sample rate	8kHz
Algorithm update delay	2 samples (250μs)
Adaptive filter order	100 (maximum)

Table 8.2 Optimal ANR algorithm configuration

All performance measurements were preformed with this algorithm configuration. The DSP assembler program code for this implementation is shown in Appendix D of this thesis.

8.4. System performance measurements

A number of performance measurements were investigated. They were the spectral attenuation of noise, A-weighted attenuation of noise, energy attenuation of noise, the effect of an added interference signal on the system attenuation performance and the directionality of noise cancellation in the headpiece.

8.4.1. Spectral, A-weighted and energy attenuation

It is appropriate at this stage to investigate A-weighting sound levels. A-weighting is a commonly used measure of sound level and approximates the 40-phon equal–loudness-level contour of the 1933 Fletcher-Munson data and the 10-phon contour of the Robinson-Dadson data [36]. A-weighting was preformed by filtering the recorded noise signals through a filter that possesses the A-weighting characteristics as found in Table 12.1 of [36]. The response of the used A-weighting filter is shown in Figure 8.7.

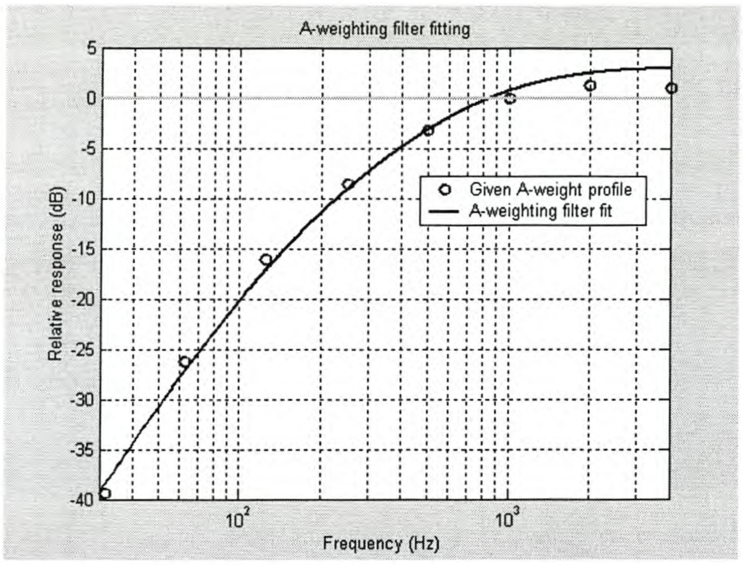


Figure 8.7 A-Weighting filter response

Since the A-weighting curve of [36] is an approximation from the 1933 Fletcher-Munson and more resent Robinson-Dadson data, the small variation between the A-weighted filter fit and given A-weighting data of [36] is given little consideration and is still seen as a good approximation of the true A-weighting curve. This A-weighting filter is only applicable for signals recorded at 8kHz. The filter transfer function is given in equation 8.1.

$$F(z) = \frac{z^2 - 2z + 1}{z^2 - 1.4z + 0.45} \qquad 8.1$$

Two recordings of the noise levels inside the headset cavity were made to evaluate the attenuation of the ANR system. The first recording was made with the ANR system not operating. The second recording was made with the ANR system in operation and after the adaptive filter algorithm had converged.

The two recordings of noise were A-weighed and the power density spectra of the signals were calculated. The A-weighed power density spectra of the signals are shown in Figure 8.8.

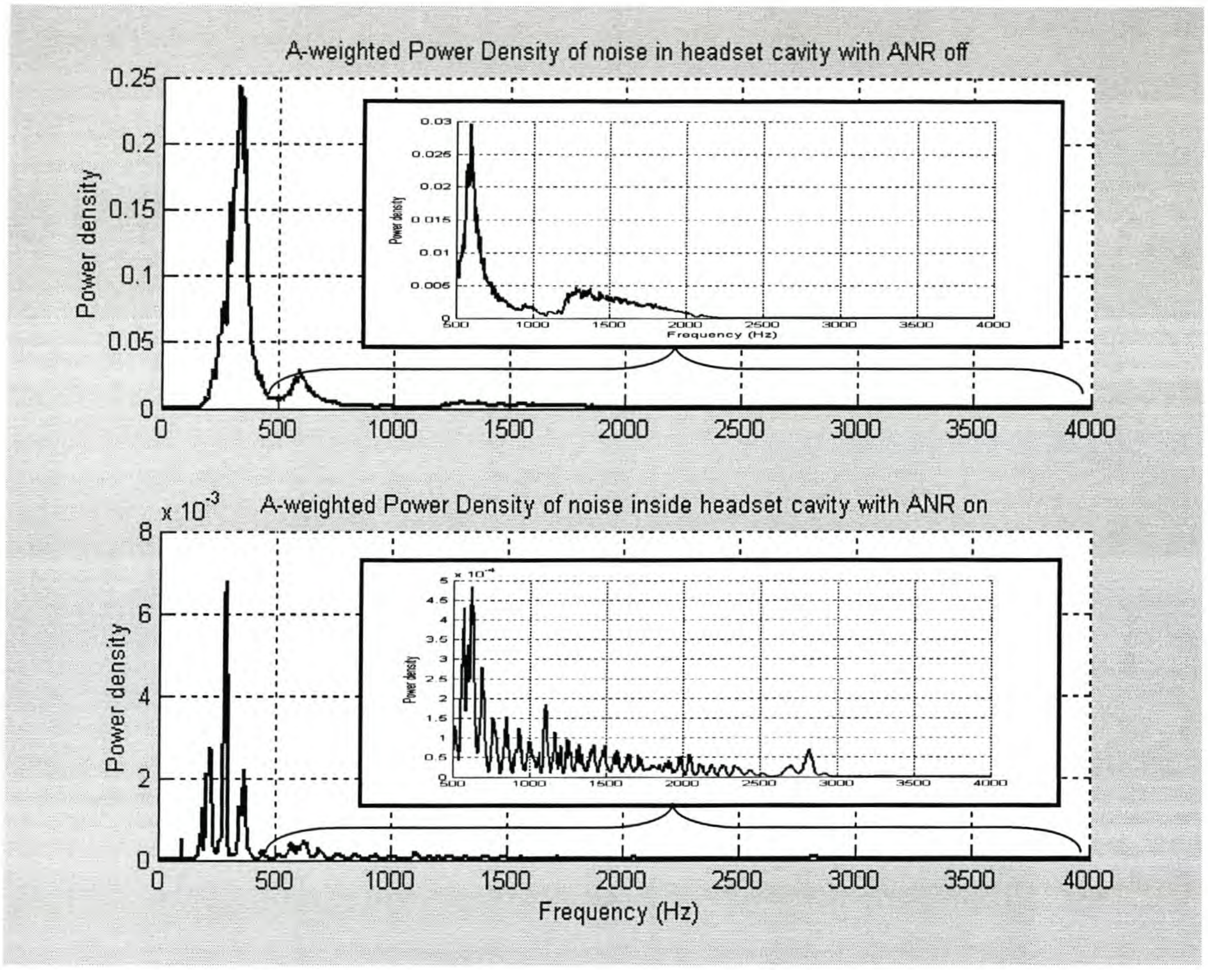


Figure 8.8 A-weighted power density spectra of noise before and after ANR

From Figure 8.8 it is clear that very little noise energy is present inside the headset cavity at the higher frequencies. It is however visible that noise energy does exist up to approximately 3kHz, as can be seen in the zoomed in figure of the A-weighted power density spectrum of noise in the headset cavity with the ANR system on. This shows that attenuation measurement due to the addition of the ANR system might be unrealistic above 3kHz since a unknown amount of energy is present in this bandwidth to perform the measurement.

A spectral attenuation plot was generated from the above mentioned power density spectra to show the affectivity of this ANR system over the bandwidth limited by the sampling rate of the system. This figure is shown in Figure 8.9.

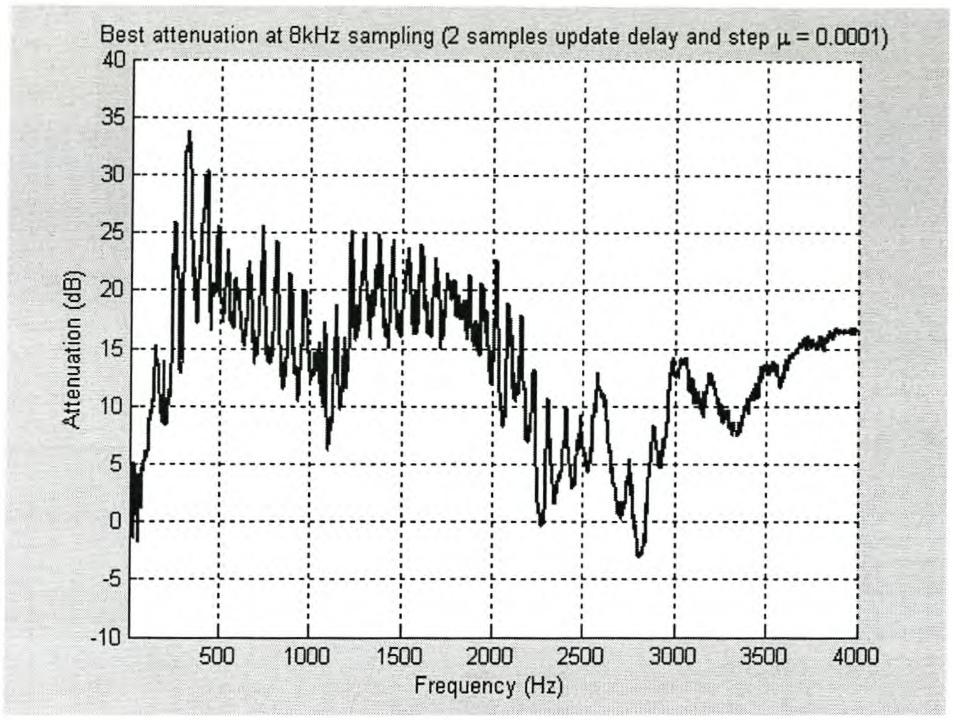


Figure 8.9 Measured spectral attenuation due to ANR

At first glance Figure 8.9 shows that the ANR system produces significant attenuation in the 250Hz to 2.2kHz range. Above this bandwidth the anti-aliasing filter complicates attenuation since the filter has its 3dB cut-off at 2.6kHz. From the power density spectra shown in Figure 8.8 it should also be kept in mind that very little noise energy is present in the headset cavity above 3kHz and this can influence the accuracy of the attenuation measurement above 3kHz.

The seemingly bad performance at the low frequency range (0-250Hz) can be due to a lack of low frequency signal energy from the external noise source since the external noise source was expected to have a bad low frequency response. The lack of low frequency energy is also visible in the power density spectra of Figure 8.8.

The bandwidth energy attenuation of the system was calculated in the frequency domain. The spectral power of the reference noise (No ANR) was related to the spectral power of the attenuated noise (ANR on) in a logarithmic scale as shown in equation 8.2.

$$\text{Energy Attenuation} = 10\log\left(\frac{\sum_{n=1}^{N}\text{Power Density Specrum (No ANR)}}{\sum_{n=1}^{N}\text{Power Density Specrum (ANR)}}\right) \qquad 8.2$$

From equation 8.2 the energy attenuation of the system was found to be 10.43dB.

The A-weighted attenuation of the system was calculated in the same manner with A-weighed power density spectra as shown in equation 8.3.

$$\text{Attenuation (dBA)} = 10\log\left(\frac{\sum_{n=1}^{N}\text{A-Weighted Power Density Specrum (No ANR)}}{\sum_{n=1}^{N}\text{A-Weighted Power Density Specrum (ANR)}}\right) \quad 8.3$$

The A-weighted attenuation was calculated to be 18.6dBA.

8.4.2. Spectral, A-weighted and energy attenuation with a additional interference signal

Since the headset must be able to receive additional communications signals that must be relayed to the pilot of the aircraft, it is essential to determine what effects this additional interference signal could have on the performance of the ANR system.

From section 2.1.1 it was shown that the interference signal is uncorrelated with the reference microphone signal, this makes it possible to transmit a communications signal to the pilot via the loudspeaker whilst the adaptive filter only cancels unwanted noise.

It was however seen in section 5.2.3 that an added communications signal makes it harder for the adaptive filter to estimate the headset transfer. It must be kept in mind that no transducer transfers were included for the system of section 5.2.3.

The spectral attenuation of the system was investigated for the case where a tonal interference signal is present. A 1.5kHz sinusoid was used as an interference signal since it could easily be detected in the spectral analysis.

This easy detection of the interference made it possible to calculate the bandwidth energy attenuation using equation 8.2. The energy around the detected tone is ignored in the attenuation calculation. The bandwidth attenuation with an added 1.5kHz interference signal is shown in Figure 8.10.

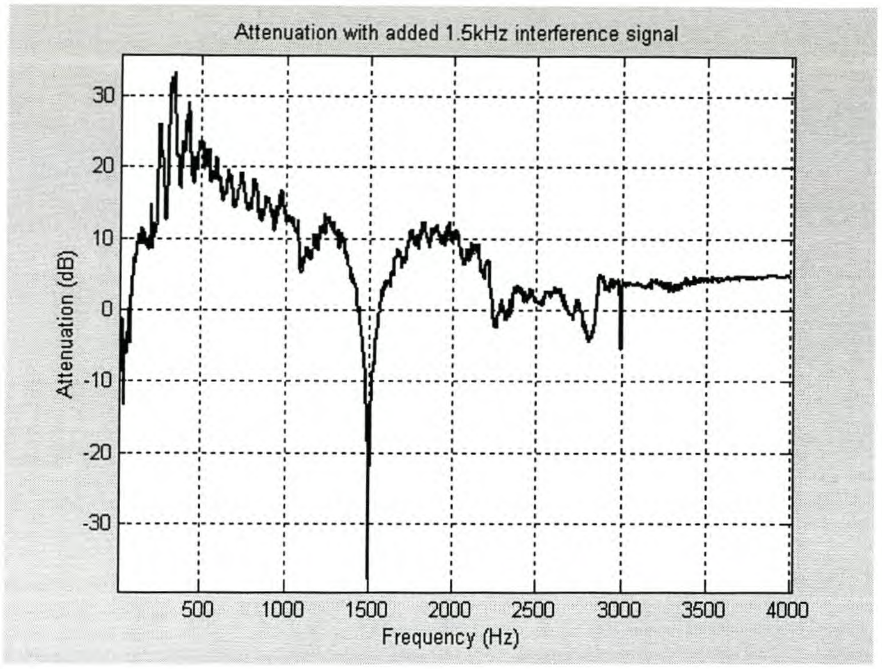


Figure 8.10 Attenuation of ANR system with 1.5kHz tonal interference signal

The noise energy attenuation was calculated to be 10.1dB and the A-weighted attenuation was calculated to be 15.3dBA. This shows that the interference signal has minimal effect on the attenuation of the noise whilst the tone is unaffected. Considerations concerning the attenuation in the 0-250Hz and 3-4kHz bandwidths as mentioned in section 8.4.1 are also applicable to Figure 8.10.

The continues attenuation despite the interference signal confirms the theoretical approach followed in section 2.1.1 and also shows an drop in attenuation as expected from section 5.2.3. To determine the full effect of an added communication signal, it is suggested that subjective testing be done in further work.

8.4.3. Directionality of noise cancellation

The adaptive filter algorithm calculates the headset transfer as a function of the correlation between the noise inside and outside the earmuff.

The microphones measuring the inside and outside noise are always directional to a certain extent. Inside the earmuff this has no implication since the earmuff cavity is small. Outside the earmuff the microphone is placed in free space and noise received by the outside microphone is received as a function of the microphone polar frequency response.

This suggests that the direction of the reference microphone with respect to the noise source should thus have an effect on the amount of noise suppression experienced inside the earmuff cavity.

Figure 8.11 shows the noise energy and dBA attenuation inside the headset cavity in dB scales with respect to the rotation of the reference microphone on the headset from the noise source in degrees.

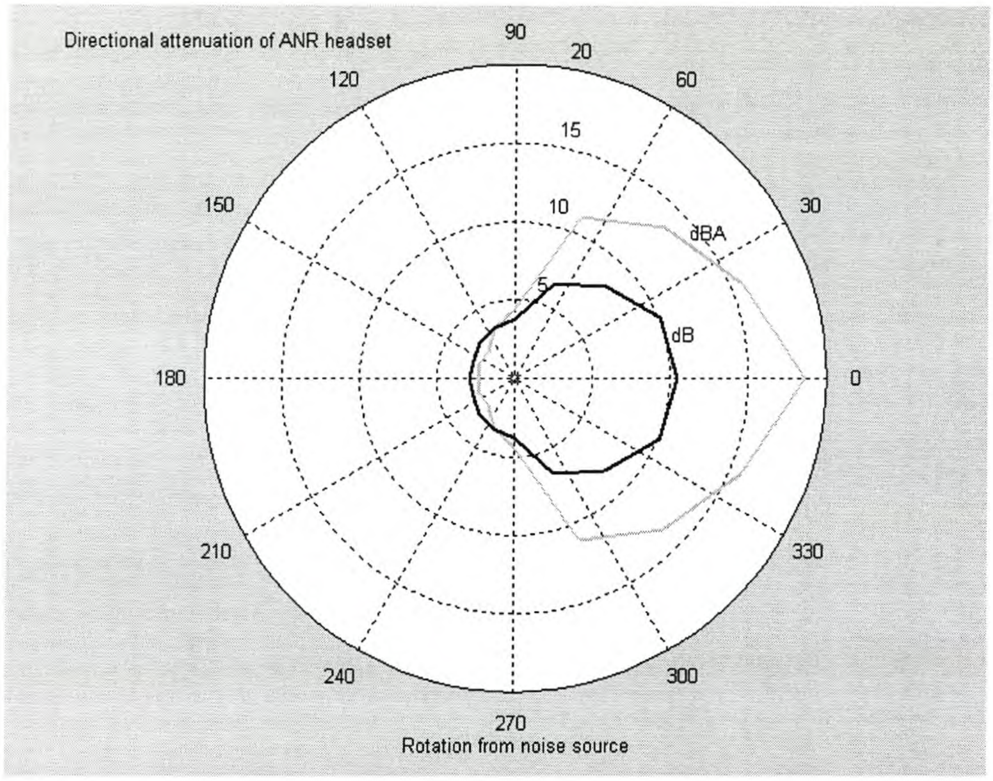


Figure 8.11 Directional polar response of attenuation by ANR headset

This shows that the digital feedforward method attenuates noise directional with respect to the bearing of the outside reference microphone toward the noise source.

This implies that the reference microphone of the ANR system should always be omni-directional or should be placed at such an orientation that the main lobe of the microphone polar frequency response points in the direction of the primary noise source that requires noise cancellation.

8.5. <u>Simulation versus measurements</u>

The simulated operation of a number of adaptive filter algorithms was investigated in chapter 3, 4 and 5. A simulation environment was constructed according to specifications set out in chapter 2 and could be approximated as a simplified version of the practical system. The following differences however do existed between the simulated and practical systems.

- The headset transfer was approximated from an expected practical transfer.
- The acoustic headset transfer phase response was approximated as being linear.
- Acoustic and electrical delays were timed exactly to the simulation sample rate.
- No anti-aliasing filters were included in the simulated environment.
- The simulation adaptive filter error signal was ideally constructed without any losses.
- All simulated acoustic propagation was directional to one dimension.
- The simulation sample rate was different from the practical system sample rate.

From the above considerations it can be seen that it is difficult to compare simulated attenuation levels with actual measured results. This was not the aim for constructing the ANR simulation environment.

The ANR simulation environment was constructed to evaluate different adaptive filter algorithms and their behavior for different situations. Chapter 6 showed that the simulated environment could provide significant results for establishing the best adaptive filter algorithm. The simulation environment also allowed the investigation of ways to ensure that the chosen algorithm is set-up for optimal performance.

8.6. <u>Measurements conclusion</u>

The above measurements give a performance estimate for the feedforward adaptive filter topology with a Kaczmarz Projection algorithm. Chapter 9 and 10 will conclude this thesis with suggestions for further research and a summary.

Chapter 9

9 New Feedforward ANR possibilities

Further work that could improve feedforward ANR systems will be investigated in broad terms in this chapter. These improvements were not fully investigated since they diverged from the main scope of this thesis, namely to find a good adaptive filter algorithm and implement broadband noise cancellation. It was however felt appropriate that this work be mentioned since it could lead to new original topologies for feedforward ANR.

9.1. Further research

9.1.1. Algorithm possibilities

This thesis investigated a number of well known adaptive filter algorithms to perform ANR. New advances in the field of blind signal separation have led to the discovery of independent component analyses (ICA). [23]

Although [23] states that practical considerations hinder ICA from being a likely solution for noise cancellation, [24] shows that an ICA based algorithm can be derived that performs better noise cancellation than the conventional LMS algorithm.

The improvement is attributed to the ability of ICA algorithms to utilise higher order statistics, where the normal LMS, KP and RLS algorithms only utilise second order statistics.

Further investigation in to the theory of ICA and ICA for noise cancellation is suggested.

9.1.2. Topology possibilities

In the previous measurements, a 1.5kHz sinusoid was used to access the influence of a communications or external interference signal. These experiments showed good attenuation results. It was however shown in chapter 5 that a more random interference signal might not have such good results.

Some ideas were investigated to improve the ANR system response for more random interference signals. The adapted block diagram of Figure 9.1 is suggested. The aim of this topology will be to limit the speech component taken up in the error microphone signal to the adaptive filter.

A representation of the voice signal component in the error signal can be generated by filtering the incoming communications signal with a filter that represents the speaker to error microphone transfer inside the headset. This filtered communications signal will represent the communications signal component inherent to the adaptive filter feedback error.

The communication signal predominantly consists of a voice signal. By subtracting this known communications signal component from the adaptive filter error signal will thus de-voice the adaptive filter error signal.

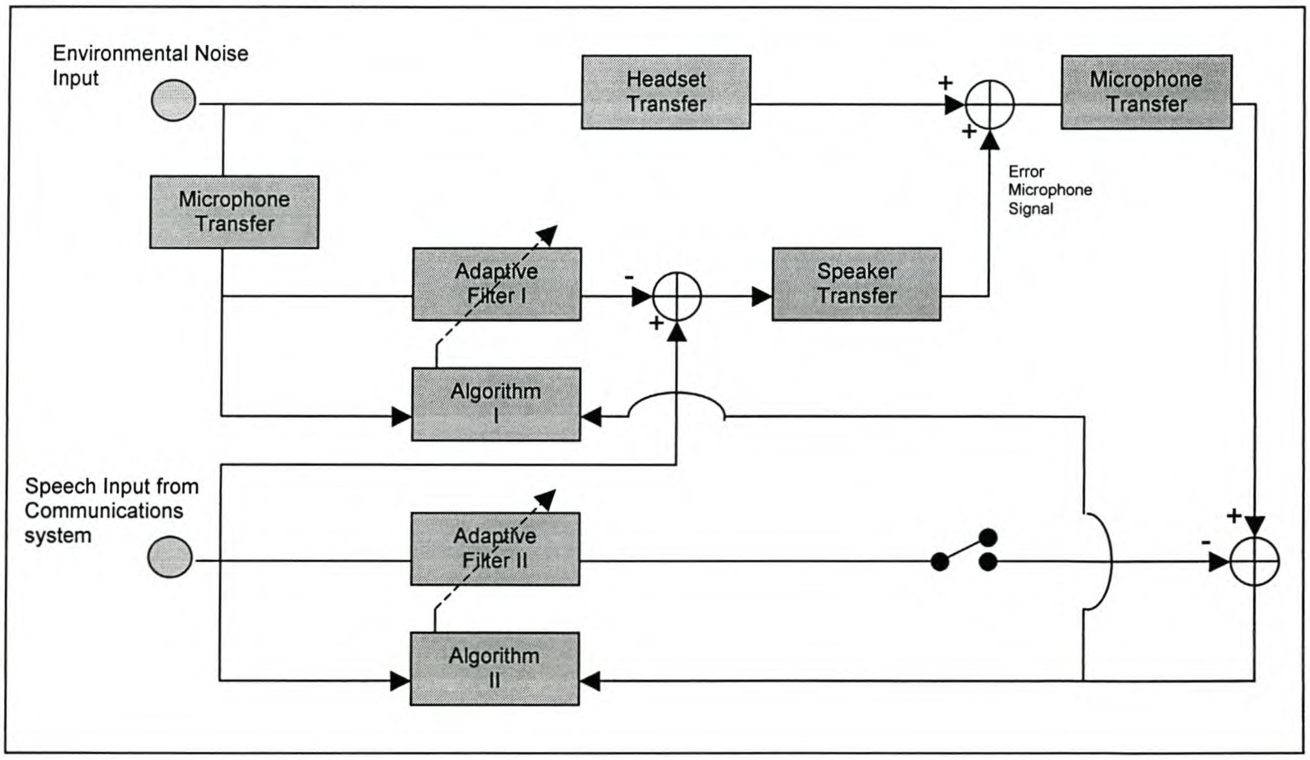


Figure 9.1 Voice limiting error signal topology

The first challenge of this topology will be to estimate the speaker to error microphone transfer. This can be achieved by including another adaptive filter to the system. This adaptive filter will estimate the speaker to error microphone transfer from the communications signal as reference input and an error signal that is generated by subtracting the error microphone signal from the filtered communications signal.

To ensure conversion of the second adaptive filter the adaptive filter coefficients should only be updated when a communications signal is present. To establish if a communications signal is present was the second challenge of this new topology.

9.1.2.1. Speaker to error microphone transfer estimation filter

The algorithm for this estimation requires a high interference signal rejection capability. The filter for this estimation did not have to contend with any interference filters since the reference signal originates directly from the aircraft communication system. The adaptive filter for this application would not be required to estimate a high filter order, since the speaker to microphone transfer is not expected be a high order system.

From section 6.4 it was found that the RLS algorithm would best acquire the system transfer in question.

9.1.2.2. Communications signal presence

For this topology to work, it is essential to know if a communications signal is passing through the speaker to microphone transfer system. This is essential since it would be impossible to estimate the speaker to error microphone transfer if no communications signal passes through the system. The adaptive filter should thus only operate if a communication signal is present.

The process of determining if a speech signal is present can be preformed by filtering the communications signal through a 80-800Hz band pass filter. This must be done since most of the power in a speech signal is concentrated in this bandwidth. The 80Hz high pass cut off also ensured that no dc offset for the signal is present.

The signal power for every sample of the communication signal must be calculated. If the signal power of the current sample exceeds a chosen threshold, a speech signal is assumed present. If the signal power is found below this threshold, it should be assumed that no speech signal is present. Such a power/threshold plot is shown in Figure 9.2.

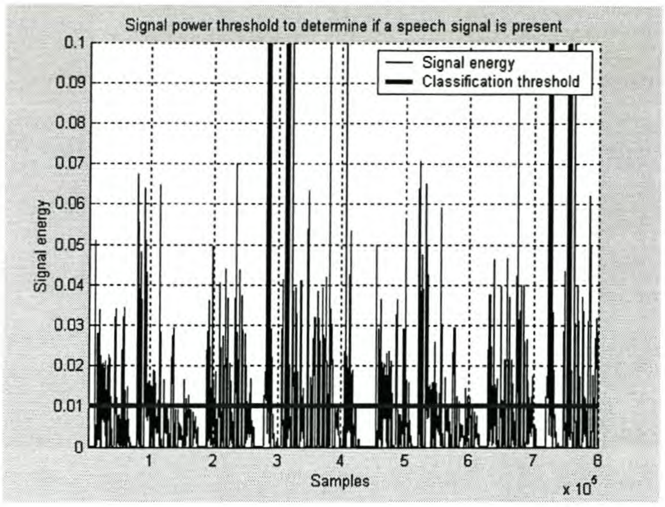


Figure 9.2 Communications signal presence

The second adaptive filter tap weights must only be updated if a communications signal is classified as being present (above the threshold). If the communications signal is classified as not being present, the filter tap weights must be restored to the adaptive filter tap weights most recently updated with a speech signal. This is to ensure stability of the new topology.

The implementation of the second adaptive filter should be done in such a way that devoicing should only take place if the second adaptive filter output ensured a smaller error signal to the first adaptive filter. If the second adaptive filter increases the error signal it should be decoupled from the system to allow more time for the second adaptive filter algorithm to converge until such a time that the error signal improves.

Prelimary simulations showed an improvement in attenuation with this topology but further investigation would be needed before such a topology can be implemented. Such work is recommended for further work in the field of ANR.

9.2. Conclusion to new feedforward ANR possibilities

In conclusion to this chapter it can be seen that there exist some new avenues to improve feedforward ANR. Most research in this field has been concentrated on wide sense stationary statistical algorithms and the basic feedback topology as given in section 2.1. It is therefore suggested that the above-mentioned further work be attempted as part of new research in feedforward ANR for headsets.

Chapter 10

10 Conclusion

This chapter will collaborate on the contributions and achievements of this thesis in relation to statements from literature and measurements made of the current SAAF employed ANR analog feedback system.

10.1. Contributions

10.1.1. Adaptive filter algorithm investigation

Three adaptive filter algorithms were investigated in a simulated environment. It was found that the Kaczmarz Projection algorithm was the most practically implementable solution with the best performance characteristics. The optimal adaptive filter evaluation was summarised in Table 6.1 of chapter 6.

Throughout the evaluation of the different adaptive filters the effects of communications speech and different transducer transfers were investigated. This led to a proposed new topology strategy for feedforward ANR, as described in section 9.1.2 of chapter 9.

10.1.2. Broadband noise attenuation

One of the objectives of this thesis, as set out in chapter 1, was to achieve broadband noise attenuation with the feedforward method. Previous work on feedforward ANR showed that broadband noise attenuation has not been achievable due to acoustic and electric delay limitations of current experimental systems.

According to [2] a digital feedforward control system is used for narrow band acoustic noise attenuation and analog feedback methods are more applicable to broadband noise attenuation.

[2] also states that broadband noise attenuation is unpractical for a digital system since it is to difficult to ensure that the electric delays (sampling delays, D/A converters and low pass filter delays) of the system is shorter than the acoustic delays for sound to travel from the reference microphone to the loudspeaker.

[7] classified broadband active noise reduction to a bandwidth of up to 900Hz. In [7] the combination of a digital feedforward and analog feedback system was simultaneously employed. The feedforward system reduced tonal noise while the feedback system was used to reduce broadband noise (900Hz). [7] also states that good broadband performance for the feedforward method requires shorter electrical delays than acoustic delays.

These electrical versus acoustic delay limitations were confirmed by findings in this thesis as shown in section 7.2. Section 7.2 was also dedicated to solving this phenomenon. It is due to this solution that broadband noise attenuation was found possible with this digital feedforward algorithm and assembly.

It was also shown that some digital filtering could resolve algorithm divergence due to uncorrelated DC offset values produced by the A/D converters of the system. This is described in section 7.3.

10.2. Final Summary

This thesis has set out to investigate broadband feedforward active noise reduction for aircraft headsets.

An adaptive filter simulation environment was formulated to investigate different adaptive filter solutions for the feedforward topology. It was found that the Kaczmarz Projection algorithm would ensure optimal attenuation for this topology.

A practical system was constructed to implement the chosen adaptive filter algorithm. Acoustic and electronic delay problems that hinder broadband noise attenuation were identified whilst constructing the practical system. These problems were adequately solved to ensure broadband noise reduction. This thesis thus shows that broadband feedforward noise reduction is viable, contrary to [2] and [7].

Since this thesis was motivated by the SAAF, it was thought applicable to show the improvement in attenuation from the current SAAF feedback ANR system to the feedforward system developed in this thesis.

The A-weighted power density spectrums of the two systems were compared to show the amount of energy in the measurements compared to each other. These spectra are shown in Figure 10.1.

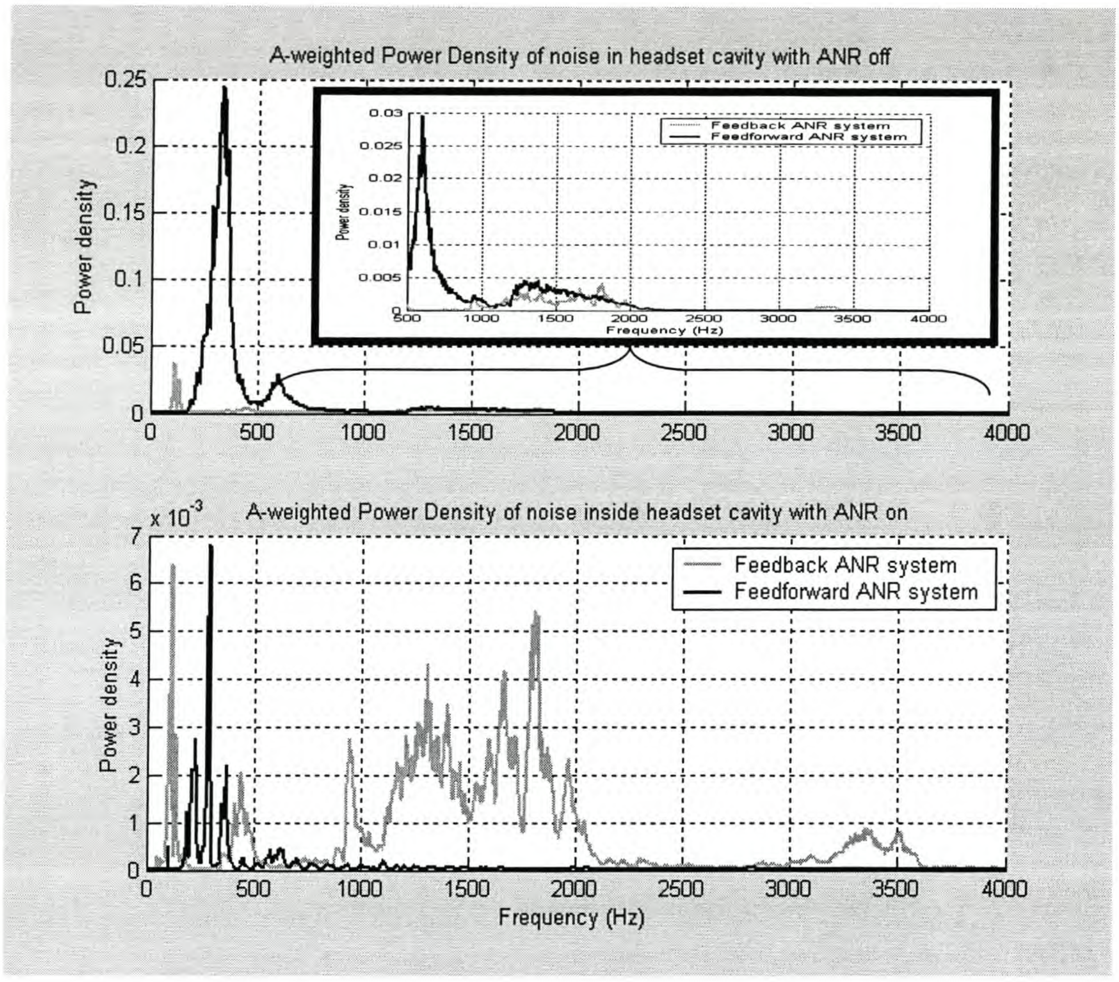


Figure 10.1 A-weighted power density spectra for feedback and feedforward ANR

The top graph of Figure 10.1 shows the energy inside the headset cavity before the ANR systems was switched on. It can be seen that the energy inside the headset cavity for the feedback method was much less than for the feedforward method.

The power density of the measurements with noise reduction is shown in the bottom graph of Figure 10.2. This shows that the residue noise energy after noise reduction is much higher for the Feedback system than for the Feedforward system. This constitutes improved system performance for the Feedforward system since more energy was present in the Feedforward measurement before the ANR system was activated. This attenuation improvement is also visible in the A-weighted and energy attenuation measurements of the systems as shown in Table 10.1.

	A-weighted Attenuation (dBA)	**Energy Attenuation (dB)**
Feedback ANR	0.4 dBA	4.9 dB
Feedforward ANR	18.6 dBA	10.4 dB

Table 10.1 Attenuation of Feedback versus Feedforward ANR systems

The big difference in the dBA attenuation for the two systems can be attributed to the fact that the A-weighting curve values high frequency attenuation more than low frequency attenuation since the human ear is more susceptible to high frequencies.

A frequency versus attenuation plot was constructed from the above power density spectra and the resulting figure is shown in Figure 10.2.

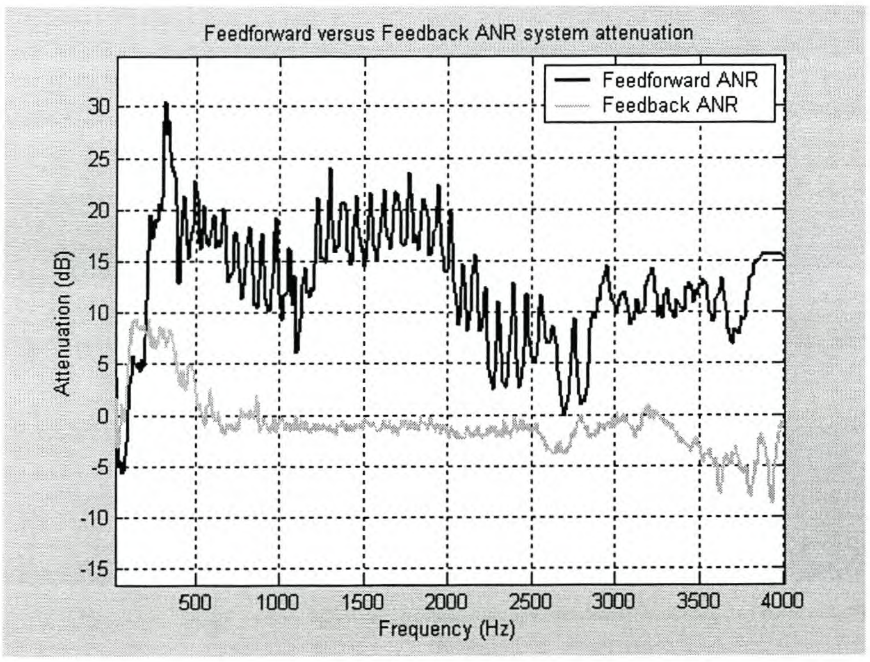


Figure 10.2 Feedforward versus feedback active noise reduction

The feedforward topology thus ensures higher broadband attenuation than previously experienced in the feedback system. It must be mentioned however that the feedforward system will be a much more expensive system to implement as apposed to the feedback system.

References

[1] Lueg, P. 1936. *Process of silencing sound oscillators*. USA. US patent No. 2 043 416

[2] 2000. *Active Noise Reducing Headset*. Institute of Sound and Vibration Research. University of Southampton. UK. Online Symposium for Electronics Engineers.

[3] Roure, A. 1985. *Self adaptive broadband active sound control system*. Journal of sound and vibration. 101 429-441

[4] Eriksson, L.J. Allie, M.C. 1989. *Use of random noise for on-line transducer modelling in an adaptive active attenuation system*. Journal of the Acoustic Society America. 85 797-802

[5] Smith, C.J. 2002. *Headset transfer and headset transfer in the presence of Active Noise Reduction.* RSA, University of Stellenbosch.

[6] Guy, C.G. Sherratt, R.S. Townsend, D.M. February 2002. *Cancellation of siren noise from two-way voice communication inside emergency vehicles*. Computing & Control Engineering Journal.

[7] Rafaely, B. Jones, M. 2002. *Combined feedback-feedforward active noise-reducing headset-The effect of the acoustics on broadband performance*. Institute of Sound and Vibration Research. UK.

[8] Haykin, S. 1991. *Adaptive Filter Theory*. Second Edition. USA. Prentice Hall, Inc.

[9] Chen, S.J. Gibson, S.G. March 2001. *Feed forward Adaptive Noise Control with multivariable Gradient Lattice Filters.* IEEE Transactions on Signal Processing.

[10] Ikeda, S. Sugiyama, A. March 1999. *An Adaptive Noise Canceller with Low Signal Distortion for Speech Codecs.* IEEE Transactions on Signal Processing.

[11] Haykin, S. 2002. *Adaptive Filter Theory.* Fourth Edition. USA. Prentice Hall, Inc.

[12] Oppenheim, A.V. Schafer R.W. 1975 Digital Signal Processing. USA. Prentice Hall, Inc

[13] Aström, K.J. Wittenmark, B. *Adaptive Control.* Second Edition. Addison-Wesley Inc.

[14] Franklin, G.F. Powell, J.D. Workman, M. 1997. *Digital Control of Dynamic Systems.* Third Edition. Addison-Wesley Inc.

[15] Shaw, E.A.G. Thiessen, G.J. 1962. *Acoustics of circumaural earphones.* Journal of the Acoustic Society America, 34(9), 1233-1246

[16] Rafealy, B. Carrilho, J. Gardonio, P. *Novel active noise-reducing headset using earshell vibration control.* Institute of Sound and Vibration Research. UK.

[17] 1995*. Data pack for Active Noise Reduction Headset for Aircrew.* Helmet Integrated Systems Limited Research and Development Department. Version 6. UK. Helmet Integrated Systems Limited

[18] Snyder, S.D. Hansen, C.H. 1989, *The influence of transducer transfer functions and acoustic time delays on the implementation of the LMS algorithm in active noise control.* University of Adelaide. Australia.

[19] Tobias, O.J. Bermudez, J.C.M. Bershad, N.J. April 2000. *Mean Weight Behaviour of the Filtered-X LMS Algorithm.* IEEE Transactions on Signal Processing, Vol. 48. No. 4.

[20] Woodbury, M. 1950. *Inverting modified matrices.* Statistical Research Group, Princeton University. USA.

[21] Bronzel, M.J. Fuller C.R. 1995 *Implementation of fast transversal filters for structural acoustic control*. USA.
http://www.auditory.org/asamtgs/asa95wsh/2aEA/2aEA9.html

[22] ANSI S3.19, 1974, *Method for the measurement of real ear protection of hearing protectors and physical attenuation of earmuffs*. USA, Acoustic society of America, New York.

[23] Hyvårinen, A. Karhunen, J. Oja, E. 2001. *Independent Component Analysis*. John Wiley & Sons, Inc.

[24] Hyung-Min, P. Sang-Hoon, O. Soo-Young, L. *On Adaptive Noise Cancelling based on Independent Component Analyses*, Korea advanced Institute of Science and Technology.

[25] *AD7538 LC^{2}MOS µP-Compatible 14-bit DAC*. Analog Devices

[26] *ADS7805 16 bit,10µs Sampling CMOS Analog-to-Digital converter*. Burr-Brown

[27] October 1999. *DSP56311 User's Manual*. Motorola

[28] *74LVXC4245 8-Bit Dual Supply Configurable Voltage Interface Transceiver with 3-STATE Outputs*. Fairchild.

[29] August 1999. *Updated DSP56300 24-bit Digital signal processor family manual*. Motorola

[30] January 1998. *UAF42 Universal filter*. Burr-Brown

[31] *DSP56300 24-bit Digital signal processor family manual*. Motorola

[32] February 1999. *DSP56311 Semiconductor Technical Data*. Motorola

[33] November 1999. *DSP56311EVM Evaluation Model Hardware Reference Manual*. Motorola

[34] Proakis, J.G. Manolakis, D.G. 1996. *Digital Signal Processing*. Third Edition. USA. Prentice Hall, Inc.

[35] Peebles, P.Z. 1993. *Probability, Random Variables, and Random Signal Principles*. Third Edition. USA. McGraw-Hill, Inc.

[36] Kinsler, L. E. Frey, A.R. Coppens, A.B. Sanders, J.V. 1982. *Fundamentals of Acoustics.* Third Edition. USA. John Wiley & Sons, Inc.

Appendix A

A Real time data converters for the DSP56311

This appendix explains the construction of a conversion system with no pipeline delays. This conversion system will be able to take a current analog sample, digitise the sample and release it for signal processing. After processing the processed digital value must be reconstructed to an analog sample within the same sample that it was received or processing. The following converters were interfaced to the Motorola DSP56311EVM

- AD7538 digital-to-analog converter was configured to operate from the host port (HI08) of the DSP56311EVM
- Two ADS7805 analog-to-digital converters were configured to interface to port A of the DSP56311EVM.

A.1. Device interfacing

A.1.1. Analog Devices AD7538

The Analog Devices AD7538 is a 24-pin device that performs 14-bit digital to analog conversions in a conversion time of approximately 300ns per conversion. This allows the D/A converter to perform up to 3.3 MSPS. [25]

This converter has three control signals namely the chip select (CS), write (WR) and load digital to analog converter (LDAC) signals. The WR and LDAC signals were tied to digital ground (DGND). This enables fast conversion by just clocking the CS line low to perform a conversion.

The D/A was configured for bipolar operation. The D/A circuitry was configured as shown in Figure A.1.

The host port of the DSP56311EVM was used to interface the digital signals to the AD7538. The host port was set up as I/O pins by configuring the host data direction register, host control register and the host polarity control register. [27]

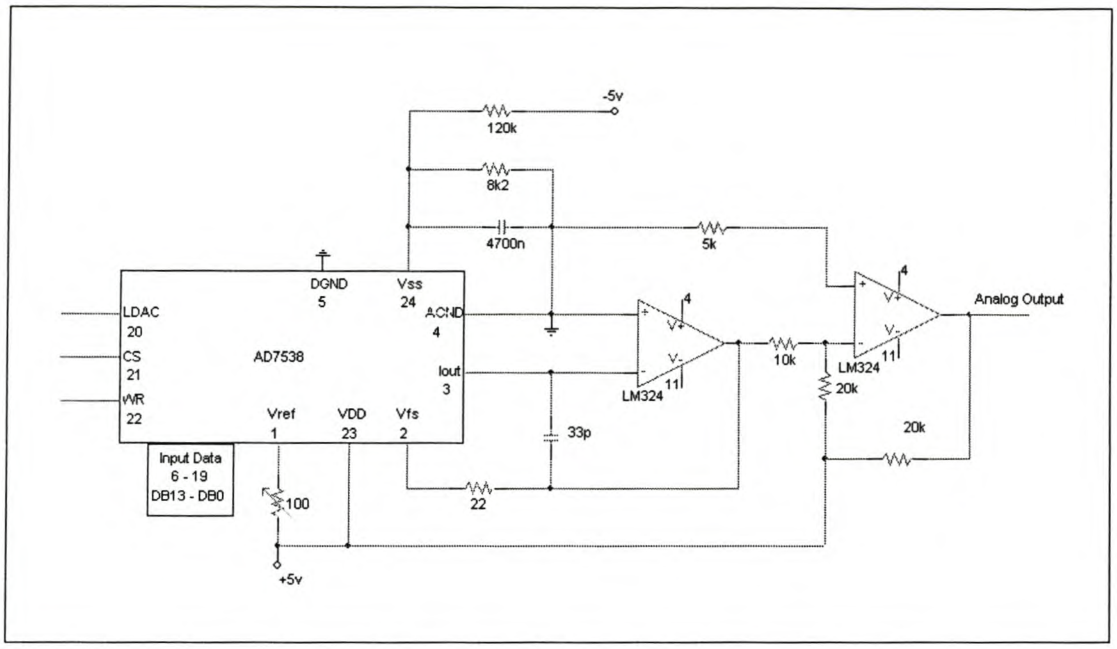


Figure A.1 Bipolar operation of AD7538

The host data direction register was setup to make the entire host port bus outputs. The host control register was cleared to disable all possible interrupts that work with the host port. The host polarity register was configured to activate I/O mode for the host port. Assembler code for this setup is shown below.

```
movep   #$FFFF,x:M_HDDR        ; Host Data Direction Register (all outputs)
movep   #0,x:M_HCR             ; Host Control Register (all interrupts disabled)
movep   #1,x:M_HPCR            ; Host Polarity Control Register (GPIO active)
```

The bipolar configuration of the AD7538 does not support two's compliment binary operation [22]. The DSP56311 does however support two's compliment operation [25].

A software conversion had to be implemented in the DSP to convert the DSP two's compliment values to binary values of the same analog value for the AD7538. The DSP assembler code for the conversion and A/D triggering is shown below.

```
Host_out
        bset    #15,x:M_HDR                ; Disable A/D CS high

Anti_2s_compliment

        move    a2,x0                      ; Change the DSP 2's compliment value
        jset    #1,x0,negative_2_comp      ; to not 2's compliment for D/A

        lsr     #10,a                      ; shift right since host port reads
        nop                                ; least significant 14 bits
        add       #$002000,a              ; 2's compliment compensation for positive values
        nop
        jmp       end_2_comp

negative_2_comp

        neg       a                       ; 2's compliment compensation for negative values
        lsr       #10,a                   ; shift right since host port reads
        nop                               ; least significant 14 bits
        sub       #$001FFF,a              ; 2's compliment compensation for negative values
        neg       a                       ; 2's compliment compensation for negative values

end_2_comp
        nop
        nop

        move   a1,x:M_HDR                 ; Write out value of A to the host port
        bclr      #14,x:M_HDR             ; Clock the CS of D/A low
        rep #90                           ; Clock the CS of D/A low
        nop
        bset      #14,x:M_HDR             ; Clock the CS of D/A high again

        rts
```

This configuration for the AD7538 preformed well as a real time D/A converter for the DSP56311.

A.1.2. Burr Brown ADS7805

The Burr Brown ADS7805 is a bipolar 28-pin device that performs 16-bit analog to digital conversions in a conversion time of approximately 10µs per conversion. The ADS7805 can perform a maximum of 100 KSPS and gives a parallel output. [26].

The A/D converter has two control signals. A chip select signal (CS) selects the device while the (R/C) signal is a conversion enable signal. Pulling both CS and R/C low enables a conversion. CS and R/C must be reset before a conversion is completed. If CS is pulled low a second time, the data is put on the 16 pit parallel data bus.

The configuration used to interface the ADS7805 with the DSP56311 is shown in Figure A.2. It should be noted that the ADS7805 operates at a 5 Volt digital voltage. The 5V digital output bus and control signals were converted to a 3.3 Volt bus for the DSP using the 74LVX4245. [28].

Port A of the DSP56311 was used to interface the digital bus of the A/D to the DSP. To configure this port, two registers had to be configured. They are the external bus control register and the address attribute register 1. [29].

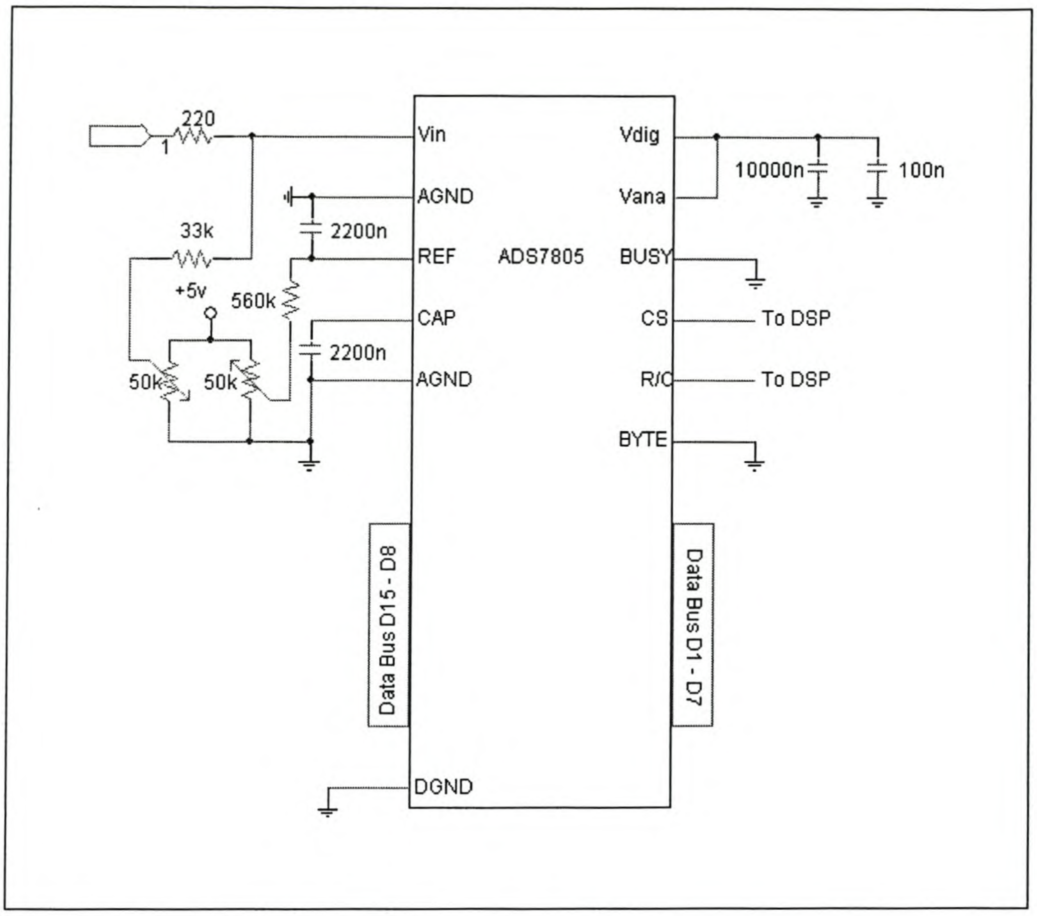


Figure A.2 ADS7805 interface circuit

Address attribute register 1 was set-up to access the external bus for data when a memory address larger than 00C000 hexadecimal in the Y memory space is addressed with an assembler move instruction. The 24 bit external data bus values are then returned the to the register instructed in the move instruction.

The external bus control register was set-up to have one wait state to wait for data to settle on the external data bus. The external data bus was also made active by toggling bit 0 of the external bus control register.

Register set-up and data access code is shown below.

```
movep    #$012421,x:M_BCR          ; AARx - 1 wait state
movep    #$00CB20,x:M_AAR3          ; Set up port A from y:$00CXXX

bset     #0,x:M_BCR                 ; Set up to Read from Port A
move     y:$00C00C,a               ; Read from Port A
bclr     #0,x:M_BCR                ; Put Port A off
```

To clock the 2 A/D converters correctly to convert analog values to digital values was done using control signals from the DSP SCI serial and host port. Three signals were set-up as I/O signals to control the conversion process.

The SCI port was set-up as I/O signals using the SCI control register and SCI direction register. The appropriate bits were set in the mentioned registers to set-up the SCI read and write signals as output signals. Code for this set-up is shown below.

```
bclr    #0,x:M_PCRE                ; SCI Control register (PC0 GPIO active)
bclr    #1,x:M_PCRE                ; SCI Control register (PC1 GPIO active)
bset    #0,x:M_PRRE                ; SCI Direction Register (PC0 output)
bset    #1,x:M_PRRE                ; SCI Direction Register (PC1 output)
```

The host port signal was set-up as shown in section A.1.1.

The code for appropriately triggering the A/D converters and acquiring the data of the two A/D converters to the general registers A and B is shown below.

```
    bset    #0,x:M_PDRE                ; A/D control pin (CS_b pin)
    bset    #1,x:M_PDRE                ; Conversion clock(Conv)
    bset    #15,x:M_HDR                ; A/D control pin (CS_a pin)

    bclr    #1,x:M_PDRE                ; Conversion clock (Conv)

    ; First A/D start conversion

    bclr    #0,x:M_PDRE                ; A/D control pin (CS_b pin)
    rep #10
    nop
    bset    #0,x:M_PDRE                ; A/D control pin (CS_b pin)

    ; Second A/D start conversion

    bclr    #15,x:M_HDR                ; A/D control pin (CS_a pin)
    rep #10
    nop
    bset    #15,x:M_HDR                ; A/D control pin (CS_a pin)

    bset    #1,x:M_PDRE                ; Conversion clock (Conv)

    ; Conversion Wait

    do #7,_end
    rep #90
    nop
_end
    ; Read data A to D 1

    bclr    #15,x:M_HDR                ; A/D control pin  (CS_a pin)
    rep #5
    nop
```

```
bset    #0,x:M_BCR              ; Set up to Read from Port A
move    y:$00C00C,a             ; Read from Port A
bclr    #0,x:M_BCR              ; Put Port A off
bset    #15,x:M_HDR             ; A/D control pin (CS_a pin)

; Read data A to D 2

bclr    #0,x:M_PDRE             ; A/D control pin (CS_b pin)
rep #5
nop
bset    #0,x:M_BCR              ; Set up to Read from Port A
move    y:$00C00B,b             ; Read from Port A
bclr    #0,x:M_BCR              ; Put Port A off
bset    #0,x:M_PDRE             ; A/D control pin (CS_b pin)
```

This configuration for the ADS7805 preformed well as a real time value A/D converter for the DSP56311.

A.1.3. Anti-Aliasing filter

The ADS7805 required an anti-aliasing filter to prevent any aliasing of signals when an A/D conversion is preformed. The bandwidth of the ANR system was set at 3kHz in the design specifications. The chosen sample rate of the system is 8kHz. The anti-aliasing filter thus had to be designed to have a low pass characteristic with a cut-off frequency of between 3-4kHz.

A cut-off frequency of 3kHz was chosen to ensure best adequate attenuation of the signal at 4kHz. This will ensure the least aliasing.

The UAF42 Universal filter [30] was used to realise a second order Butterworth low pass filter. The closest filter to a Butterworth filter that could be realised with commercially available resistor values was of the following form.

$$F(s)=\frac{3.119\times10^{8}}{s^{2}+2.558\times10^{4}s+3.127\times10^{8}} \qquad \text{A.1}$$

A bode plot of this filter is shown in Figure A.3. This design has a theoretical cut-off frequency of 2.74kHz. This is still an acceptable cut-off frequency. The configuration of this filter with the UAF42 is shown in Figure A.4.

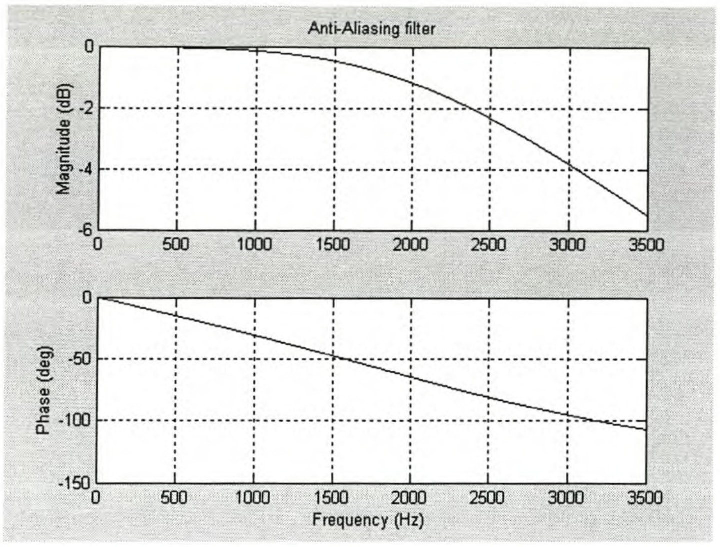


Figure A.3 Anti-aliasing filter frequency response

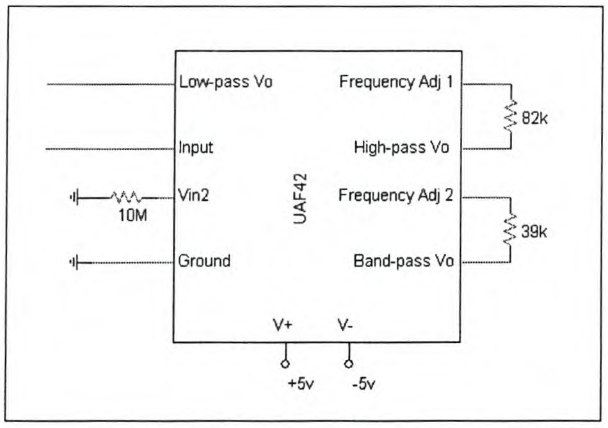


Figure A.4 UAF42 configuration

The measured response of the built filter was compared to the designed specifications as set out above. The active filter responded similarly to the design specifications. The 3dB cut-off frequency was measured to be 2.6kHz. This is adequate for the purposes of this filter.

A.2. Real time conversion system conclusion

The A/D and D/A transfer of this conversion system was measured with a frequency sweep method. The resulting amplitude frequency response is shown in Figure A.5. This response includes the anti-aliasing filter, a dc limiting filter and additional single pole low pass reconstruction filter with cut-off frequency of 4kHz. The additional reconstruction filter was added to limit the high frequency harmonics caused by the D/A sample and hold.

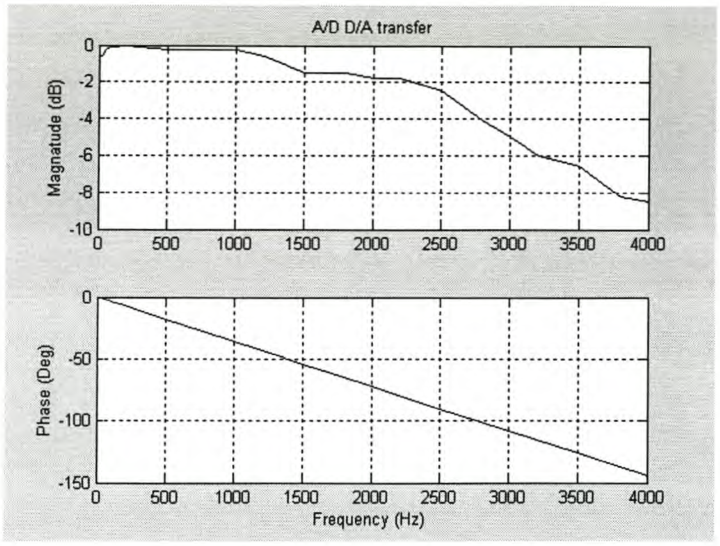


Figure A.5 A/D to D/A transfer at 8kHz

The non-logarithmic phase plot shows that the phase of the transfer changes almost linearly. This is due to the linear phase behaviour of the anti-aliasing filter. A photo of the completed interface is shown in Figure A.6.

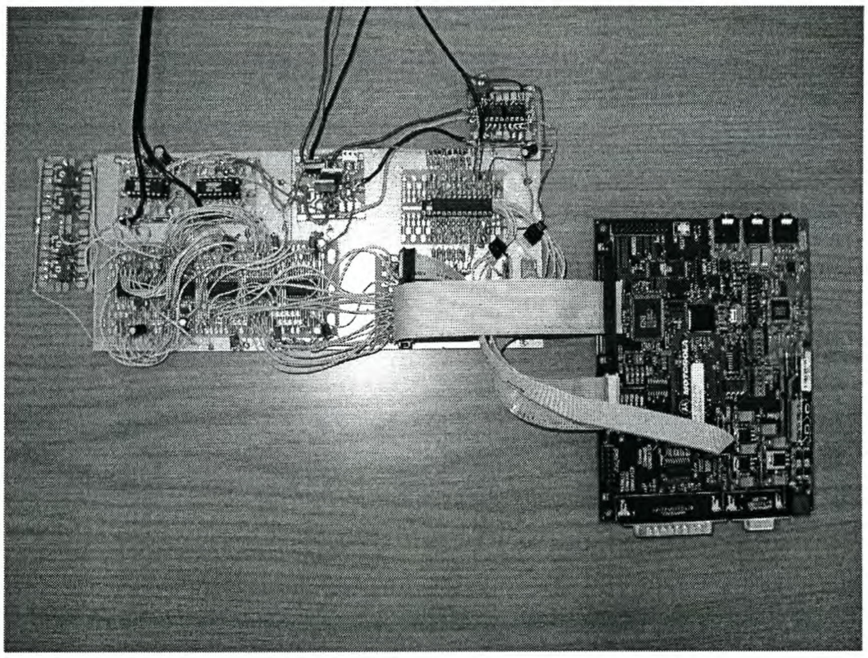
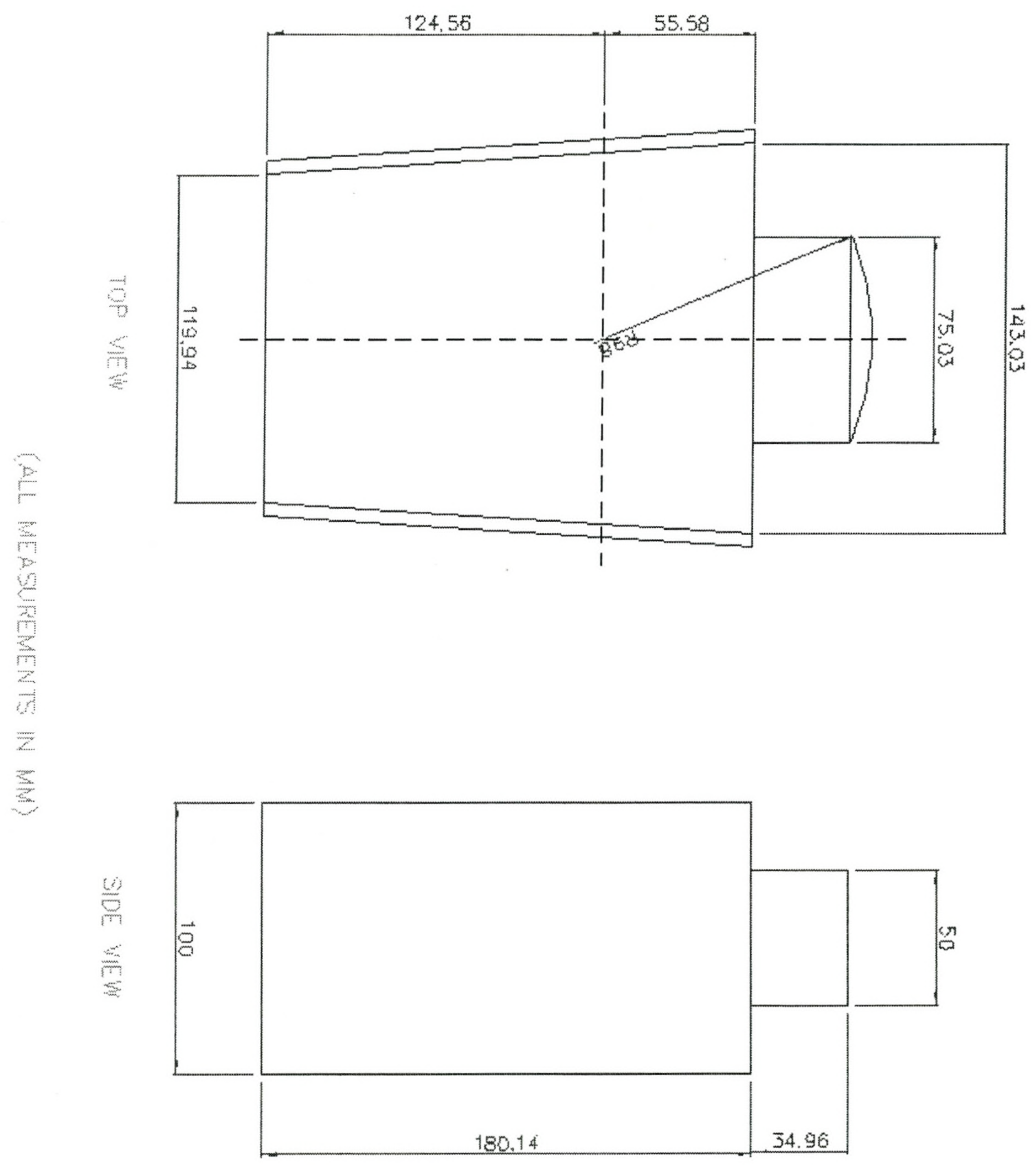


Figure A.6 Completed A/D and D/A conversion interface with DSP56311EVM

This conversion system was successfully implemented and complied with all the requirements set out for the conversion system without pipeline delay and minimal filter delays.

Appendix B

B Artificial head test fixture



Appendix C

C Basic simulation program code

C.1. Main program code

```
% Apadtive Filter evaluation simulation

% Settings

% ================================================================
Timelength          =  10;      % Time duration in Seconds (Max time with Comms = 21s)
Tsample             =  6000;    % Sample rate

Comss_y_n           =  1;       % 0 = no comms / 1 = comms
Comms_mult          =  1;       % Multiplication factor for unity scaled speech
Noise_var           =  2;       % Input noise variance before headset filtering

Speaker_fil_y_n     =  0;       % Speaker filter on == 1 off == 0
Mic_fil_y_n         =  0;       % Microphone filter on == 1 off == 0
Microphone_cutoff   =  100;     % Microphone cuttoff frequency
Speaker_cutoff      =  100;     % Speaker cuttoff frequency

Tap_poles           =  0;       % Number of estimation Poles
Tap_zeros           =  100;     % Number of estimation zeros

Adjust_per          =  0.01;    % Non-stationary adjustment period (seconds) _n must be 1)
Multi_filter_cal    =  1;       % Calculate filters for non stationary and stationary == 1

x_fil               =  0;       % X-filter in operation (0 = no 1 =yes)

% ================================================================
% Analysis Settings
% ================================================================
begin_offset        =  8 ;      % System settling time before frequency analyses start
Bartlett_Windows    =  100;     % Number of Bartlett_estimate windows
lamda               =  1;       % Forgetting factor
mu                  =  0.001;   % 0 > mu < 2/total input power
alfa                =  0.65;
% ================================================================

if Multi_filter_cal == 1

transfer_file_starttime=clock;
disp('Transfer file compiling....')
disp([transfer_file_starttime(4) transfer_file_starttime(5)])

Wanted_sample_rate = Tsample;

P1      =  3;
Z1      =  40;
gain1   =  12;
TPhi1   =  Headset_transfer_estimation(P1,Z1,gain1,10000,Wanted_sample_rate);
```

```
figure(1)
hold

transfer_file_endtime=clock;
disp('Transfer file completed')
disp([transfer_file_endtime(4) transfer_file_endtime(5)])

end


starttime=clock;

disp('Simulation started at')
disp([starttime(4) starttime(5)])

Number_of_samples       =  round(Timelength*Tsample);
Tap_size                =  Tap_poles+Tap_zeros;
Tap_1                   =  P1+Z1;

Max_Tap_size            =  max([Tap_size Tap_1 ]);
X                       =  sqrt(Noise_var).*randn(1,Number_of_samples); % Environmental Noise

if Comss_y_n == 1

[IN,Fs]                 =  wavread('ANR_Speech.wav');

  if Fs == Tsample                                            % No-resampling requird
    Comms               =  IN;                                % Communications Speech
  else
    Wanted_sample_rate  =  Tsample;                           % Resampleing of
    Comms               =  resample(IN,Wanted_sample_rate,Fs); % Comms Signal
  end

else
Comms          =  zeros(1,Number_of_samples)';                % No Communications Speech
end

Max_sampels      =  length(Comms)
Comms          =  Comms_mult.*Comms(1:Number_of_samples);

% ----------------------------
%  Simulation Initialisation
% ----------------------------

T    =  1/Tsample;

% ======================================================================
% Filter coeefficeints INIT
% ======================================================================

Pole   =  P1;
Zero   =  Z1;
HPhi   =  TPhi1;

format long

if Speaker_fil_y_n == 1;

  [BS,AS]  =  butter(1,Speaker_cutoff/Tsample,'high');
  SPhi    =  [BS  AS(2:length(AS))]';              % Speaker 700Hz/4kHz bandpass filter
```

```
else
   BS      =  [1 0];
   AS      =  [0 0];
   SPhi    =  [BS AS(2:length(AS))]';
end

if Mic_fil_y_n == 1;

 [BM,AM]  =  butter(1,[Microphone_cutoff/Tsample],'high');
   MPhi      =  [BM  AM(2:length(AM))]';                      % Microphone 100 Hertz
                                                               % cut-off high pass filter
else
   BM      =  [1 0];
   AM      =  [0 0];
   MPhi    =  [BM AM(2:length(AM))]';
end


% ========================================================================
% Simulation Init
% ========================================================================

Startvar          =  zeros(1,(Max_Tap_size+1));
HU                =  Startvar;
SU                =  Startvar;
out1              =  Startvar;
out               =  Startvar;
error             =  Startvar;
Nucomms           =  Startvar;
XM                =  Startvar;
S                 =  Startvar;
XBL               =  Startvar;
errorM            =  Startvar;
errorBL           =  Startvar;
stop_est          =  round(length(X));
convergence       = [];

% ========================================================================
% Adaptive filter init
% ========================================================================

a                    = 10;
P                    = diag(a*ones(1,(Tap_size))+1);
K                    = diag(a*ones(1,(Tap_size))+1);
Phi                  = [zeros(1,(Tap_size+1))]';
T                    = [];
var                  = 0;
k                    = 0;
sum_uniform_rand                  = 0;
sum_squared_uniform_rand          = 0;
headsetvar_counter                = 0;
cut_adaptive_filter               = 0;
Phianal                           = [];
a                                 = 0;
Theta1    =  [fliplr(S(Max_Tap_size+1-(Tap_zeros):Max_Tap_size+1)) -fliplr(out(Max_Tap_size+1-
(Tap_poles):Max_Tap_size+1-1))]';

% ____________________
%
% Simulation loop
% ______________________

h = waitbar(0,'Please wait...simulation in progress...This can take a while');

for n = (Max_Tap_size+1):length(X)
        waitbar(n/length(X),h)

% Outside headset noise through reference microphone filter
   M1Theta    =  [X(n) X(n-1) -XM(n-1) ]';
   XM(n)      =  M1Theta'*MPhi;                % Reference microphone filter

% Outside headset noise through headset transfer

   HTheta     =  [fliplr(X(n-Zero:n)) -fliplr(HU(n-Pole:n-1))]' ;
   HU(n)      =  HTheta'*HPhi   ;              % Headset filter

   if x_fil == 1

% Electronic signal through speaker filter

   XFTheta    =  [XM(n) XM(n-1) -S(n-1)]';
   S(n)       =  XFTheta'*SPhi;                % Speaker filter

   else

   S(n)       =  XM(n);
   end


% Anti-Noise creation with adaptive filter

   Theta      =  [fliplr(S(n-(Tap_zeros):n)) -fliplr(out(n-(Tap_poles):n-1))]';
   TTheta     =  [fliplr(XM(n-(Tap_zeros):n)) -fliplr(out(n-(Tap_poles):n-1))]';
   out(n)     =  TTheta'*Phi;

% ----------------------------------------------
% Speech + anti-Noise mix before speaker filtered
% ----------------------------------------------

   Nucomms(n) =  Comms(n)-out(n);              % Electronic compensation signal transmitted by speaker

   % Electronic signal through speaker filter
   STheta     =  [Nucomms(n) Nucomms(n-1) -SU(n-1)]';
   SU(n)      =  STheta'*SPhi;                 % Speaker filter

   error(n)   =  HU(n)+SU(n);                  % Signal inside headset

   % Error mic transfer inside earcup

   M2Theta    =  [error(n)  error(n-1) -errorM(n-1) ]';  % Error microphone filter theta
   errorM(n)  =  M2Theta'*MPhi;                % Error microphone filter

   % --------------------------------
   % LMS algorithm
   % --------------------------------

   %Phi       = Phi + mu*Theta1*errorM(n-1);

   % --------------------------------
   % KP algorithm
```

```
    % --------------------------------

    T     =  (Theta1'*Theta1);
    P     =  alfa/(T+0.1);
    Phi    =  Phi + P*Theta1*errorM(n-1);

    % ----------------------------------------
    % RLS algorithm with exponential forgetting
    % ----------------------------------------

    %K      =  lamda*P*Theta1*inv(1+lamda*Theta1'*P*Theta1);
    %Phi    =  Phi + K*errorM(n-1);
    %P      =  lamda*P-lamda*K*Theta1'*P;

    % --------------------------------
    % end of algorithms
    % --------------------------------

    %One delay for microphone acoustic delay

    Theta1      = Theta;

    %Phianal  =  [Phianal   Phi];
    converge    = sum(Phi);
    convergence = [convergence converge];



end
   % --------------------------------
   % End of SIMULATION
   % --------------------------------
end
close(h)

disp('simulation completed - evaluation start')

% ====================================================
% Creation of a NO-AANR signal for evaluation purposes
% ====================================================

if Speaker_fil_y_n == 1;
   speech  =  filter(BS,AS,Comms);
else
   speech  =  Comms;
end

NoAANR  =  HU;
AANR      =  error-speech';
Speeched  =  (NoAANR+speech');


Start        =  Tsample*begin_offset;

NoAANR_Noise_power   =
(1/length(NoAANR(1,start:length(NoAANR))))*sum(NoAANR(1,start:length(NoAANR)).^2);
AANR_Noise_power     =
(1/length(AANR(1,start:length(AANR))))*sum(AANR(1,start:length(AANR)).^2);Verhouding          =
AANR_Noise_power/NoAANR_Noise_power;
ATTENUATION        =  10*log10(1/Verhouding)
```

```
time              = [(1/Tsample).*(1:1:length(NoAANR))];

% Tap weigth plots
%figure
%for i= 1:Tap_size+1
%subplot((Tap_size+1),1,i)
%plot(Phianal(i,:))
%hold on
%plot(HPhi(i).*ones(1,length(Phianal(i,:))),'r:')
%end


figure
plot(time,NoAANR)
hold on
plot(time,out,'r')
title('Actual noise vs Estimated noise')
ylabel('Amplitude')
grid on

Max_dev_1    = max([max(errorM)  max(NoAANR+speech')]);
Max_dev_1    = Max_dev_1 + 0.01*Max_dev_1;

wavwrite((Speeched./Max_dev_1),Tsample,16,'No_ANR.wav')
wavwrite((error./Max_dev_1),Tsample,16,'With_ANR.wav')

figure
subplot(2,1,1)
plot(time,Speeched)
title('Earcup signal with no AANR')
ylabel('Amplitude')
grid on
subplot(2,1,2)
plot(time,error)
title('Earcup signal with AANR')
ylabel('Amplitude')
xlabel('Time (s)')
grid on

time=[(1/Tsample).*(1:1:length(convergence))];

figure(4)
hold on
plot(time,convergence)
grid on
hold on
title('Convergence of Algorithm')
xlabel('Time (s)')
ylabel('Convergence of the sum of the tap weights')

conv_mean = 1/length(convergence).*sum(convergence)
conv_variance = real_time_variance(convergence)

Max_dev    = max([max(NoAANR)  max(AANR)]);
Max_dev    = Max_dev + 0.01*Max_dev;

wavwrite((NoAANR./Max_dev),Tsample,16,'output_noise1.wav')
wavwrite((AANR./Max_dev),Tsample,16,'output1.wav')
```

```
Offset_samples = round(begin_offset*Tsample);
[Attenuation,Frequency_scale_out,Frequency_scale_in]=ANR_Preformance(Tsample,Bartlett_Windows,Offset
_samples);  % AANR Preformance

figure(5)
hold on
plot(Frequency_scale_out(1,2:length(Frequency_scale_in))./max(Frequency_scale_out),-
Attenuation(1,2:length(Frequency_scale_in)),'green')
hold on
grid on
title('Attenuation due to noise reduction (Non-Stationary voiced)')
ylabel('Amplification (dB)')
xlabel('Normalised Frequency [(value)(0.5)(sample rate) = frequency]')

[Head_IN,IN_freq]       =  Bart_esst(X,Tsample);
[Head_OUT,OUT_freq]  =  Bart_esst(HU,Tsample);

Attenuation                =  10*log10(Head_OUT./Head_IN);

A                          =  [Phi(1:Tap_zeros+1)'];
B                          =  [1 Phi((Tap_zeros+2):length(Phi))'];
testout                    =  filter(A,B,X);

[Est_OUT,Est_freq]         =  Bart_esst(testout,Tsample);

EstAtt                     =  10*log10(Est_OUT./Head_IN);

figure(6)
plot(OUT_freq,Attenuation)
hold on
plot(OUT_freq,EstAtt,'r')
legend('Given Headset Transfer','Estimated HEadset Transfer')
grid on

save NLMS_6kHz_stat_voised_xfiltered_15s

endtime=clock;
disp('Simulation completed')
disp([endtime(4) endtime(5)])
```

C.2. <u>Sub-routine program code</u>

C.2.1. Headset transfer filter estimation code

```
% Program code to estimate headset z-transform.

% Data has to be loaded in to variables IN and OUT before
% the function can run.

function [TPhi] = Headset_transfer_estimation(P,Z,gain,LS_samples,Wanted_sample_rate);

%% The Data load function must be run before to acquire
% the data in question

[IN,OUT,Fs]=dataload;

% Data is now loaded and can be manipulated.
% Input data Bartlett estimated transfer.
% To speed up calculations a low resolution Bartlett
% estimation is done to determine the system transfer.

Sample_rate            = Fs;
IN                     = RESAMPLE(IN,Wanted_sample_rate,Sample_rate);
OUT                    = RESAMPLE(OUT,Wanted_sample_rate,Sample_rate);
max_sample_length      = length(OUT);
Sample_rate            = Wanted_sample_rate;

[F_IN,IN_freqscale]    = Bart_esst(IN',Sample_rate);
[F_OUT,OUT_freqscale]  = Bart_esst(OUT',Sample_rate);

Attenuation            = 10*log10(F_OUT./F_IN);

figure(1)
plot(IN_freqscale(1,2:length(IN_freqscale)),Attenuation(1,2:length(IN_freqscale)))
hold on
title('Measured headset transfer(blue) vs. simulated headset transfer(red)')
ylabel('Amplification (dB)')
xlabel('Frequency (Hz)')


% Transfer estimation with Least Squares estimation

IN              = IN(1:LS_samples);
OUT             = OUT(1:LS_samples);

PHI=[];
lengte          = length(IN)-1;
val             = max([Z+1 P]);

for n=0:Z
  PHI           = [PHI IN(val-n:lengte-n)];
end
for n=0:P-1
  PHI           = [PHI -OUT(val-n:lengte-n)];
end

OUT             = OUT((val+1):(length(IN)));
Theta           = inv(PHI'*PHI)*PHI'*OUT;
T               = 1/Sample_rate;
t               = T:T:length(OUT)*T;

A               = gain*[Theta(1:Z+1)'];
B               = [1 Theta((Z+2):length(Theta))'];

TPhi            = [A [Theta((Z+2):length(Theta))]']';
sys             = tf(A,B,T);

% Filter test

FilterIN        = randn(1,max_sample_length);
Sample_rate     = Wanted_sample_rate;
FilterOUT       = filter(A,B,FilterIN);

[F_IN,IN_freqscale]     = Bart_esst(FilterIN,Sample_rate);
[F_OUT,OUT_freqscale]   = Bart_esst(FilterOUT,Sample_rate);

Attenuation        = 10*log10(F_OUT./F_IN);
```

```
figure(1)
plot(OUT_freqscale(1,2:length(OUT_freqscale)),Attenuation(1,2:length(OUT_freqscale)),'r')
hold on
grid on
```

C.2.1.1. Data loading of recorded noise files

```
%Data load
function [IN,OUT,Fs]=dataload;

[IN,Fs]=wavread('WhiteN_Elno_in.wav');
[OUT,Fs]=wavread('WhiteN_Elno_out.wav');

max_data_length=length(OUT);
IN=IN(1:max_data_length);
```

C.2.1.2. Subroutines for headset transfer filter estimation code

Bartlett averaging of frequency spectrum

```
function [Average_in,Frequency_scale_in]=Bart_esst(Y,Sample_rate)
clear Average_in

samples          = length(Y);
Windows          = 100;                    % Ensures a 0.5 Hz resolution

if Windows <= 1
   disp('Amount Wimdows to small')
end

In              = (Y(1:samples));

Samples_per_window      = fix(length(In)/Windows);
Total_Samples           = Windows*Samples_per_window;
Average_in              = zeros(1,Samples_per_window);

for count = 1:Windows;

in              = In([((count-1)*Samples_per_window)+1]:[count*Samples_per_window]);
Fourier_of_in   = (1/Samples_per_window).*abs([fft(in)].^2);
Average_in      = Average_in+Fourier_of_in;

end

Average_in              = (1/Windows).*Average_in;
Average_in              = Average_in(1:fix(length(Average_in)/2));
Frequency_scale_in      =
Sample_rate/Samples_per_window:Sample_rate/Samples_per_window:length(Average_in)*(Sample_rate/Samp
les_per_window);

%figure
%plot(Frequency_scale_in,Average_in)
```

C.2.2. Simulated ANR performance measurement calculation code

```
function [Attenuation,Frequency_scale_out,Frequency_scale_in] =
ANR_Preformance(Sample_rate,Windows,begin_offset);
```

```
%Data load
function [samples,IN,OUT] = dataload(begin_offset);

[IN,Fs]                    =  wavread('output_noise1.wav');
OUT                        =  wavread('output1.wav');
samples                    =  length(OUT);

IN                         =  IN(begin_offset:samples);
OUT                        =  OUT(begin_offset:samples);

samples                    =  samples-begin_offset;

In                         = rot90(IN(1:samples));
Out                        = rot90(OUT(1:samples));

Samples_per_window         = fix(length(In)/Windows);
Total_Samples              = Windows*Samples_per_window;
Average_in                 = zeros(1,Samples_per_window);
Average_out                = zeros(1,Samples_per_window);

for count = 1:Windows;

in              = In([((count-1)*Samples_per_window)+1]:[count*Samples_per_window]);
out             = Out([((count-1)*Samples_per_window)+1]:[count*Samples_per_window]);
Fourier_of_in   = (1/Samples_per_window).*abs([fft(in)].^2);
Fourier_of_out  = (1/Samples_per_window).*abs([fft(out)].^2);

Average_in      = Average_in+Fourier_of_in;
Average_out     = Average_out+Fourier_of_out;

end

Average_in      = (1/Windows).*Average_in;
Average_in      = Average_in(1:fix(length(Average_in)/2));
Frequency_scale_in =
Sample_rate/Samples_per_window:Sample_rate/Samples_per_window:length(Average_in)*(Sample_rate/Samp
les_per_window);

Average_out     = (1/Windows).*Average_out;
Average_out     = Average_out(1:fix(length(Average_out)/2));
Frequency_scale_out =
Sample_rate/Samples_per_window:Sample_rate/Samples_per_window:length(Average_out)*(Sample_rate/Sam
ples_per_window);

Attenuation     = 10*log10(Average_out./Average_in);
```

C.2.3. Signal variance calculation

```
function [var]=real_time_variance(uniform_rand)

sum_uniform_rand = 0;
sum_squared_uniform_rand=0;
k=0;

for n=1:length(uniform_rand)

  k=k+1;
  sum_uniform_rand        =  uniform_rand(k) + sum_uniform_rand;
```

```
    sum_squared_uniform_rand = (uniform_rand(k))^2 + sum_squared_uniform_rand;

    first_moment        = sum_uniform_rand/k;
    second_moment       = sum_squared_uniform_rand/k;

    variance = second_moment-(first_moment)^2;

end


    var=variance;
```

Appendix D

D Program code for ANR implementation on the DSP56311EVM and conversion system B interface

```
;****************************************************************************
;     Single channel FIR Normalised LMS algorithm implementation
;
;
;     By Corné J. Smith
;
;
;
;****************************************************************************
      nolist
      include 'ioequ.asm'
      include 'intequ.asm'
      include 'ada_equ.asm'
      include 'vectors.asm'
      list

;****************************************************************************
;Buffers for talking to the CS4218
;****************************************************************************

Fil_order       equ     100                            ; Filter order specified
fil_K           equ     $00000a                        ; filter coefficients        y:
DataR           equ     $000078                        ; DataR save location 1      x:
Step            equ     0.0002                         ;LMS step size parameter
delay           equ     2

          org    x:0
RX_BUFF_BASE            equ     *
RX_data_1_2             ds      1       ; data time slot 1/2 for RX ISR (left audio)
RX_data_3_4             ds      1       ; data time slot 3/4 for RX ISR (right audio)

TX_BUFF_BASE            equ     *
TX_data_1_2             ds      1       ; data time slot 1/2 for TX ISR (left audio)
TX_data_3_4             ds      1       ; data time slot 3/4 for TX ISR (right audio)

RX_PTR                  ds      1       ; pointer for RX buffer
TX_PTR                  ds      1       ; pointer for TX buffer

Mu_error                ds      1       ; Step times error buffer
Fil_order_var           ds      1       ; Variable filter order

A_Fil_out               ds      1       ; Adaptive filter output
R_Fil_out               ds      1       ; Real filter output
E_out                   ds      1       ; Error signal output

CTRL_WD_12              equ    MIN_LEFT_ATTN+MIN_RIGHT_ATTN+LIN2+RIN2
CTRL_WD_34              equ    MIN_LEFT_GAIN+MIN_RIGHT_GAIN
```

```
        org    y:fil_K

        include 'zero100.asm'
        ;include 'test_band_pass.asm'

;*****************************************************************************
;Main Program
;*****************************************************************************

     org    p:$400
START
     movep  #$040006,x:M_PCTL            ; PLL 7 X 12.288 = 86.016MHz
     movep  #$012421,x:M_BCR             ; AARx - 1 wait state
     ori    #3,mr                        ; mask interrupts
     movec  #0,sp                        ; clear hardware stack pointer
     move   #0,omr                       ; operating mode 0
     move   #$40,r6                      ; initialise stack pointer
     move   #-1,m6                       ; linear addressing

; -------- Host port I/O init -------------

        movep   #$FFFF,x:M_HDDR          ; Host Data Direction Register (all outputs)
        movep   #0,x:M_HCR               ; Host Control Register (all interrupts disabled)
        movep   #1,x:M_HPCR              ; Host Polarity Control Register (GPIO active)

; -------- SCI I/O init          -----------------

        bclr    #0,x:M_PCRE              ; SCI Control register (PC0 GPIO active)
        bclr    #1,x:M_PCRE              ; SCI Control register (PC1 GPIO active)
        bset    #0,x:M_PRRE              ; SCI Direction Register (PC0 output)
        bset    #1,x:M_PRRE              ; SCI Direction Register (PC1 output)

; -------- Port A I/O init       -----------------

        movep   #$00CB20,x:M_AAR3        ; Set up port A from y:$00CXXX

; -------- ESSI port inits -----------------

        jsr   ada_init                   ; initialise codec

        jsr      clear_data_space

        bset     #0,x:M_PDRE             ; A/D control pin (CS_b pin)
        bset     #1,x:M_PDRE             ; Conversion clock(Conv)
        bset     #15,x:M_HDR             ; A/D control pin (CS_a pin)

loop
     jset   #3,x:M_SSISR0,*              ; wait for RX frame sync
     jclr   #3,x:M_SSISR0,*              ; wait for RX frame sync

;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;                  Outside Reference Microphone Calculations
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

        jsr      AtoDX                   ; Receive Outside mic and inside mic signal



;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;                          Calculate new ANR output
```

```
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

          move    #DataR+Fil_order+delay,r2          ; load data count back address in r2
          move    #DataR+Fil_order+delay-1,r1        ; load data count back address in r1
          jsr   Data_arrange_for_filter             ; Arrange data for filtering
          move    a,x:(DataR)                       ; Place new data in data space

          move    #DataR,r0                         ; load data start address
          move    #fil_K,r4                         ; load filter coefficients

          jsr     Filter                            ; Filter ref mic (gives output in a)

          neg     a
          nop

          move   x:RX_BUFF_BASE,b                   ; Disturbance signal
          nop
          add     b,a
          nop
     move         a,x:A_Fil_out                     ; Negative Adaptive filter Save output
          jsr     DtoA                              ; Output value to headset speaker

;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;                              Error Gain Calculation
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
          jsr     AtoDE
          nop
          move   b,x:TX_BUFF_BASE                   ; Recording
          move   b,x:TX_BUFF_BASE+1                 ; Recording

;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;                              Error Signal Save
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

          move    b,x:E_out
          nop
          move    x:E_out,b
          nop
          move    b,x1
          move    #Step,y1

          mpyr    x1,y1,a                           ; step times error(n)
          nop

     move    a,x:Mu_error                           ; Save Error mic*step

          move    #DataR+delay,r0                   ; load data start address
          nop
          jsr     Data_squared

          move    b,x0
          move    x:Mu_error,a                      ; Get Error mic*step

          jsr Devide                                ; [step*error(n)]/{[DataR]*[DataR]'+0.1}
          move    a,x0                              ; Save final Coefficient gain


;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;                              Filter Coefficient update
```

```
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

        move    #DataR+delay,r0         ; load Data start address
        move    #fil_K,r4               ; load filter Coefficients start address

        jsr Coef_update

    jmp    loop

;***************************************************************************
;$$$$$$$$$$$$$$$$$$$$$$$ FUNCTIONS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;***************************************************************************
clear_data_space                        ; Function clears data space where
        move    #DataR,r0               ; data samples will be stored
        move    #$000000,a

        rep #Fil_order+2
        move  a,x:(r0)+
        rts
;***************************************************************************
Data_arrange_for_filter

        do #Fil_order+delay,_end_arrange    ; reposition data for next entry

        move    x:(r1)-,b
        nop
        move    b,x:(r2)-
_end_arrange
        rts
;***************************************************************************
Filter

        clr a                   x:(r0)+,x0  y:(r4)+,y0      ; clear filter output;
        rep #Fil_order                                      ; loop filter multiplication
        mac     x0,y0,a         x:(r0)+,x0  y:(r4)+,y0
        rts

;***************************************************************************
Coef_update

        move  x:(r0)+,x1                ; Data sample   [u(n)]
        move    y:(r4),a                ; Filter Coefficient [H(n)]
        nop

        do #Fil_order,_end_update

        mpyr    x1,x0,b         x:(r0)+,x1          y:(r4),a
        add     a,b
        nop
        move    b,y:(r4)+       ; H(n+1) = H(n) + K*u(n) where K=k*error(n)

_end_update
        rts
;***************************************************************************
Data_squared

        clr b       x:(r0)+,a               ; Get input data at x0:r0 to square

        do #Fil_order-1,_end_data_s                 ; square all data points and add
        move  a,x0
        mac    x0,x0,b          x:(r0)+,a
        nop
_end_data_s

        move  a,x0
        macr   x0,x0,b                      ; Complete square function
        add    #0.00000001,b                ; Add small value to ensure not-zero
        rts

;*********************************************************************************
Devide

        move a,b
        div x0,b                    ; reset condition code register ??? Don't know how else
        rep #24
        div x0,a
        move    a0,b                ; Check sign changes and corrects sign
        nop
        move    b,a
        move    x0,b
        nop
        jclr    #23,b,not_neg       ; If Denominator negative jump to
        neg a
not_neg
        rts

;*************************************************************************************
DtoA

Anti_2s_compliment

        move    a2,x0               ; Change the DSP 2's compliment value
        jset    #1,x0,negative_2_comp   ; to not 2's compliment for D/A

        lsr     #10,a               ; shift right since host port reads
        nop                         ; least significant 14 bits
        add     #$002000,a          ; 2's compliment compensation for positive values
        nop
        jmp     end_2_comp

negative_2_comp

        sub     #$7FFF00,a          ; 2's compliment compensation for negative values
        nop
        lsr     #10,a               ; shift right since host port reads

end_2_comp
        nop
        nop


        or      #$008000,a
        move   a1,x:M_HDR           ; Write out value of A to the host port
        bclr    #14,x:M_HDR         ; Clock the CS of D/A low
        rep #90                     ; Clock the CS of D/A low
        nop
        bset    #14,x:M_HDR         ; Clock the CS of D/A high again

        rts
```

```
;*****************************************************************************

AtoDX

          bclr      #1,x:M_PDRE              ; Conversion clock (Conv)

          ; First A/D start conversion

          bclr      #15,x:M_HDR              ; A/D control pin (CS_a pin)
          rep #10
          nop
          bset      #15,x:M_HDR               ; A/D control pin (CS_a pin)

          bset      #1,x:M_PDRE              ; Conversion clock (Conv)

          ; Conversion Wait

          do #8,_end
          rep #90
          nop
_end
          ; Read data A to D 1

          bclr      #15,x:M_HDR               ; A/D control pin  (CS_a pin)
          rep #5
          nop
          bset      #0,x:M_BCR                ; Set up to Read from Port A
          move      y:$00C00C,a               ; Read from Port A
          bclr      #0,x:M_BCR                ; Put Port A off
          bset      #15,x:M_HDR              ; A/D control pin (CS_a pin)

          sub       #$000800,a                ; Subtract unwanted DC offset

          rts

;*****************************************************************************

AtoDE

          bclr      #1,x:M_PDRE              ; Conversion clock (Conv)

          ; Second A/D start conversion

          bclr      #0,x:M_PDRE               ; A/D control pin (CS_b pin)
          rep #10
          nop
          bset      #0,x:M_PDRE              ; A/D control pin (CS_b pin)

          bset      #1,x:M_PDRE              ; Conversion clock (Conv)

          ; Conversion Wait

          do #8,_end
          rep #90
          nop
_end
          ; Read data A to D 2

          bclr      #0,x:M_PDRE              ; A/D control pin (CS_b pin)
```

```
        rep #5
        nop
        bset    #0,x:M_BCR              ; Set up to Read from Port A
        move    y:$00C00B,b             ; Read from Port A
        bclr    #0,x:M_BCR              ; Put Port A off
        bset    #0,x:M_PDRE             ; A/D control pin (CS_b pin)

        sub     #$000500,b              ; Subtract unwanted DC offset

        rts
;*******************************************************************************

     include 'ada_init.asm'
      end
```