Nonlinear Acoustic Echo Cancellation for Mobile Phones: A Practical Approach
Abstract
Acoustic echo cancelation (AEC) composes a fundamental property of speech processing to enable a pleasant telecommunication conversation. Without this property of the telephone the communicator would hear an annoying echo of his own voice along with the speech from the other communicator. This would make a conversation through any telecommunication device an unpleasant experience.

AEC has been subject of interest since 1950s in the telecom industry and very efficient solutions were devised to cancel linear echo. With the advent of low cost hands free communication devices the issue of non linear echo became prominent because these devices use cheap loudspeakers that produce artifacts in addition to the desired sound which will cause non linear echo that cannot be cancelled by linear echo cancellers.

In this thesis a Harmonic Distortion Residual Echo Cancelation algorithm has been chosen for further investigations (HDRES). HDRES has many of those features that are desirable for an algorithm which is dealing with nonlinear acoustic echo cancelation, such as low computational complexity and fast convergence. The algorithm was first implemented in Matlab where it was tested and modified. The final result of the modified algorithm was then implemented in C and integrated with a complete AEC system. Before the implementation a number of measurements were done to distinguish the nonlinearities that were cause by the mobile phone loudspeaker. The measurements were performed on three different mobile pones which were documented to have problems with nonlinear acoustic echo.

The result of this thesis has shown that it might be possible to use an adaptive filter, which has both low complexity and fast convergence, in an operating AEC system. However, the request for such a system to work would be that a doubletalk detector is implemented along with the adaptive algorithm. That way the doubletalk situation could be found and the adaptation of the algorithm could be stopped. Thus, the major part of the speech would be saved.
Acknowledgment
First and foremost we would like to express our sincere gratitude to our supervisor Jonas Lundbäck at ST-Ericsson for his constant support and timely guidance throughout the thesis. His guidance in showing us ways to apply our theoretical knowledge in the practical constrained environment was priceless.

We are also very grateful to Professor Sven Nordebo for his knowledgeable ideas and encouragement. We are indebted to audio department staffs of ST-Ericsson for their unreserved support and in providing a comfortable working atmosphere with a friendly approach.

Finally we want to thank all who have been with us to reach at this level and gave us the opportunity to work with an interesting practical thesis in the industry.
### Variable notation

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x(n)$</td>
<td>Loudspeaker input signal in time domain</td>
<td>Vector</td>
</tr>
<tr>
<td>$y(n)$</td>
<td>Microphone input signal in time domain</td>
<td>Scalar</td>
</tr>
<tr>
<td>$X(n)$</td>
<td>Loudspeaker input signal in frequency domain</td>
<td>Vector</td>
</tr>
<tr>
<td>$Y(n)$</td>
<td>Microphone input signal in frequency domain</td>
<td>Scalar</td>
</tr>
<tr>
<td>$h(n)$</td>
<td>Adaptive linear filter</td>
<td>Vector</td>
</tr>
<tr>
<td>$w_1(n)$</td>
<td>Linear kernel of Volterra filter</td>
<td>Vector</td>
</tr>
<tr>
<td>$w_2(n)$</td>
<td>Quadratic kernel of Volterra filter</td>
<td>Vector</td>
</tr>
<tr>
<td>$\hat{y}(n)$</td>
<td>Estimated acoustic echo in time domain</td>
<td>Scalar</td>
</tr>
<tr>
<td>$\hat{Y}(n)$</td>
<td>Estimated acoustic echo in frequency domain</td>
<td>Scalar</td>
</tr>
<tr>
<td>$\hat{y}_l(n)$</td>
<td>Output of linear filter</td>
<td>Scalar</td>
</tr>
<tr>
<td>$\hat{y}_v(n)$</td>
<td>Output of Volterra filter</td>
<td>Scalar</td>
</tr>
<tr>
<td>$e_v(n)$</td>
<td>Error from Volterra filter</td>
<td>Scalar</td>
</tr>
<tr>
<td>$e_l(n)$</td>
<td>Error from linear filter</td>
<td>Scalar</td>
</tr>
<tr>
<td>$e(n)$</td>
<td>Error from combined filter (overall error)</td>
<td>Scalar</td>
</tr>
<tr>
<td>$\mu_{vl}$</td>
<td>Step size of linear kernel in Volterra filter</td>
<td>Scalar</td>
</tr>
<tr>
<td>$\mu_{vq}$</td>
<td>Step size of quadratic kernel in Volterra filter</td>
<td>Scalar</td>
</tr>
<tr>
<td>$\mu_l$</td>
<td>Step size of linear filter</td>
<td>Scalar</td>
</tr>
<tr>
<td>$\lambda(n)$</td>
<td>Weighting parameter</td>
<td>Scalar</td>
</tr>
<tr>
<td>$q(n)$</td>
<td>Output signal of HDRES in time domain</td>
<td>Scalar</td>
</tr>
<tr>
<td>$Q(n)$</td>
<td>Output signal of HDRES in frequency domain</td>
<td>Scalar</td>
</tr>
<tr>
<td>$w(p; \chi)$</td>
<td>Nonlinear modelling</td>
<td>Vector</td>
</tr>
<tr>
<td>$b(n)$</td>
<td>Basis functions</td>
<td>Vector</td>
</tr>
<tr>
<td>$S(n)$</td>
<td>Matrix of the bias function</td>
<td>Matrix</td>
</tr>
<tr>
<td>$c(n)$</td>
<td>Signal after non linear model</td>
<td>Vector</td>
</tr>
<tr>
<td>$\theta(n)$</td>
<td>Non linear coefficient</td>
<td>Vector</td>
</tr>
<tr>
<td>$d(n)$</td>
<td>Output of shortening filter</td>
<td>Scalar</td>
</tr>
<tr>
<td>$d(n)$</td>
<td>Estimated shortening filter output</td>
<td>Scalar</td>
</tr>
<tr>
<td>$z(n)$</td>
<td>Shortening filter</td>
<td>Vector</td>
</tr>
<tr>
<td>$(\cdot)^T$</td>
<td>Transpose operation</td>
<td></td>
</tr>
<tr>
<td>$(\cdot)^H$</td>
<td>Hermitian transpose operation</td>
<td></td>
</tr>
</tbody>
</table>
Symbol explanations

- Loudspeaker
- Microphone
- Multiplication of two or more signals
- Addition of two or more signals
- Subtraction of two signals
- Adaptive filter
- Update of an Adaptive filter
List of figures
Figure 1.1 Acoustic Echo in hands free communication ................................................................. 3
Figure 1.2 Block diagram of acoustic echo canceller ..................................................................... 4
Figure 1.3 Adaptive filter in acoustic echo canceller ......................................................................... 5
Figure 1.4 Simplified figure of the function of a loudspeaker ........................................................... 12
Figure 1.5 Sinusoids with different frequencies. From above: 25, 50 and 75 Hz respectively ......... 13
Figure 1.6 Summation of the three sinusoids ................................................................................... 14
Figure 1.7 The corresponding signal of figure 1.6 in frequency domain ........................................... 14
Figure 1.8 Block diagram of acoustic echo control ........................................................................... 16
Figure 2.1 An illustration of the measurement setup .......................................................................... 18
Figure 2.2 Capturing measurement data ............................................................................................ 18
Figure 2.3 Hands free mode .............................................................................................................. 19
Figure 2.4 External microphone ......................................................................................................... 19
Figure 2.5 Frequency sweep with decreasing amplitude, time domain plot .................................... 20
Figure 2.6 Pulses of white cyclo stationary noise, time domain ......................................................... 20
Figure 2.7 Loudspeaker input. Frequency sweep with a high amplitude, frequency domain .......... 21
Figure 2.8 Microphone input. Frequency sweep with a high amplitude, frequency domain ............ 21
Figure 2.9 Loudspeaker input. Frequency sweep with a low amplitude, frequency domain .......... 21
Figure 2.10 Microphone input. Frequency sweep with a low amplitude, frequency domain .......... 21
Figure 2.11 Loudspeaker input. Frequency sweep with a high amplitude, frequency domain .......... 22
Figure 2.12 Microphone input. Frequency sweep with a high amplitude, frequency domain .......... 22
Figure 2.13 Microphone input. Frequency sweep with a low amplitude, frequency domain .......... 22
Figure 2.14 Loudspeaker input. Frequency sweep with a low amplitude, frequency domain .......... 22
Figure 2.15 Loudspeaker input. White noise, frequency domain ..................................................... 22
Figure 2.16 Microphone input. White noise, frequency domain ......................................................... 22
Figure 3.1 Combination of filters scheme .......................................................................................... 31
Figure 3.2 Simplified combination of kernels scheme ........................................................................ 32
Figure 3.3 Nonlinear AEC – Hammerstein model (adaptive) ............................................................... 34
Figure 3.4 AEC nonlinear Hammerstein design together with a shortening filter ............................... 35
Figure 3.5 A illustration of the architecture of a shortening filter ..................................................... 35
Figure 3.6 A simplified illustration of how the HDRES and the linear AEC works together ............. 39
Figure 3.7 Illustrates the architecture of the HDRES model together with a linear AEC .................. 40
Figure 4.1 Flowchart of the HDRES model ...................................................................................... 45
Figure 4.2 Table of the mapping matrix for the first overtone ............................................................. 48
Figure 4.3 Table of the mapping matrix for the first overtone ............................................................. 49
Figure 4.4 The input signal created for testing of the HDRES algorithm in Matlab ......................... 50
Figure 4.5 Plot of the true residual echo ......................................................................................... 51
Figure 4.6 Plot of the estimate of the true residual echo ................................................................. 51
Figure 4.7 The error function, gives the difference between the true residual echo and its estimation .......................................................... 52
Figure 4.8 Block diagram of acoustic echo canceller ....................................................................... 55
Figure 5.1 Illustration of the loudspeaker input signal ................................................................. 59
Figure 5.2 Illustration of the microphone input signal ...................................................................... 59
Figure 5.3 Illustration of the output of the linear AEC ..................................................................... 59
Figure 5.4 Illustration of the output of the HDRES algorithm .......................................................... 60
Figure 5.5 Illustration of the output of the static method output ........................................................ 60
Figure 5.6 Illustration of the output of the linear AEC magnified ...................................................... 60
Figure 5.7 Illustration of the output of the HDRES output magnified ................................................ 61
Figure 5.8 Illustration of the output of the static output magnified .................................................... 61
Figure 5.9 Illustration of the loudspeaker input signal. It contains speech with different amplitudes ........................................................................................................ 61
Figure 5.10 Illustration of the microphone input signal .................................................................... 62
Figure 5.11 Illustration of the output of the linear AEC .................................................................... 62
Figure 5.12 Illustration of the output of HDRES .............................................................................. 62
Figure 5.13 Illustration of the static output ......................................................................................... 62
Figure 5.14 Illustration of the HDRES output magnified ................................................................ 63
Figure 5.15 Illustration of the static output magnified ................................................................. 63
List of tables
Table 3.1 Frequency bands for narrow band speech with their frequency range ................................................. 26
Table 3.2 The result of calculating harmonics for the 6th band ...................................................................... 27
Table 3.3 Complete mapping table .................................................................................................................. 28
Table 4.1 Extract of the Matlab code for the calculation of the frequency band ........................................... 46
Table 4.2 Extract of the Matlab code for the call of the mapping matrix ....................................................... 46
Table 4.3 Extract of the Matlab code for the calculation of the estimation ................................................... 46
Table 4.4 Extract of the Matlab code for the averaging calculation ................................................................. 47
Table 4.5 Limiting the weights ......................................................................................................................... 47
Table 4.6 A part of the Matlab code that calculated the MappingMatrix ........................................................ 47
Table 4.7 Residual echo estimation ................................................................................................................. 54
Table 4.8 Updating of the weights .................................................................................................................. 55
Table 4.9 Double to SPL type conversion ...................................................................................................... 56
# Table of contents

Abstract .................................................................................................................................................. i

Acknowledgement ................................................................................................................................. ii

Variable notation ................................................................................................................................... iii

Symbol explanations ................................................................................................................................. iv

List of figures ........................................................................................................................................... v

List of tables ........................................................................................................................................... vii

CHAPTER 1 INTRODUCTION ................................................................................................................... 1

1.1 Report organisation .......................................................................................................................... 2

1.2 Acoustic echo cancellation ................................................................................................................ 3

1.2.1 Types of echo ............................................................................................................................... 3

1.2.2 Echo cancellation .......................................................................................................................... 4

1.2.3 LEM System ............................................................................................................................... 4

1.2.4 Adaptive Filter ........................................................................................................................... 4

1.2.5 Non linear Processor (NLP) ....................................................................................................... 4

1.3 Adaptive filter algorithms ................................................................................................................ 5

1.3.1 The Steepest Descent method ..................................................................................................... 6

1.3.2 Least Mean Square (LMS) algorithm ......................................................................................... 7

1.3.3 Normalized LMS (NLMS) Algorithm ....................................................................................... 8

1.3.4 Recursive Least Squares (RLS) algorithm .................................................................................. 9

1.4 Loudspeaker nonlinear distortion .................................................................................................... 12

1.4.1 Loudspeaker description ............................................................................................................ 12

1.4.2 Nonlinear distortion .................................................................................................................... 12

1.4.3 Origin of the nonlinearities ......................................................................................................... 13

1.4.4 Harmonics ................................................................................................................................. 13

1.5 Modelling physical systems ............................................................................................................ 15

1.5.1 Data collection ........................................................................................................................... 15

1.5.2 Selecting model structure ......................................................................................................... 15

1.5.3 Parameter estimation .................................................................................................................. 15

1.5.4 Model validation ......................................................................................................................... 15

1.6 Description of Acoustic Echo Control Implementation .................................................................... 16

1.6.1 Overview of Acoustic Echo Control Implementation ................................................................. 16

CHAPTER 2 MEASUREMENTS ................................................................................................................. 17

2.1 Measurement procedure .................................................................................................................. 18

2.1.1 Measurement type 1: Hand held mode ....................................................................................... 19

2.1.2 Measurement type 2: Hands free mode ..................................................................................... 19

2.1.3 Measurement type 3: Measurement with an external microphone ......................................... 19

2.2 Measurement analysis .................................................................................................................... 20

2.2.1 Input signals .............................................................................................................................. 20
CHAPTER 1 INTRODUCTION
1.1 Report organisation

The thesis report has the following content:

In chapter 1 introduction to the subject matter where the concept of acoustic echo cancelling and loudspeaker nonlinearities are explained.

In chapter 2 a description of the performed measurements is done. It includes among others the measurement setup and the discoveries that were made.

Chapter 3 gives a deeper description of the different methods to cancel nonlinear acoustic echo that were chosen for further investigation.

In chapter 4 the implementation of the choosen algorithm in Matlab and in C is presented. It includes descriptions of important sections of the algorithm, implementation difficulties and testing results.

In chapter 5 results of the thesis are presented. The results are shown by plots and explanation of the plots according to the expected theoretical result.

In chapter 6 the conclusion and the further work is stated.
1.2 Acoustic echo cancellation
Webster dictionary defines echo as: “A wave that has been reflected or otherwise returned with sufficient magnitude and delay to be detected as a wave distinct from that directly transmitted.”
Echo in a telecommunication system degrades the quality and intelligibility of voice communication. Although echo is used in detection and navigation applications such as radar and infrared imaging as a useful phenomenon, it’s undesirable in communications systems. Hence efforts are done to reduce it as much as possible.

The effect of echo on voice communication is dependent on amplitude and time delay. A delay above 20ms is considered as annoying while more than 200ms is disruptive [3].

1.2.1 Types of echo
There are two types of echo in Telecommunication

1. Hybrid Echo
   Hybrid echo is an electrical echo generated by public-switched telephone network (PSTN) due to impedance mismatch between the two wire subscriber line and four wire long distance telephone lines.

2. Acoustic echo
   Acoustic echo is observed when sound from a loudspeaker is picked up by microphone in the same device and sent back to the original source of the sound with significant delay. If the delay is small, then the acoustic echo is perceived as soft reverberation. But if the delay is too large it will be annoying and disruptive hence should be cancelled. Acoustic echo is mostly experienced in hands free communications as shown in figure 1.1 [1].

![Figure 1.1 Acoustic Echo in hands free communication](image-url)
1.2.2 Echo cancellation

The endeavor to cancel echo started in the late 1950s. Primitive echo suppressors use voice activated switches to turn off the receiving line allowing voice to pass on the transmitting line only. They effectively block echo, but have a problem of allowing only half-duplex communication where only one end is allowed to talk at a time. Line echo cancellers were constructed to remove electrical echo from telephone lines and are very efficient [11].

However the problem of acoustic echo cancellation is a more challenging task since the acoustic path is not known at priori. Moreover it changes constantly. That’s why development of adaptive methods was crucial for this task. Acoustic echo cancellers (AEC) try to cancel echo by adaptively modeling the loudspeaker-enclosure-microphone (LEM) system and hence estimating the acoustic echo by using loudspeaker input signal coming from far end to be subtracted from the microphone input signal. Figure 1.2 shows block diagram of simplified acoustic echo canceller which consists of three components [14].

![Figure 1.2 Block diagram of acoustic echo canceller](image)

1.2.3 LEM System

The loudspeaker-enclosure-microphone system includes a loudspeaker, a microphone and the direct and reflected acoustic echo paths of the echo. The performance of the AEC depends on how well the adaptive filter modelled the LEM system, which directly dictates how close the estimated echo is to the real microphone input echo [14].

1.2.4 Adaptive Filter

This is the crucial component of the acoustic echo canceller that estimates echo adaptively by keeping track of the changes in room acoustics. For a very good LEM system which does not introduce nonlinear artifacts to the loudspeaker input signal the echo can be considered as a delayed and attenuated version of the loudspeaker input signal. In such cases a linear AEC can accurately estimate the echo by using linear adaptive filters. Generally the LEM is modelled as a linear system which explains why linear adaptive filters are usually used at this position of the acoustic echo canceller. An explanation of the most commonly used linear adaptive algorithms, LMS and RLS, is given in section 1.3 [14].

1.2.5 Non linear Processor (NLP)

NLP in AEC is the block which handles non linear processing that cannot be done by the linear adaptive filter. Studies have shown that loudspeakers produce nonlinear artifacts. Consequently linear adaptive filters cannot estimate these nonlinearities and hence such echo will not be cancelled if only the usual linear adaptive filters are used. Residual echo control in NLP tries to suppress residual echo by modeling the non linearity in LEM system. The NLP then generates comfort noise if output power of residual echo control is lower than background noise level. Otherwise the far end may think the communication link is disconnected. Introduction to modeling of physical systems is given in section 1.5 [14].
1.3 Adaptive filter algorithms

For any type of adaptive filter applications the choice of adaptive algorithm plays an important role for the performance of the overall system. Two widely used adaptive algorithms are LMS and RLS. Both algorithms try to minimize the mean square error. The choice of algorithm depends on rate of convergence, amount of error at convergence, robustness of the algorithm in initial states, computational complexity and memory usage [3].

Before explaining the LMS and RLS algorithm, a description of the steepest descent algorithm is given since it is the base for derivation of LMS. Figure 1.3 shows the overall system with input, output and intermediate signals which will be used in subsequent sections.

The error and the estimated desired signals are defined as:

\[ e(n) = y(n) - \hat{y}(n) \quad (1.1) \]
\[ \hat{y}(n) = h^T(n)x(n) \quad (1.2) \]

Substituting equation (1.2) in equation (1.1)

\[ e(n) = y(n) - h^T(n)x(n) \quad (1.3) \]

The mean square error:

\[ E\{e^2(n)\} = E\{[y(n) - h^T(n)x(n)]^2\} \quad (1.4) \]
\[ E\{e^2(n)\} = E\{(y(n))^2 - 2h^T(n)y(n)x(n) + [h^T(n)x(n)]^2\} \quad (1.5) \]
\[ E\{e^2(n)\} = E\{(y(n))^2\} - 2h^T(n)y(n)x(n) + h^T(n)E\{x(n)x^T(n)\}h(n) \quad (1.6) \]

Computing the mean for each term in equation (1.6) results in:

\[ E\{e^2(n)\} = R_{yy} - 2h^T(n)R_{xy} + h^T(n)R_{xx}h(n) \quad (1.7) \]
where,
\( R_{yy} \) is autocorrelation matrix of the desired signal
\( R_{yx} \) is cross correlation of the reference and desired signal
\( R_{xy} \) is autocorrelation matrix of the reference signal

1.3.1 The Steepest Descent method

The Steepest Descent method is a method used to find the optimum solution (minimum point) of a function by taking small steps in the negative gradient direction [3]. In adaptive filtering applications the error, the difference between the estimated and desired signals, is expressed as a function of filter coefficients. Applying this method to the mean square error function results in optimum filter coefficients of the adaptive filter. Starting with initial filter coefficients, the method continuously updates the coefficients in a downward direction until a minimum point, where the gradient is zero, is reached.

The steepest descent method adaptation:

\[
h(n + 1) = h(n) + \mu(-\nabla E[e^2(n)]) \tag{1.8}\]

where,
\( E[e^2(n)] \) is the mean square error
\( \nabla E[e^2(n)] \) is gradient of the mean square error
\( \mu \) is the adaptation step size

From equation (1.7) one has the expression for the mean square error. Computing its gradient gives:

\[
\nabla E[e^2(n)] = -2R_{xy} + 2R_{xx}h(n) \tag{1.9}\]

The optimum filter coefficient \( (h_o) \) is found when the gradient is zero. Equating equation (1.9) to zero and solving for \( h \) gives:

\[
h_o = R_{xx}^{-1}R_{xy} \tag{1.10}\]

Substituting equation (1.9) in equation (1.8)

\[
h(n + 1) = h(n) + \mu(R_{xy} - R_{xx}h(n)) \tag{1.11}\]

Let us define filter coefficient error \( \tilde{h} \) as:

\[
\tilde{h}(n) = h(n) - h_o \tag{1.12}\]

Subtracting \( h_o \) from both sides of equation (1.12) and then substituting \( R_{xx} \) for \( R_{xy} \) finally using equation (1.12) yields:

\[
\tilde{h}(n + 1) = \tilde{h}(n) - 2\mu R_{xx}\tilde{h}(n) \tag{1.13}\]

The correlation matrix \( R_{xx} \) can be expressed in terms of eigen vectors and eigen values as follows:

\[
R_{xx} = Q\Lambda Q^T \tag{1.14}\]

where,
\( Q \) is orthonormal matrix of eigen vectors of \( R_{xx} \)
\( \Lambda \) is diagonal matrix with eigen values of \( R_{xx} \) as diagonal elements
Substituting equation (1.14) in equation (1.13):

\[ \tilde{h}(n + 1) = \tilde{h}(n) - 2\mu Q\Lambda Q^T\tilde{h}(n) \]  

(1.15)

Multiply both sides of equation (1.15) by \( Q^T \)

\[ Q^T \tilde{h}(n + 1) = Q^T \tilde{h}(n) - 2\mu Q\Lambda [Q^T]^2\tilde{h}(n) \]  

(1.16)

Let

\[ v(n) = Q^T \tilde{h}(n) \]  

(1.17)

Hence equation (1.16) becomes

\[ v(n + 1) = v(n) - 2\mu \Lambda v(n) \]  

(1.18)

Since \( \Lambda \) is a diagonal matrix, equation (1.19) can be expressed in terms of individual eigen vectors

\[ v_k(n + 1) = v_k(n) - 2\mu \lambda_k v_k(n) \]  

(1.19)

The solution for the recursion in equation (1.19) can be written as

\[ v_k(n + 1) = (1 - 2\mu \lambda_k)^k v_k(n) \]  

(1.20)

The convergence of equation (1.20) is guaranteed if the following condition is fulfilled

\[ |1 - 2\mu \lambda_k| < 1, \text{ for all } k \]  

(1.21)

Or equivalently, if

\[ \mu < \frac{1}{\lambda_k}, \text{ for all } k \]  

(1.22)

The convergence criterion of equation (1.22) is summarized as

\[ \mu < \frac{1}{\lambda_{\text{max}}} \]  

(1.23)

where \( \lambda_{\text{max}} \) is the largest eigen value

For ease of implementation the convergence criteria is given as:

\[ \mu < \frac{1}{\text{trace}(R_{xx})} \leq \frac{1}{\lambda_{\text{max}}} \]  

(1.24)

1.3.2 Least Mean Square (LMS) algorithm

LMS is a simple yet effective adaptive algorithm that is used in echo cancellation, channel equalization, adaptive noise cancellation, and time-delay estimation. LMS simplifies the Steepest Descent method by computing the gradient of instantaneous squared error function instead of the average squared error [3].

Step size determines the rate of convergence and amount of Mean Square Error (MSE) at convergence. A large step size will result in a fast rate of convergence but with high MSE, while a small step size results in a slow rate of convergence with minimum MSE.

The Steepest Descent adaptation in equation (1.8) with instantaneous squared error is:
\[ h(n + 1) = h(n) + \mu(-\nabla e^2(n)) \]  
\[ \text{(1.25)} \]

where,
\[ e(n) = y(n) - h^T(n)x(n) \]  
\[ \text{(1.26)} \]

Computation of the gradient is:
\[ \nabla e^2(n) = -2x(n)[y(n) - h^T(n)x(n)] \]  
\[ \text{(1.27)} \]

Substituting the error in equation (1.27)
\[ \nabla e^2(n) = -2x(n)e(n) \]  
\[ \text{(1.28)} \]

Inserting equation (1.27) in equation (1.25) gives the adaptation for LMS algorithm:
\[ h(n + 1) = h(n) + 2\mu x(n)e(n) \]  
\[ \text{(1.29)} \]

Incorporating the constant in the step size \( \mu \), the adaptation becomes
\[ h(n + 1) = h(n) + \mu x(n)e(n) \]  
\[ \text{(1.30)} \]

Summary of LMS algorithm

Given initial parameter \( h_0 \),
Repeat for \( n = 1,2,3, \ldots \)
\[ e(n) = y(n) - h^T(n)x(n) \]  
\[ \text{(1.31)} \]
\[ h(n + 1) = h(n) + \mu x(n)e(n) \]  
\[ \text{(1.32)} \]

1.3.3 Normalized LMS (NLMS) Algorithm

NLMS is a variant of the LMS algorithm that normalizes the adaptation step size according to the power of the input signal, so that the convergence of the LMS algorithm will not slow down by small signals and increase by large signals [4].

Consider the convergence criteria of the Steepest Descent method in equation (1.25). It relies on the autocorrelation matrix to set the step size. Practically the autocorrelation matrix is unknown. Therefore its approximate is calculated as:
\[ \text{trace}(R_{xx}) = (\alpha + 1)E\{|x(n)|^2\} \]  
\[ \text{(1.33)} \]

where,
\( \alpha \) is the size of \( R_{xx} \)
\( E\{|x(n)|^2\} \) is power of input signal

Power of the input signal can be estimated as:
\[ E\{|x(n)|^2\} \approx \frac{1}{\alpha+1} \sum_{k=0}^{\infty} |x(n-k)|^2 \]  
\[ \text{(1.34)} \]

Substituting equation (1.31) in equation (1.30), the convergence criteria becomes
\[ \mu < \frac{1}{\text{trace}(R_{xx})} = \frac{1}{\sum_{k=0}^{\infty} |x(n-k)|^2} = \frac{1}{x^H(n)x(n)} \]  
\[ \text{(1.35)} \]
Computing step-size in time one has:

$$\mu = \frac{\beta}{|x(n)|^2}$$  \hspace{1cm} (1.36)

where $\beta$ is normalized step size in the range $0 < \beta < 2$

To avoid division by zero in equation (1.36) a very small number (machine precision number) is added in the denominator:

$$\mu = \frac{\beta}{|x(n)|^2 + a}$$  \hspace{1cm} (1.37)

The adaptation algorithm for NLMS is found by substituting equation (1.37) in equation (1.29)

$$h(n + 1) = h(n) + \frac{\beta}{|x(n)|^2 + a} x(n) e(n)$$  \hspace{1cm} (1.38)

where $a$ is a small positive number and $\beta$ is the step size.

### 1.3.4 Recursive Least Squares (RLS) algorithm

RLS is an adaptive algorithm which tries to minimize the Mean Square Error (MSE). It is a time update version of the well-known wiener filter. It has fast convergence rate to the optimal solution, high performance for non-stationary signals and low minimum MSE at convergence. These attributes makes it suitable for speech enhancement, channel equalization, and echo cancellation applications [3][5].

The least square cost function as $f_n(h)$ is given by

$$f_n(h) = \sum_{k=1}^{n} |h^H x(k) - y(k)|^2 + (h - h_0)^H \lambda^n R_0 (h - h_0)$$  \hspace{1cm} (1.39)

where $R_0$ and $h_0$ are initial values

Define

$$f_0(h) = (h - h_0)^H R_0 (h - h_0)$$  \hspace{1cm} (1.40)

$$p_n(h) = R_0 h_0$$  \hspace{1cm} (1.41)

Introduce A and b as:

$$A = \left[ \begin{array}{c} x^H(1), \ldots, x^H(n) \end{array} \right]^T, \quad b = \left[ \begin{array}{c} y^*(1), \ldots, y^*(n) \end{array} \right]^T$$

Hence, the cost function in equation (1.39) can be written as:

$$f_n(h) = (Ah - b)^H (Ah - b) + (h - h_0)^H \lambda^n R_0 (h - h_0)$$  \hspace{1cm} (1.42)

The “forgetting factor”, $\lambda$, determines where the emphasis will be depending on how large $n$ is. The forgetting factor is neglected when $\lambda = 1$.

The solution for the cost function is found by differentiating equation (1.42) with respect to $h$:

$$h_n = R_n^{-1} p_n$$  \hspace{1cm} (1.43)

$$R_n = A^H A + \lambda^n R_0 = \sum_{k=1}^{n} \lambda^{n-k} x(k)x^H(k) + \lambda^n R_0$$  \hspace{1cm} (1.44)

$$p_n = A^H b + \lambda^n p_0 = \sum_{k=1}^{n} \lambda^{n-k} x(k)y^*(k) + \lambda^n p_0$$  \hspace{1cm} (1.45)
Rearranging terms and taking out \( \lambda \) as common factor, the recursive relation becomes:

\[
R_n = \sum_{k=1}^{n} \lambda^{n-k} x(k)x^H(k) + x(n)x^H(n) + \lambda^n R_0 \\
R_n = \lambda \left( \sum_{k=1}^{n} \lambda^{n-1-k} x(k)x^H(k) + \lambda^n R_0 \right) + x(n)x^H(n) \\
R_n = \lambda R_{n-1} + x(n)x^H(n)
\]  

(1.46)  
(1.47)  
(1.48)

Similarly, the recursive relation for \( p_n \):

\[
p_n = \sum_{k=1}^{n} \lambda^{n-k} x(k)y^*(k) + x(n)y^*(n) + \lambda^n p_0 \\
p_n = \lambda \left( \sum_{k=1}^{n-1} \lambda^{n-1-k} x(k)y^*(k) + \lambda^n p_0 \right) + x(n)y^*(n) \\
p_n = \lambda p_{n-1} + x(n)y^*(n)
\]  

(1.49)  
(1.50)  
(1.51)

Rewrite the recursive relation for \( R_n \) as:

\[
\lambda^{-1} R_n = R_{n-1} + x(n)\lambda^{-1}x^H(n)
\]  

(1.52)

To compute the inverse, use matrix inversion lemma stated below:

Matrix Inversion Lemma [6]

If \( A = B^{-1} + CD^{-1}C^H \),

Then \( A^{-1} = B - BC(CHBC + D)^{-1}C^HB \)

Let \( A = \lambda^{-1} R_n \), \( B^{-1} = R_{n-1} \), \( C = x(n) \) and \( D = \lambda^{-1} \)

After substitution the inverse will be:

\[
R_n^{-1} = \lambda^{-1} R_{n-1}^{-1} - \frac{\lambda^{-1} R_{n-1}^{-1} x(n)x^H(n)R_{n-1}^{-1}}{x^H(n)R_{n-1}^{-1}x(n)+\lambda}
\]  

(1.53)

Multiplying both sides by \( x(n) \) and then taking common denominator one gets the following important relation:

\[
R_n^{-1} x(n) = \lambda^{-1} R_{n-1}^{-1} x(n) - \frac{\lambda^{-1} R_{n-1}^{-1} x(n)x^H(n)R_{n-1}^{-1} x(n)}{x^H(n)R_{n-1}^{-1}x(n)+\lambda}
\]  

(1.54)  
\[
R_n^{-1} x(n) = \frac{R_{n-1}^{-1} x(n)}{x^H(n)R_{n-1}^{-1}x(n)+\lambda}
\]  

(1.55)

The recursive least square solution of equation (1.43) can now be calculated as:

\[
h_n = R_n^{-1} p_n = R_n^{-1} \left( \lambda p_{n-1} + x(n)y^*(n) \right)
\]  

(1.56)  
\[
h_n = R_n^{-1} \lambda p_{n-1} + R_n^{-1} x(n)y^*(n)
\]  

(1.57)

Substitute the value of \( R_n^{-1} \) from equation (1.54) to equation (1.57)

\[
h_n = \left( \lambda^{-1} R_{n-1}^{-1} - \frac{\lambda^{-1} R_{n-1}^{-1} x(n)x^H(n)R_{n-1}^{-1}}{x^H(n)R_{n-1}^{-1}x(n)+\lambda} \right) \lambda p_{n-1} + R_n^{-1} x(n)y^*(n)
\]  

(1.58)
\[ h(n) = h_{n-1} + R_{n-1}^{-1}x(n)e^*(n) \]

where,
\[ e(n) = y(n) - h_{n-1}^H x(n) \]

Summary of RLS algorithm:

Given initial parameters, \( h_0, R_0, R_0^{-1} \)

Repeat for \( n = 1, 2, 3, \ldots \)

\[ e(n) = y(n) - h_{n-1}^H x(n) \quad (1.60) \]

\[ R_n^{-1} = \lambda^{-1} R_{n-1}^{-1} - \frac{\lambda^{-1} R_{n-1}^{-1} x(n)x^H(n)R_{n-1}^{-1}}{x^H(n)R_{n-1}^{-1}x(n)+\lambda} \quad (1.61) \]

\[ h(n) = h_{n-1} + R_{n-1}^{-1}x(n)e^*(n) \quad (1.62) \]
1.4 Loudspeaker nonlinear distortion

1.4.1 Loudspeaker description
In resemblance with an electrical motor a loudspeaker is a device that converts electrical energy into mechanical energy. The mechanical energy is then transformed into sound waves. In a simplified sense, a loudspeaker consists of three major parts; the motor, the membrane (also called diaphragm) and the suspension. A simple way to describe the function of a loudspeaker is the following: (An illustration of the function of a loudspeaker is shown in figure 1.4.)

The membrane is fixed to a coil which is connected to electrical cables. Just behind the coil a permanent magnet is located. When the electricity is flowing through the cables into the coil it will become an electro-magnet. Depending on the direction of the electricity, the coil will either attract or repel the permanent magnet. Thus, the coil will start to move and the membrane will move with it. When the membrane moves the air starts to vibrate and there is sound [7].

![Illustration of the function of a loudspeaker](image)

Figure 1.4 Simplified figure of the function of a loudspeaker

1.4.2 Nonlinear distortion
Loudspeakers and other equipment that produce vibrations are amplitude dependent. This property has a strong binding to nonlinear behaviour in a system [8]. The loudspeaker nonlinearities produce different kinds of distortion such as harmonic distortion (HD), intermodulation distortion (IMD) and amplitude modulation distortion (AMD). The harmonic distortion states the generation of harmonics which are multiples of the fundamental frequency. A detail explanation of this phenomenon is found in section 1.4.4. The intermodulation distortion is a form of amplitude modulation of a signal which contains two or more frequencies. The amplitude peaks that are produced are normally not multiples of the fundamental frequency and are therefore not harmonics. In amplitude modulation there is a variation of the first tone, i.e. the carrier, in accordance with the second tone. The phase of the carrier is not affected by this variation [8].
1.4.3 Origin of the nonlinearities

1.4.3.1 Suspension
The suspension system in a loudspeaker is used to make the coil fall back to its original position after its movement. At low amplitudes there is a linear relationship between the back-falling force and the displacement widthways but for higher amplitudes this is not the case. These properties will cause nonlinear stiffness which generates harmonic distortion [8].

1.4.3.2 The force factor
The force factor describes the relation between the electricity in the coil and the force that attracts or repels the permanent magnet. The force factor depends on the position of the coil and the magnetic field generated by the permanent magnet. The asymmetry of the force factor and the voice coil displacement causes the nonlinearity which generates harmonic distortion, intermodulation distortion and amplitude distortion [8].

1.4.4 Harmonics
The frequency of a harmonic is a multiple of the fundamental frequency of the original wave. The first multiple, fundamental frequency times one is defined to be the first harmonic. Thus, the first harmonic is equivalent with the fundamental frequency. The second multiple of the fundamental frequency is the second harmonic and so forth. For a deeper understanding consider the following example [10].

Three sinusoidals which have the frequencies 25, 50 and 75 Hz are represented in figure 1.5. These three signals are summed in $y_{tot}$ into one signal.

$$y_1 = \sin(2\pi \cdot 25 t)$$

$$y_2 = \sin(2\pi \cdot 50 t)$$

$$y_3 = \sin(2\pi \cdot 75 t)$$

$$y_{tot} = y_1 + y_2 + y_3$$

Figure 1.5 Sinusoidals with different frequencies. From above: 25, 50 and 75 Hz respectively

The sinusoidal with the lowest frequency of all the sinusoidals, in this case $y_1$, is the first harmonic. $y_2$, which has twice the frequency of $y_1$, is called the second harmonic. $y_3$ is called the third harmonic because it has three times the frequency of $y_1$. The summation, $y_{tot}$, of these three sinusoidals is shown in figure 1.6 and the corresponding signal in frequency domain in figure 1.7 [10].
In frequency domain it is easy to see that the fundamental frequency is the first peak at 25 Hz and that the two harmonics appear at 50 and 75 Hz respectively.

1.4.4.1 Harmonic distortion

The harmonic distortion is a measure of the effect of harmonics in an audio signal. It describes the amplitude relation between the fundamental frequency and the harmonics and it is given as a percentage. The calculation of total harmonic distortion is given by:

$$ THD = \frac{\sqrt{H_2^2 + H_3^2 + \cdots + H_n^2}}{H_1} \times 100\% $$

where $H_1$ is the first harmonic, $H_2$ is the second and so forth.
1.5 Modelling physical systems
Physical systems can be well analysed and simulated if one has an accurate mathematical model describing them. Usually physical systems are complex to model in their entirety so the way to model them is by focusing on certain aspects of the system behaviour. System identification deals with modelling a dynamic system from its input and output measurements [11] [12].

To develop a working model, system identification has the following major procedures:

1.5.1 Data collection
Data collection from experiment: what type of input signal should be applied to the system to observe the main features of the system should be selected by the user.

1.5.2 Selecting model structure
This is the most important and difficult part of the system identification. If there is a prior model reflecting the physical characteristics of the system gray box modelling is used where one tries to find the unknown parameter in the model.

In this thesis one way to model the loudspeaker was to use an already defined state space model with unknown parameters [13]. This method was not selected because it requires knowledge of the mass of coil, inductance of coil, electrical resistance, input voltage and mechanical resistance for every speaker that is used in a mobile phone and it was not possible to get all these parameters from manufacturers since some of the parameters are not included in the specification of the speakers. Furthermore it would be impractical to request these parameters for a mobile platform maker from its customer for every type of speaker they are going to install on their mobile phones.

Black box modelling is used when there is no prior model describing the system, in which case standard linear or nonlinear models are used. The parameters may not have a direct physical interpretation as in gray box models. Example of linear models are FIR (Finite Impulse Response) and ARMAX (AutoRegressive Moving Average with external input). Volterra filters, Neural networks and sigmoid functions are some example of nonlinear models.

In this thesis nonlinear modelling of loudspeakers using Volterra filters and Hammerstein Models are considered which will be described in detail in Chapter 3 [13].

1.5.3 Parameter estimation
There are different adaptive algorithms to estimate model parameters. Common examples are LMS (Least Mean Squares) and RLS (Recursive Least Squares) [13].

1.5.4 Model validation
Once the model is identified and its parameters are determined the next step is to check whether or not the model is good enough to describe the intended properties of the system. This procedure is done by comparing the outputs of the real system and the model for a similar input. It is possible that the model will not pass the validation test in which case the above steps should be considered again. Probable reasons for the failure are that the collected data does not show the peculiar property of the system and as a result a good model will not be selected, or that the selected model does not describe the system [13].
1.6 Description of Acoustic Echo Control Implementation

1.6.1 Overview of Acoustic Echo Control Implementation
The Acoustic Echo Control implementation is comprised of three functional blocks, Echo Estimation (EE), Echo Subtraction (ES) and Residual Echo Control (REC) which are used to limit the effect of echo. A block diagram of an acoustic echo control can be seen in figure 1.8.

![Block diagram of acoustic echo control](image)

1.6.1.1 Echo Estimation (EE)
EE block estimates the linear echo by using a standard linear adaptive filter where the loudspeaker input signal is used as a regression variable.

1.6.1.2 Echo Subtraction (ES)
ES block subtracts linear echo estimated by EE block from the microphone input signal. The output of ES block is assumed to be a combination of residual echo, near-end speech and background noise.

1.6.1.3 Residual Echo Control (REC)
REC block estimates residual echo by using a loudspeaker signal or estimated echo as a regression variable and tries to suppress its effect computing a gain for each spectral frequency band based on estimated residual echo and true residual echo. The output of ES will be multiplied by this gain to produce a residual echo-free signal.

1.6.1.4 Residual Echo Estimation in REC
Residual echo in REC is estimated as the sum of nonlinear loudspeaker effects and digital clipping effects by the microphone. The choice of the residual echo estimation method depends on computational complexity, accuracy of the estimation and adaptability of the method with the real residual echo.
CHAPTER 2 MEASUREMENTS
2.1 Measurement procedure
To make sure that the harmonic distortion is caused by the loudspeaker of the phone a number of measurements where performed on three different mobile phones that were documented to have problems with nonlinear echo. The mobile phones are either a standard mobile phone or a cheap smartphone. It is therefore fair to assume that the components that are used in the mobile phones, e.g. the loudspeaker, are not of the highest quality.

The measurement setup was the following: A laptop was connected to the mobile phone through a USB cable. For a visual view of the measurement setup see figure 2.1. The specially developed software that was used during this measurement captured data from two points in the system. The first measurement point was the loudspeaker input and the second measure point was the microphone input. This is shown in figure 2.2. To analyze the distortion from different aspects, measurements with different inputs were performed. The inputs were generated from an internal network server modem by calling the modem with the mobile phone, using a special SIM card that was placed inside the mobile phone. The different inputs that were used were tones, frequency sweep and white noise. The choice of input was done through pressing the corresponding button on the phone. Each phone call lasted for about 30 seconds and it was during that time that the data was captured. The measurements were done in a small measurement room. The room was not extra isolated since it was assumed that a room without extra isolation would give a more realistic measurement environment.

Figure 2.1 An illustration of the measurement setup. The laptop is connected to the mobile phone by a USB cable. The phone is calling the server modem which is generating the input to the mobile phone

Figure 2.2 Capturing measurement data. The upper picture shows that the measurement data is captured before the loudspeaker and the lower picture shows that the microphone input was measured
2.1.1 Measurement type 1: Hand held mode
Hand held mode means that the mobile phone is used in the normal way, thus the mobile phone is held against the ear when a phonecall is performed. The setup for the hand held measurement is shown in figure 2.1.

2.1.2 Measurement type 2: Hands free mode
With the mobile phone’s hands free mode on, a phone call can be performed keeping the mobile phone at a distance from the mouth and ear. Thus, the cell phone could lie on the table during the phonecall. This usage allows several people from the same end to participate in a phonecall. The hands free mode sets higher requirements on the mobile phone since the sound from the loudspeaker and the capturing capability of the microphone needs to be higher compared to hand held mode. During hands free call the cell phone may use a different speaker than the hand held mode speaker. This special loudspeaker was placed on the back of the phones that were measured, see figure 2.3.

2.1.3 Measurement type 3: Measurement with an external microphone.
To make sure that it is the loudspeaker and not e.g. the microphone that is causing the problem, measurement type 1 and 2 were also performed using an external microphone. Thus, during the measurement the cell phone loudspeaker was still used but not its microphone. The external microphone was connected to the microphone port of the laptop through an audio capture device. The microphone was held close to the cell phone loudspeaker during the measurement. An illustration of this measurement setup can be seen in figure 2.4.

Figure 2.3 Hands free mode. On the backside of the mobile phone there is an extra speaker that may be used for hands free calls

Figure 2.4 External microphone. During measurement type 3 an external microphone was connected to the personal computer to capture the sound waves that were produced by the cell phone loudspeaker
2.2 Measurement analysis

2.2.1 Input signals
The most frequently used inputs during this artifact analysis were frequency sweep and white noise. The reason for this choice is that a frequency sweep plot may give a reasonably good view of harmonic distortion and that white noise has a similar behaviour to speech.

2.2.1.1 Frequency sweep
Frequency sweep is a signal that sweeps through frequencies in a certain broadband. The sweep starts from a low frequency and then the frequency is increased successively until it has gone through all the frequencies in the frequencyband. In Figure 2.5 a plot of a frequency sweep with decreasing amplitude is shown. This signal is the measured loudspeaker input.

![Figure 2.5 Frequency sweep with decreasing amplitude, time domain plot](image)

2.2.1.2 White noise
White noise is stationary stochastic process where all frequencies contain the same average power[1]. A plot of pulses of white cyclo stationary noise is shown in figure 2.6. It is a loudspeaker input signal.

![Figure 2.6 Pulses of white cyclo stationary noise, time domain](image)
2.2.2 Hand held mode

In the first graph that is shown in figure 2.7 the loudspeaker input is observed with a frequency sweep as an input. The signal has high amplitude. In figure 2.8 the microphone input is shown. If the two graphs are compared it is possible to distinguish two additional peaks that are shown in figure 2.8. The peaks are marked by red circles in the figure. Since the two additional peaks are multiples of the first peak, that could be seen to the left, they can be determined as harmonics.

![Figure 2.7 Loudspeaker input. Frequency sweep with a high amplitude, frequency domain](image1)

![Figure 2.8 Microphone input. Frequency sweep with a high amplitude, frequency domain](image2)

In figure 2.9 and 2.10 the loudspeaker input and the microphone input are shown but this time with lower amplitude. If the two graphs are compared with the upper ones it could be seen that the peaks seems to be gone. Thus, the conclusion must be that the harmonic distortion is amplitude dependent for the hand held mode.

![Figure 2.9 Loudspeaker input. Frequency sweep with a low amplitude, frequency domain.](image3)

![Figure 2.10 Microphone input. Frequency sweep with a low amplitude, frequency domain.](image4)

The input that is used for the third pair of graphs, 2.11 and 2.12, is white noise. Here is the harmonics not visible at all and this was not expected. This means that nonlinearities such as harmonic distortion are not visible for white noise input signal, which means that they are not present for speech signal either.

![Figure 2.11 Loudspeaker input. White noise, frequency domain.](image5)

![Figure 2.12 Microphone input. White noise, frequency domain.](image6)
2.2.3 Hands free mode

For these measurement settings and frequency sweep as an input the harmonic distortion is even more visible compared to the measurement with hand held mode, see figure 2.13 and 2.14. This may depend on the fact that the loudspeaker needs to generate a greater volume of the sound. This is because the communicator at the near-end must be able to understand the message from the far-end even though the mobile phone is placed on a distance from the near-end communicator.

In figure 2.15 and 2.16 the same input signal is illustrated as in figure 2.13 and 2.14 but with the difference that it has lower amplitude. Compared to the hand held mode where the harmonics were almost gone for the low amplitude signal the same relation is not found with the hands free mode. Here the harmonic distortion is almost the same for high and low amplitude inputs.

The result for the white noise input is the same for hands free mode as for hand held mode. This can be seen in figures 2.17 and 2.18.
2.2.4 Measurement with an external microphone.
To be sure that the harmonic distortion is not generated by the microphone on the mobile phone, measurements with an external microphone were performed. The result of this measurement is shown in figure 2.19 and 2.20. This measurement was done with hands free mode and the harmonics is still present in the microphone input signal. Thus, the conclusion is that the harmonics is not generated by the microphone in the mobile phone. Most likely the harmonics is generated by the loudspeaker.

![Figure 2.19 Measured with an external microphone. Loudspeaker input, frequency sweep, frequency domain](image1)

![Figure 2.20 Measured with an external microphone. Microphone input, frequency sweep, frequency domain](image2)

2.3 Measurement result
The purpose of the measurement was to distinguish the problem with nonlinear echo. The measurements were done with three different setups:

- Hand held mode
- Hands free mode
- External microphone

From the measurement result, where frequency sweep is used as an input, it is seen that it is likely that harmonic distortion is causing the nonlinear echo. The harmonics is above all present in hands free mode when the loudspeaker needs to generate a loud volume sound, but it is also present to some extent for hand held mode when the input signal has high amplitude. Thus, for hand held mode the harmonic distortion is amplitude dependent, however this is not the case for the hands free mode.
3.1 Static method

3.1.1 Nonlinear loudspeaker effects
The measurement described in chapter 2 clearly showed cheap loudspeakers used in mobile phones produce nonlinear effects where the major one is harmonic distortion. The static method estimates the harmonic distortion based on the loudspeaker input signal without considering the true residual echo as described in detail below.

3.1.2 Harmonic distortion
The power of harmonics produced by loudspeakers depends on fundamental frequency and input signal power. Estimation of harmonic distortion in this implementation assumes that the loudspeaker output at frequency $f_k$ contains the sum of the first six harmonic contributions from input frequencies below $f_k$ which can be described as follows mathematically:

$$\|X_{\text{acoustic}}(f_0)\|^2 \propto \|X_{\text{linear}}^\text{linear}(f_0)\|^2 + \|X_{\text{nonlinear}}(f_0)\|^2$$

(3.1)

where $\|X_{\text{acoustic}}(f_0)\|$ is loudspeaker output power at frequency $f_0$

Assuming the nonlinear power at $f_0$ is only due to harmonic overtones:

$$\|X_{\text{nonlinear}}(f_0)\|^2 = \sum_{k=1}^6 \|X_{\text{harmonic}}(f_0/(k+1))\|^2$$

(3.2)

The harmonic power is expressed in terms of linear power at fundamental frequency $f_0$ as follows

$$\|X_{\text{harmonic}}(f_0/(k+1))\|^2 = h_k g(f_0) \|X_{\text{linear}}(f_0)\|^2$$

(3.3)

where $k = 1,2,...$

Parameters used for the actual implementation of harmonic distortion estimation based on the above mathematical model are Harmonic Activation Level, Harmonic Gains, Fundamental Gain and Mapping table.

3.1.3 Harmonic Activation Level
Harmonic Activation Level is the minimum loudspeaker input signal power above which harmonics will be produced. Hence if at any instant the loudspeaker input signal power is greater than Harmonic Activation Level, harmonics will be produced by the loudspeaker.

3.1.4 Harmonic Gains
Harmonic Gains are defined as amplitude or power ratio of each of the six harmonics to the fundamental in dB scale.

$$G_k = 20 \log \left( \frac{A_k}{A_0} \right), \quad k = 1,2,...,6$$

(3.4)

where $A_k$ is amplitude of $k^{th}$ harmonic signal.

$A_0$ is amplitude of fundamental frequency

Harmonic Gain is used to calculate amplitude or power of harmonics produced for a given fundamental frequency.
3.1.5 Fundamental Gains
Fundamental Gains describe the relative level of overtones produced by fundamentals in the frequency bands. Several tests for a specific loudspeaker are made to set a reasonable static Fundamental Gain describing how the overtones are emphasized in the frequency bands.

3.1.6 Mapping table
The Mapping table is constructed based on the mathematical formula given in section 3.1.2 above to show which of the frequency bands below band k contributed harmonics that lie in band k.

3.1.7 Harmonic distortion estimation
Estimation of harmonic distortion is implemented using the above parameters. First a regression variable is selected, either loudspeaker input signal or estimated echo. Then the power of the regression variable is compared with Harmonic Activation Level. If it is less than the Harmonic Activation Level then harmonics are not expected to be produced. Hence estimation will not be done. If it is greater than the Harmonic Activation Level then harmonics are expected to be produced and estimation of harmonics will continue. For a given regression variable input amplitude (or power), the first six harmonics amplitude (or power), are calculated from Harmonic Gains. The location of the frequency bands to hold these overtones and which harmonics should be added in a particular frequency band is determined by the mapping table. Then multiply the result by Fundamental Gain of each frequency band. The resulting estimated harmonic distortion is scaled by weights of linear echo estimator to track echo path effects of the harmonic distortion. Then the over all gain will be calculated based on the estimate of harmonic distortion. This gain is used to multiply the signal coming out of the linear AEC to reduce the non linear echo.

3.1.8 Mapping table construction
In a banded frequency spectrum, identifying frequency bands those contributed overtones to the current band cannot be done by just dividing or multiplying the fundamental frequency as is the case with unbanded frequency bins. Instead a mapping table is constructed beforehand.

The current implementation divides the frequency spectrum into frequency bands; each with a bandwidth of 250Hz. If we consider a narrow band speech, the frequency extends to 4000Hz in which case the frequency spectrum will have 16 bands (4000Hz/250Hz). Table 3.1 shows 16 bands with their range of frequencies

<table>
<thead>
<tr>
<th>Band Number</th>
<th>Minimum frequency</th>
<th>Maximum frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>249</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>499</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td>749</td>
</tr>
<tr>
<td>4</td>
<td>750</td>
<td>999</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>1249</td>
</tr>
<tr>
<td>6</td>
<td>1250</td>
<td>1499</td>
</tr>
<tr>
<td>7</td>
<td>1500</td>
<td>1749</td>
</tr>
<tr>
<td>8</td>
<td>1750</td>
<td>1999</td>
</tr>
<tr>
<td>9</td>
<td>2000</td>
<td>2249</td>
</tr>
<tr>
<td>10</td>
<td>2250</td>
<td>2499</td>
</tr>
<tr>
<td>11</td>
<td>2500</td>
<td>2749</td>
</tr>
<tr>
<td>12</td>
<td>2750</td>
<td>2999</td>
</tr>
<tr>
<td>13</td>
<td>3000</td>
<td>3249</td>
</tr>
<tr>
<td>14</td>
<td>3250</td>
<td>3499</td>
</tr>
<tr>
<td>15</td>
<td>3500</td>
<td>3749</td>
</tr>
<tr>
<td>16</td>
<td>3750</td>
<td>3999</td>
</tr>
</tbody>
</table>

Table 3.1 Frequency bands for narrow band speech with their frequency range
To illustrate how the mapping table is constructed, calculation of the frequency bands which contribute overtones for the 6th band is described below.

Calculate the harmonics for the 6th band:

1. Could the fundamental frequencies lie in band 6 for the 1st overtone?
   → 1499 x 2 = 2998, 2998 > 1499 → No, the overtone will lie outside band 6 if that were the case.

2. Could the fundamental frequencies lie in band 6 for the 1st overtone?
   → 1250 x 2 = 2500, 2500 > 1499 → No, the overtone will lie outside band 6 if that were the case.

3. Could the fundamental frequencies lie in band 5 for the 1st overtone?
   → 1000 x 2 = 2000, 2000 > 1499 → No, the overtone will lie outside band 6 if that were the case.

4. Could the fundamental frequencies lie in band 4 for the 1st overtone?
   → 750 x 2 = 1500, 1500 > 1499 → No, the overtone will lie outside band 6 if that were the case.

5. Could the fundamental frequencies lie in band 3 for the 1st overtone?
   → 500 x 2 = 1000, 1000 < 1449 → Yes, the overtone will lie inside band 6.

6. Could the fundamental frequencies lie in band 3 for the 2nd overtone?
   → 500 x 3 = 1500, 1500 > 1449 → No, the overtone will lie outside band 6 if that were the case.

7. Could the fundamental frequencies lie in band 2 for the 2nd overtone?
   → 250 x 3 = 750, 750 < 1449 → Yes, the overtone will lie inside band 6.

8. Could the fundamental frequencies lie in band 2 for the 3rd overtone?
   → 250 x 4 = 1000, 1000 < 1449 → Yes, the overtone will lie inside band 6.

9. Could the fundamental frequencies lie in band 2 for the 4th overtone?
   → 250 x 5 = 1250, 1250 < 1449 → Yes, the overtone will lie inside band 6.

10. Could the fundamental frequencies lie in band 2 for the 5th overtone?
    → 250 x 6 = 1500, 1500 > 1449 → No, the overtone will lie outside band 6 if that were the case.

11. Could the fundamental frequencies lie in band 1 for the 5th overtone?
    → 1 x 6 = 6, 6 < 1449 → Yes, the overtone will lie inside band 6.

12. Could the fundamental frequencies lie in band 1 for the 6th overtone?
    → 1 x 7 = 7, 7 < 1449 → Yes, the overtone will lie inside band 6.

The result of the procedure for the 6th band will be as in Table 3.2.

<table>
<thead>
<tr>
<th>Overtone</th>
<th>1st overt.</th>
<th>2nd overt.</th>
<th>3rd overt.</th>
<th>4th overt.</th>
<th>5th overt.</th>
<th>6th overt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple</td>
<td>2x</td>
<td>3x</td>
<td>4x</td>
<td>5x</td>
<td>6x</td>
<td>7x</td>
</tr>
<tr>
<td>freq. band</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.2 The result of calculating harmonics for the 6th band
A full list of the overtones in each frequency band is shown in Table 3.3.

<table>
<thead>
<tr>
<th>Band</th>
<th>Min freq</th>
<th>Max freq</th>
<th>h1</th>
<th>h2</th>
<th>h3</th>
<th>h4</th>
<th>h5</th>
<th>h6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>249</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>499</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td>749</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>750</td>
<td>999</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>1249</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1250</td>
<td>1499</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1500</td>
<td>1749</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>1750</td>
<td>1999</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>2000</td>
<td>2249</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>2250</td>
<td>2499</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>2500</td>
<td>2749</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>2750</td>
<td>2999</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>3000</td>
<td>3249</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>3250</td>
<td>3499</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>3500</td>
<td>3749</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>3750</td>
<td>3999</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3.3 Complete mapping table

3.1.9 Summary
The static method has low computational complexity and good performance in reducing non linear echo if reasonably accurate parameters are set for the particular mobile phone. The drawback of static method is measurements should be done for each cell phone to come up with parameters specific to the device. In addition to that, it computes the worst case scenario where it assumes if the harmonics are produced there will be maximum harmonic distortion all the time regardless of the amplitude or power of the loudspeaker input signal. This causes the overall gain calculation to be aggressive. Consequently it tries to remove more echo than what actually exists and this affects the near end speech.
3.2 Volterra filters

3.2.1 Introduction
The Volterra series expansion was developed by the Italian mathematician Vito Volterra (1860-1940) in the 1880s. It is basically a generalization of Taylor series to accommodate memory. The idea of using Volterra series for filtering applications was introduced by Norbert Wiener (1894-1964) in the 1950s. Application of Volterra series for non linear system modeling is achieved by truncating the infinite series by considering only finite order and memory. Volterra filters are used in noise and echo cancellation, channel equalization, signal detection and estimation, spatial and temporal discrimination [15][16].

3.2.2 Mathematical description
The Volterra series is defined as[6]:

\[
\hat{y}(n) = w_0 + \sum_{m_1=0}^{\infty} w_1(m_1)x(n - m_1) + \sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} w_2(m_1, m_2)x(n - m_1)x(n - m_2) + ... \\
+ \sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} \sum_{m_N=0}^{\infty} w_N(m_1, m_2, ..., m_N)x(n - m_1)x(n - m_2) ... x(n - m_N) + ...
\]

(3.5)

where \( w_N(m_1, m_2, ..., m_N) \) is Nth order kernel
\( x(n) \) is the input and \( \hat{y}(n) \) is the output

To model real systems the series is truncated in two ways: limiting the order of the kernel (N) and limiting the memory of the system to M.

The second order Volterra filter with memory M is shown below:

\[
\hat{y}(n) = w_0 + \sum_{m_1=0}^{M-1} w_1(m_1)x(n - m_1) + \sum_{m_1=0}^{M-1} \sum_{m_2=0}^{M-1} w_2(m_1, m_2)x(n - m_1)x(n - m_2)
\]

(3.6)

An important property of the Volterra filter which minimizes the computational complexity significantly is the symmetry of its kernels. That is:

\[
w_2(m_1, m_2) = w_2(m_2, m_1)
\]

(3.7)

Moreover the filter coefficients are linearly dependent which makes it suitable it for adaptive filter application, and hence on can use standard adaptive algorithms.

3.2.3 Adaptation of Volterra filters
Volterra filter coefficients can be updated recursively using any adaptive algorithm since the error signal (difference between estimated and actual) can be written as a linear combination of the input signal to each filter coefficient [8][10]. Due it its ease of computation and fairly good convergence rate the LMS algorithm is widely used. Consequently its description for second order Volterra filter is discussed below.
First order Volterra kernel vectors:
\[
x_1(k) = [x(k), x(k-1), ..., x(k-M+1)]^T
\]
\[
w_1 = [w_1(0), w_1(1), ..., w_1(M-1)]
\]

Second order Volterra kernel vectors:
\[
x_2(k) = [x^2(k), x(k)x(k-1), ..., x(k)x(k-M+1),
\]
\[
x(k-1)x(k-1), ..., x(k-M+1)x(k-M+1)]^T
\]
\[
w_2 = [w_2(0,0), w_2(0,1), ..., w_2(0,M-1), w_2(1,1), ..., w_2(M-1,M-1)]
\]

LMS adaptation is formulated as:
\[
y(k) = w_0 + w_1(k)x_1(k) + w_2(k)x_2(k)
\]
\[
e(k) = y(k) - \hat{y}(k)
\]
\[
w_0(k+1) = w_0(k) + \mu_0 e(k)
\]
\[
w_1(k+1) = w_1(k) + \mu_1 e(k)x_1(k)
\]
\[
w_2(k+1) = w_2(k) + \mu_2 e(k)x_2(k)
\]  

where \( \mu_0, \mu_1, \mu_2 \) are step sizes for Volterra kernels and \( e(k) \) is the error.

The convergence speed of the LMS algorithm depends on the eigen vectors of the autocorrelation matrix of the input signal. The step size controls the speed of convergence and ensures stability of the filter.

3.2.4 Application of Volterra filter for Acoustic Echo Cancellation
Adaptive Volterra filters are being applied for Nonlinear Acoustic Echo Cancellation (NAEC) purposes in hands-free communications due to the increased computational power of devices and low cost amplifiers and loudspeakers that introduce significant nonlinear echo. Efficiency of the overall AEC, both linear and nonlinear, depends on the ratio of linear to nonlinear echo signal power (LNLR). For high LNLR, linear echo dominates the nonlinear one in which case linear AEC is more efficient. Adding a Volterra filter in such scenarios will merely increase the computational complexity and possibly introduce gradient noise caused by the adaptation of higher order kernels. [17]

LNLR is unknown at priori and will be time-varying for non-stationary signals like speech. Thus deciding either to use a simple linear filter or Volterra filter is difficult. Two types of filter combination methods are suggested in [17] to alleviate the problem. The first combines linear and Volterra filters, while the second method relies on the combination of kernels. Both methods perform at least as well as the best contributing filter.

3.2.5 Combination of Filters Scheme (CFS)
A convex combination of an adaptive linear filter and adaptive Volterra filter including linear and quadratic kernel is considered as a possible solution for NAEC. Each filter independently updates its parameters using a separate adaptation step size and error signal power and then their output will be weighted by an adaptive parameter, \( \lambda(n) \), determining which output should be emphasized in the overall estimated echo signal. Pictorial description of the combination is shown in figure 3.1.
3.2.6 Description of the combination

The outputs of the linear and Volterra filters can be computed as:

\[
\hat{y}_l(n) = h^T(n)x(n) \quad (3.13)
\]

\[
\hat{y}_v(n) = w_1^T(n)x(n) + x^T(n)w_2(n)x(n) \quad (3.14)
\]

And the output of the combined filter is defined as:

\[
\hat{y}(n) = \lambda(n)\hat{y}_l(n) + [1 - \lambda(n)]\hat{y}_v(n) \quad (3.15)
\]

where \(\lambda(n)\) is an adaptive weighting parameter that controls the combination.

Better performance of the combination is achieved if each filter updates its parameters by its own rule[18]. Using standard gradient descent method for adaptation:

\[
h(n + 1) = h(n) + \mu_l e_l(n)x(n) \quad (3.16)
\]

\[
w_1(n + 1) = w_1(n) + \mu_{vl} e_v(n)x(n) \quad (3.17)
\]

\[
w_2(n + 1) = w_2(n) + \mu_{vq} e_v(n)x(n)x^T(n) \quad (3.18)
\]

where \(\mu_l\), \(\mu_{vl}\) and \(\mu_{vq}\) are step sizes and

\[
e_l(n) = y(n) - \hat{y}_l(n) \quad (3.19)
\]

\[
e_v(n) = y(n) - \hat{y}_v(n) \quad (3.20)
\]

The weighting parameter \(\lambda(n)\) can be updated in the same way with the intent of minimizing the error of the combined filter, i.e.

\[
e(n) = y(n) - \hat{y}(n) \quad (3.21)
\]
where $\hat{y}(n)$ is the output of the combined filter calculated in equation 3.2.

Instead of adapting $\lambda(n)$ directly, [6] proposes adapting another parameter $\alpha(n)$ which defines $\lambda(n)$ as a sigmoidal activation function for the purpose of keeping $\lambda(n) \in (0,1)$ and to reduce gradient noise near $\lambda(n) = 1$ or $\lambda(n) = 0$. [7]

The weighting parameter is expressed as:

$$\lambda(n) = \text{sgm}[\alpha(n)] = \left[1 + e^{-\alpha(n)}\right]^{-1}$$  \hspace{1cm} (3.22)

Using the recently presented updating algorithm for $\alpha(n)$ and normalizing it for a better performance in case of time varying SNR:

$$a(n + 1) = a(n) + \frac{\mu}{p(n)} \lambda(n) [1 - \lambda(n)] e(n) [e_v(n) - e_l(n)]$$  \hspace{1cm} (3.23)

where $p(n)$ is an estimate of the power of $[e_v(n) - e_l(n)]$

It is defined as:

$$p(n) = \beta p(n - 1) + (1 - \beta) [e_v(n) - e_l(n)]^2$$  \hspace{1cm} (3.24)

Therefore CFS works in such a way that when LNLR is low, there is high nonlinear echo, the Volterra filter represents an effective model of the channel, and minimization of the overall error yields $\alpha(n) \rightarrow 0$, so that $\hat{y}(n) \approx \hat{y}_v(n)$. The reverse occurs when LNLR is high with $\lambda(n) \rightarrow 1$ and $\hat{y}(n) \approx \hat{y}_l(n)$, so that the combination is equivalent to the linear filter avoid the gradient noise introduced by high order kernel of Volterra filter.

### 3.2.7 Combination of Kernels Scheme (CKS)

This method tries to solve the problem by using only one Volterra filter and replacing one of its kernels by a convex combination of kernels, considering second order Volterra filter and replacing its quadratic kernel by two other kernels as shown in figure 3.2.

![Figure 3.2 Simplified combination of kernels scheme](image)

The output of the combination can be represented as:
\[ \hat{y}(n) = \hat{y}_{lk}(n) + \eta(n)\hat{y}_{q1}(n) + [1 - \eta(n)]\hat{y}_{q2}(n) \]  

(3.25)

where \( \hat{y}_{lk}(n) \) is the output of the linear kernel
\( \hat{y}_{q1}(n), \hat{y}_{q2}(n) \) are the outputs of the kernels in the combination
\( \eta(n) \) is a mixing parameter in the range of (0,1)

For a better performance each kernel should update using its own adaptation rule and error signal, which allows the two replaced kernels to be different in any way, either in size or in memory. Letting the linear kernel adapt according to the overall error and each kernel according to its error signal, one has:

\[ e(n) = y(n) - \hat{y}(n) \]  

(3.26)

\[ e_{q1}(n) = y(n) - [\hat{y}_{lk}(n) + \hat{y}_{q1}(n)] \]  

(3.27)

\[ e_{q2}(n) = y(n) - [\hat{y}_{lk}(n) + \hat{y}_{q2}(n)] \]

Adaptation of the mixing parameter \( \eta(n) \) is done by defining a sigmoid function as explained in CFS method above. Using Steepest Descent algorithm for its adaptation to minimize the overall squared error:

\[ b(n + 1) = b(n) + \frac{\mu_b}{p(n)} \eta(n)[1 - \eta(n)]e(n)[\hat{y}_{q1}(n) - \hat{y}_{q2}(n)] \]  

(3.28)

where \( \eta(n) = \text{sgm}[b(n)] \)

A special case of the combination is when all taps of the first quadratic kernel are zero.

\[ \hat{y}(n) = \hat{y}_{lk}(n) + [1 - \eta(n)]\hat{y}_{q2}(n) \]  

(3.29)

This shows that the mixing parameter decides the role of quadratic kernel on the overall output of the kernel combination. Equation (13) can be written as:

\[ \hat{y}(n) = \eta(n)\hat{y}_{lk}(n) + [1 - \eta(n)][\hat{y}_{lk}(n) + \hat{y}_{q2}(n)] \]  

(3.30)

Comparing equation (3.30) with equation (3.15) of CFS method, this method is only slightly more complex than the standard Volterra filter with addition of combining factor adaptation. It is better than the CFS method since it has only one linear filter while CFS updates two linear filters, one independent linear filter and the linear kernel of the volterra filter.

Experiments with speech and colored input noise in [17] have shown that two methods, CFS and CKS, give a better result than only linear filter with up to 10dB Echo Return Loss Enhancement (ERLE) gain for high nonlinearity. Adaptation of the mixing parameter is very crucial since for a varying LNLR the ERLE improvement and complexity of algorithm are adjusted to be optimum by deciding whether kernels or filter should be given more emphasis for the overall output. Moreover, CKS is better than CFS for all LNLR scenarios providing less complexity and ERLE improvement.

### 3.2.8 Limitation of volterra filter

Although Volterra filters can model a large class of nonlinear systems and its inherent property made it applicable for adaptive filter applications, it has some limitations which discourage its application for nonlinear systems:

1. High computational complexity for higher order nonlinear systems
2. Does not efficiently model strongly non linear systems
3. Slow convergence rate for non Gaussian inputs
4. Over-parameterization where large numbers of coefficient adaptations are needed even for low order representations.
3.3 Hammerstein model

3.3.1 Introduction
The (nonlinear) Hammerstein system is in resemblance with Volterra and Weiner (nonlinear) an advanced form of a nonlinear system. It consists of a series connection between a nonlinear and a linear block. The nonlinear block is static gain while the linear block is a dynamic system [19]. The blocks work together to build a model of the nonlinearities that e.g. is caused by the loudspeaker. There are two approaches of the Hammerstein model. The first approach assumes that the nonlinearities are of a polynomial kind so that the modelling could be divided into a linear and a nonlinear part. The second method does not assume that the model is polynomial all the time which makes it more difficult to create since it is necessary to make the estimation of a related function instead of the nonlinear elements. What these two methods have in common is that they create their model of a truncated series. The Hammerstein model in some cases has the capacity of creating dilated models of nonlinearities [21]. In the field of system identification the Hammerstein model has gained importance in areas like digital communication and biomedical engineering. In biomedical engineering it has been used to modelling muscle contractions [19].

System architecture

![Diagram of Nonlinear AEC – Hammerstein model (adaptive)](image)

Figure 3.3 shows a proposition of the nonlinear part of an AEC Hammerstein model. The signal from the far-end is going through the nonlinear block first and then through the linear block. The blocks are modelling the acoustic echo and the noise that is captured by the microphone. The linear block will model the linear acoustic echo and the nonlinear block will model the nonlinear acoustic echo. The received signal will then be subtracted from the modelled noise and echo. After the subtraction the modelling blocks will be updated to improve future estimation [20].
3.3.2 AEC nonlinear Hammerstein design together with a shortening filter
Like most nonlinear algorithms it brings high complexity and slow convergence. To overcome these issues the writers of article [20] propose an additional shortening filter. The idea is that the shortening filter will cut down the amount of taps needed in the linear block. Another benefit by adding the shortening filter is that the complexity of the updating algorithm to the AEC is reduced and that the grade of convergence will be improved [20]. The structure of the Hammerstein model together with a shortening filter can be seen in figure 3.4.

![Diagram of AEC nonlinear Hammerstein design with shortening filter](image)

**Figure 3.4** AEC nonlinear Hammerstein design together with a shortening filter

### 3.3.2.1 The shortening filter
To avoid that two samples belonging to two different symbols will interfere with each other one has to make sure that there is enough distance between them. In a discrete multitone transceiver (DMT) this is done by a so-called cyclic prefix (CP) which is an intelligent guard time sequence. The CP adopts the distance between the samples from different symbols by calculating the length of the impulse response of an effective physical channel. Using a CP that is too long will decrease the capacity of the transceiver. To optimise the effective channel impulse response a shortening filter can be used to keep the length of CP short [21]. The shortening filter will contribute to minimise the length of the impulse response of the effective channel.

![Diagram of AEC nonlinear Hammerstein design with shortening filter](image)

**Figure 3.5** A illustration of the architecture of a shortening filter

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC</td>
<td>Encoder</td>
</tr>
<tr>
<td>IFFT</td>
<td>Inverted Fast Fourier Transform</td>
</tr>
<tr>
<td>+ CP</td>
<td>Add the cyclic prefix</td>
</tr>
<tr>
<td>D/A</td>
<td>Digital to analog converter</td>
</tr>
<tr>
<td>PC</td>
<td>Physical channel</td>
</tr>
<tr>
<td>A/D</td>
<td>Analog to digital converter</td>
</tr>
<tr>
<td>SF</td>
<td>Shortening filter</td>
</tr>
<tr>
<td>- CP</td>
<td>Remove the cyclic prefix</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>DC</td>
<td>Decoder</td>
</tr>
</tbody>
</table>

35
In figure 3.5 there is a drawing of the structure of a shortening filter. Here follows an explanation of the flow in the figure. The bits that are coming from the far-end are transformed into carrier frequencies by the encoder. The carrier frequencies are converted to time-domain through the inverse Fast Fourier Transform. The cyclic prefix of a certain length is inserted between the symbols to make sure that samples of different symbols will not interfere. The signal is transformed from digital to analog through the D/A converter and then transferred through the physical channel where it is converted back to digital again by the A/D converter. The signal now passes through the shortening filter where the amount of taps is reduced. The next step is to remove the cyclic prefix and transform the signal into frequency domain by the Fast Fourier Transform. In the end of the process each frequency block is reconstructed into streaming bits in the decoder [21].

3.3.3 Mathematical representation

3.3.3.1 Initialization
To create the Hammerstein model one starts with the initialization of some variables needed in the system. The signal from the far-end is denoted by \( p(n) \) and is branching in two directions; one where the signal is amplified and sent to the loudspeaker and the other one going to the AEC modelling. The filters that compose the AEC modelling are the nonlinear filter \( \mathcal{G}_{123a} / \mathcal{G}_{753} / \mathcal{G}_{754} / \mathcal{G}_{1e2} / \mathcal{G}_{245} / \mathcal{G}_{123b} \) and the linear filter \( \mathcal{G}_{3} / \mathcal{G}_{744} / \mathcal{G}_{123a} / \mathcal{G}_{74a} / \mathcal{G}_{123b} \). The two filters are initialized as following:

\[
\begin{align*}
\theta &= [\theta_1, \theta_2, ..., \theta_K]^T \\
\mathcal{S}(n) &= [b(x(n)), b(x(n)), ..., b(n - L_h + 1)]^T
\end{align*}
\]

where \( L_h \) is the length of the linear filter. The nonlinear filter is composed by a linear combination of the nonlinear coefficient \( \theta \) and the basis function \( b(n) \). They are denoted by:

\[
\begin{align*}
\theta &= [\theta_1, \theta_2, ..., \theta_K]^T \\
b(n) &= [b_1(x(n)), b_2(x(n)), ..., b_K(x(n))]^T
\end{align*}
\]

And in matrix form

\( \mathcal{S}(n) = [b(x(n)), b(x(n)), ..., b(n - L_h + 1)] \)

where \( L_h \) is the length of the non linear filter.

As could be seen in the figure above a signal is sent from the loudspeaker through the acoustic room and finally captured by the microphone. The signal from the microphone, which contains echo, noise and near-end speech, is denoted by \( y(n) \) in the figure and is initialized as follows:

\( y(n) = [y(n), ..., y(n - L_w + 1)]^T \)

Note that \( L_w \) is the length of the shortening filter.

3.3.3.2 The reference signals
When the signal is sent through the first filter block from the far-field the following result is obtained:

\( c(n) = \theta^T(n)\mathcal{S}(n) \)  \hspace{1cm} (3.31)

The next path the signal passes through is the linear filter. The reference signal that is given after the filter is:

\( \hat{d}(n) = h^T(n)c(n) = \theta^T(n)\mathcal{S}(n)h(n) \)  \hspace{1cm} (3.32)
The signal that is captured by the microphone passes through the shortening filter and generates the following equation:

$$d(n) = z^T(n)y(n)$$  \hspace{1cm} (3.33)

### 3.3.3.3 The error signal

Now one has reached the point where the signal from the loudspeaker through the shortening filter and the signal that has passed through the AEC path unite. The signal that has passed the echo path is now subtracted by the model which is illustrated in this equation:

$$e(n) = d(n) - \tilde{d}(n) = z^T(n)y(n) - \theta^T(n)S(n)h(n)$$ \hspace{1cm} (3.34)

### 3.3.3.4 Adaptation

There are several adaptive algorithms that could be used to update the Hammerstein model. In the following text two of them are explained:

LMS – adaptation

$$z(n + 1) = z(n) - \mu_z e(n)y(n)$$ \hspace{1cm} (3.35)

$$z(n + 1)$$ updates the shortening filter by a subtraction of the product including the error signal and the signal captured by the microphone. $\mu_z$ denotes the step size of the adaptation of the shortening filter.

$$h(n + 1) = h(n) + \mu_h e(n)S^T(n)\theta(n)$$ \hspace{1cm} (3.36)

$h(n + 1)$ updates the linear filter by adding the product of the error signal, the basis matrix and the nonlinear coefficients. $\mu_h$ denotes the step size of the adaptation of the linear filter.

$$\theta(n + 1) = \theta(n) - \mu_\theta e(n)S(n)h(n)$$ \hspace{1cm} (3.37)

$\theta(n + 1)$ updates the nonlinear filter by subtracting with the product including the error signal, the basis matrix and the value of the linear filter. $\mu_\theta$ denotes the step size of the adaptation of the nonlinear filter.

### 3.3.3.5 NLMS – adaptation

Since the stability of the LMS adaptation cannot be ensured, the normalized LMS (NLMS) is introduced. The NLMS optimizes the value of the step size so that the adaptation algorithm both will be stable and ensure fast convergence.

### 3.3.3.6 Update for the shortening filter

$$z(n + 1) = z(n) - \frac{\mu_z e(n)y(n)}{||y(n)||_2^2}$$ \hspace{1cm} (3.38)

To improve the stability of the adaptation of the shortening filter the step size $\mu_z$ is divided by the microphone input $y(n)$.

Update for the linear filter

$$h(n + 1) = h(n) + \frac{\mu_h e(n)S^T(n)\theta(n)}{||S^T(n)\theta(n)||_2^2}$$ \hspace{1cm} (3.39)

To improve the stability of the adaptation of the linear filter the step size $\mu_z$ is divided by the product of the bias matrix $S^T(n)$ and the nonlinear coefficient $\theta(n)$.

Update for the nonlinear filter

37
\[ \theta(n + 1) = \theta(n) - \frac{\mu_\theta(n)e(n)S(n)h(n)}{\|S^T(n)h(n)\|_2^2} \] (3.40)

To improve the stability of the adaptation of the nonlinear filter the step size \( \mu_\theta \) is divided by the product of the bias matrix \( S^T(n) \) and the linear filter \( h(n) \) [20].

### 3.3.4 Summary

The Hammerstein model may be a good choice in the fields of digital communication and biomedical engineering. In good situations it may create a very accurate model of e.g. harmonic distortion. In resemblance with other nonlinear adaptive algorithms like the Volterra filter, it is computational complex and has slow convergence. According to article [20] these drawbacks could be overcome by adding a shortening filter to the Hammerstein filter architecture. It is shown in [20] that this new structure with the shortening filter uses less computational power and that the convergence speed is increased. One drawback to the addition of this shortening filter to the structure could be that the wanted signal becomes modulated in a way that could affect the sound. Thus, there could be a change in the voice characteristics. The writers of article [20] recommend using an equalizer to compensate for this problem.

Double talk, that is, when two persons that communicate through the phone talk at the same time, is an issue for most adaptive algorithms that deals with AEC. Unfortunately the Hammerstein model is not an exception. When double talk occurs, the algorithms have difficulties to decide what is echo and what is near-end speech. This will result either in that the near-end speech will be destroyed or that enjoying echo will be let through to the other end. One solution to this problem can be to make the algorithm adapt slower when there is double talk or even stop adapting during this time. This solution requires that the algorithm has access to a double talk detector [20].
3.4 Harmonic distortion residual echo suppressor (HDRES)

3.4.1 Introduction
The method of “Nonlinear Residual Acoustic Echo Suppression for High levels of Harmonic Distortion” [22] have been developed by Diego A. Bendersky from the University of Buenos Aires and Jack W. Stokes, Henrique S. Malvar from Microsoft Research.

Although there are many methods that can reduce the influence of nonlinear echoes and still keep a good quality of the near end speech, not so many of them are suitable when it comes to handheld devices. Volterra filters and neural networks for example involve complex calculation and use many variables which results in slow convergence and higher computational complexity.

A better way to speed up the convergence and minimize the computational could be to use a residual echo suppressor, also called RES. The RES works together with the linear AEC and contributes to the cancelation of the harmonic distortion (HD). Thus, the full name of this algorithm becomes Harmonic Distortion Residual Echo Suppression (HDRES). Figure 3.6 illustrates in a simplified way how the HDRES and the linear AEC filter are working together to cancel the echo. The trick that the HDRES uses is that it lets the linear AEC cancel as much of the echo as possible with the magnitude and the phase, and then the HDRES will suppress the nonlinear echo further. The HDRES algorithm that works in the frequency domain will not use so much computational capacity since it only calculates a single transform from the speaker signal at the time [22].

![Figure 3.6 A simplified illustration of how the HDRES and the linear AEC works together](image)

3.4.2 Residual echo suppression (RES)
The limitation of the linear acoustic echo canceller to subtract nonlinear echo from the echo path have created a need of an external box that could deal with this issue. The residual echo suppressor is an additional tool to suppress the nonlinear echo while still keeping the speech signal intact [24]. One condition for the RES to work is that the real signal that has its origin from the loudspeaker through the echo path and into the microphone needs to be separated from the estimated signal from the adaptive filter. To be able to do this the signal is divided into blocks in suitable sizes. The block is converted into the frequency domain through a filter bank or a discrete Fourier transform (DFT). The RES relies on a model of the nonlinear echo created in frequency domain.

3.4.3 System architecture
This acoustic echo cancelation system consists of two different filters. One of the filters is intended to cancel the linear echo and the other one will suppress the nonlinear echo. The first filter is a linear AEC adaptive algorithm that is working in the frequency domain. The second filter is the HDRES filter and it is also working in the frequency domain. Those two filters are parallel coupled with each other, this is illustrated in figure 3.7. The transformation from the time domain to the frequency domain is done by a modulated complex lapped transform (MCLT) [23].
3.4.4 Modulated Complex Lapped Transform (MCLT)
Since the HDRES algorithm is doing its arithmetic operations in the frequency domain the loudspeaker signal, that is in time domain, needs to be converted. There are many ways to convert a time domain signal into frequency domain. One possible method is to use discrete Fourier transform DFT and another method is Modulated Complex Lapped Transform, MCLT [3]. The MCLT is a complex version of the modulated lapped transform, MLT. The MLT is often used together with a system which contains noise reduction or acoustic echo cancelation, since it has perfect reconstruction. The perfect reconstruction minimizes the disturbances that other time-frequency converters are struggling with [24]. The MLT is developed from a cosine modulated filter bank which is modulating by cosines instead of exponentials. This means that it will generate two frequency shifts instead of one since the cosine is a summation of two exponentials [2]. The MLT is working with real numbers and therefore it has a limitation when it comes to working the system which contains complex information like the phase. Thus, the MLT is not suitable for systems that deal with noise subtraction and acoustic echo cancelling. To be able to use the MLT in those areas Henric Malvar on Microsoft research has developed an improved version of the MLT that also can work with complex numbers. Because of this extension the MCLT could be a good alternative to the standard DFT filter banks in areas like noise suppression and AEC. In the article [24] it is shown that the MCLT produce fewer disturbances during the convergence time to frequency domain or vice versa than the DFT filter bank [24].

![Diagram of HDRES model with AEC](image)

Figure 3.7 Illustrates the architecture of the HDRES model together with a linear AEC. A explanation of the notation is shown below

3.4.5 Mathematical representation

3.4.5.1 The linear filter
The linear filter estimates the loudspeaker signal that is travelling through the echo path by calculating a summation of the product of the complex weight $W_L$ and the loudspeaker signal $X$ [22]. Thus, the estimated echo $\hat{Y}$ is represented by:

$$\hat{Y}(n,m) = \sum_{t=0}^{T-1} W_L(t,m) X(n-t,m)$$  \hspace{1cm} (3.41)
In this function the number of taps, \([t=\text{?}]\), frame index and the frequency band are represented by \(T\), \(t\), \(n\), \(m\) respectively. The echo subtraction can be represented by the equation:

\[
E(n, m) = |Y(n, m)| - |\hat{Y}(n, m)|
\]  
(3.42)

Here the real echo is subtracted by the estimated echo. After the subtraction the linear AEC filter is updated so that the estimation can be improved.

### 3.4.5.2 The HDRES filter

The HDRES filter is suppressing the nonlinear echo by the multiplication of a constant, which has a value between 0 and 1, and the output from the linear AEC. The constant which is multiplied is called the gain \(G\) and is calculated by taking the maximum value between the noise floor \(N\) and the input \(E\) (output to the linear AEC) subtracted by the residual echo \(Y_r\) multiplied with a constant factor \(\beta\), which determines the aggressiveness of the equation. The maximum value is subtracted by the input \(E\).

\[
G(n, m) = \max(E(n,m) - \beta Y_r(n,m), N(n,m)) \overline{E(n,m)}
\]  
(3.43)

The additional bar on top of the letters \(E\), \(Y_r\) and \(N\) means that their magnitude is smoothed. This is done through the following calculation.

\[
\bar{E}(n, m) = (1 - \alpha)\overline{E}(n - 1, m) + \alpha|E(n, m)|
\]  
(3.44)

\[
\bar{Y}_r(n, m) = (1 - \alpha)\overline{Y}_r(n - 1, m) + \alpha|\overline{Y}_r(n, m)|
\]  
(3.45)

\[
\bar{N}(n, m) = (1 - \alpha)\overline{N}(n - 1, m) + \alpha|\overline{N}(n, m)|
\]  
(3.46)

\(\alpha\) makes it possible to decide how much of the functions that should be smoothed. \(\overline{Y}_r(n, m)\) denotes the estimated residual echo and is used to model the residual echo that is passed through the linear AEC. The model consists of a mapping of the spectral power, parameters of the HDRES model and the loudspeaker input in frequency domain.

\[
\overline{Y}_r(n, m) = \sum_{i=1}^M \sum_{j=1}^H \sum_{k=-K}^K \delta(i, j, k, m) W_R(i, j, k) X'(n, i)
\]  
(3.47)

\[
\delta(i, j, k, m) = \begin{cases} 
1 & \text{if } i \times j + k = m \\
0 & \text{otherwise}
\end{cases}
\]  
(3.48)

The model is used for modulating the harmonic distortion of the sound from the loudspeaker which is caused by the loudspeaker itself. The harmonics are multiples of the fundamental frequency. So if the fundamental frequency is \(f\) then the second harmonic will appear at two times the fundamental frequency \((2f)\), the third harmonic will appear at three times the fundamental frequency \((3f)\) and so forth. In the model we find \(M, H\) and \(K\) which are representing the number of subbands, the number of harmonics which are considered and the echo leakage respectively. To avoid the echo leakage it is recommended to set \(K = 1\). The fundamental frequency of the signal in this model is denoted as \(i\) and the actual number of harmonics is symbolized by \(j\). \(k\) is the index of the harmonic search window [22].

\(\delta(i, j, k, m)\) is used to decide which band the harmonics will end up in. \(W_R(i, j, k)\) is a part of the HDRES model and \(X'(n, i)\) is a delayed version of the speaker signal. To be able to make an estimation of the nonlinear echo the loudspeaker and the residual echo needs to be correlated. The correlation calculation complicates by the time delay that arises from the sound waves travelling through the echo path. To be able to handle this problem the regression is computed through a normalised transformation predicated by a delayed version of the speaker signal, \(|X'(n, m)|\). It is calculated as follows:

\[
X'(n, m) = \sum_{t=0}^{T-1} L(t, m) X(n - t, m)
\]  
(3.49)
\[ L(t, m) = \frac{|W_L(t, m)|}{\sum_{j=0}^{M}|W_L(j, m)|} \]  

(3.50)

To be able to continuously follow the development of the Harmonic distortion of the loudspeaker signal the HDRES need to be updated after each calculation. This can be done with any linear adaptive algorithm. In [22] this is done by a normalized LMS algorithm.

\[ \varsigma(n, m) = |E(n, m)| - |\hat{Y}_r(n, m)| \]  

(3.51)

\[ W_R(i, j, k) = W_R(i, j, k) + \frac{\mu}{P(n, m)} X'(n, m) \varsigma(n, m) \]  

(3.52)

As could be seen in the equations above, the error signal is calculated by a subtraction between the input of the linear AEC, \( E(n, m) \), and the residual echo estimation \( \hat{Y}_r(n, m) \). The error signal \( \varsigma(n, m) \) is then used to make a better estimation of the echo in the next calculation of the HDRES model quantity \( W_R(i, j, k) \). The step size \( \mu \) is used to set how fast the algorithm should converge. To make the algorithm more stable, the NLMS algorithm including a division of the average power of the speaker signal [22]. This is calculated by:

\[ P(n + 1, m) = (1 - \rho)P(n, m) + \rho|X'(n, m)|^2 \]  

(3.53)

3.4.6 Summary

The HDRES algorithm is a good choice when it comes to low usage of the computational capacity and fast convergence. It is therefore more suitable for hand held devices compared to other nonlinear echo cancellers in those aspects. According to the articles and [9] the RES algorithm has shown good performance to cancel nonlinear artefacts produced by the loudspeaker. However, the HDRES has a disadvantage similar to many other adaptive algorithms that is used for acoustic echo cancelling, and that is sensitivity for double talk. This means that if a near-end speech is present at the same time as the updating of the HDRES filter, a considerably amount of the near-end speech will be destroyed. The writers of the article [22] suggest a possible way to overcome this drawback. The idea is to go back a number of adaption steps when doubletalk occurs. That way the last adaption step that did not include any near-end speech could be used and the near-end speech will not be destroyed. This solution requires that the algorithm has access to a double talk detector [22].

Benefits/drawbacks of the method:

Benefits:
- Low complexity
- Fast convergence
- Small use of the computational capacity
- Suitable for hand held devices

Drawbacks
- Sensitive to doubletalk
4.1 Implementation in Matlab

4.1.1 Summary of the four methods
As stated above, the purpose of this thesis is to investigate different methods that deal with nonlinear acoustic echo cancellation. The four methods that were chosen to be investigated more closely are the Static method, Volterra filter, the Hammerstein model and HDRES. These methods were chosen because they represent a few of the most promising methods to cancel nonlinear echo. The static method has low computational complexity and since it is not adapting it has low computer power consumption. The drawback of using a static method is that adjustments have to be done for each mobile phone where it is implemented. The method might also remove more echoes than actually exist since it always models the worst case. The Volterra filter represents the general nonlinear modelling. The method has existed for a long time and is regarded reliable. It can model a large class of nonlinear system and has properties that are suitable for AEC. The drawbacks of this method is that it has high complexity and slow convergence. The Hammerstein model has similar properties to the Volterra filter and in some situations has the capacity to create a very accurate model. In resemblance with Volterra it demands high computational complexity and slow convergence. The HDRES model is different from the other models in that it is working in the frequency domain instead of in the time domain. This condition along with other adjustments gives this method other properties than the others, e.g. low computational complexity and fast convergence. These properties makes it more suitable for hand held devices and therefore is the first choice. However, all these methods have a problem with sensitivity to double talk. This property seems to be consistent for most methods that deal with nonlinear acoustic echo cancellation according to our study.

Thus, the HDRES model is chosen to be the main focus area from this point. The next step is to start the implementation of the selected method. The idea is to first implement the algorithm in Matlab, where it will be tested and analysed, and when it is good enough it will be converted to C. A flow chart of the HDRES algorithm is found in figure 4.1.

4.1.2 Changes from the article
The article “Nonlinear Residual Acoustic Echo Suppression for High levels of Harmonic Distortion” [22] that has been written by Diego A. Bendersky from the University of Buenos Aires and Jack W. Stokes, Henrique S. Malvar from Microsoft Research works as a template for the present implementation. Their method for nonlinear acoustic echo cancelation was developed for use in laptop computers. The intention of this study is to adjust the method to make it more suitable for hand held devices such as cell phones. The difference of implementing the HDRES method for use of a cell phone compared to a laptop computer is the amount of computer power that is available. In the HDRES article, calculation for harmonics is done for every frequency bin but in the present implementation this is done for every frequency band. The reason is that the algorithm will use less computer power. The drawback of this change is that the accuracy of the method will decrease. Another difference in the present implementation compared to the article is that instead of modulated complex lapped transform the Fast Fourier Transform (FFT) is used to go from time domain to frequency domain. This change is made to make the environment in Matlab, during the testing, similar to the one where it is going to be implemented in C later on.
4.1.3 Additional changes from development
During the development of the HDRES algorithm two more improvements were done, averaging and limiting of the weights. The averaging was done for every block to improve the stability of the algorithm. The reason for limiting the weights was stopping the algorithm from moving more energy to another band than what is possible for a real signal with harmonics.
Matlab – description of the code

4.1.4 Frequency banding
The banding takes the sum of eight frequency bins and put them into one frequency band. It considers FFT transformed values from 1 to N/2 to avoid redundancy, were N is size of the FFT. Extract of the Matlab code is shown in Table 4.1.
Table 4.1 Extract of the Matlab code for the calculation of the frequency band

```
for fb = 1:8:N/2
    Xk(ceil(fb/8)) = sum(abs(Xktemp(fb:fb+7)).^2);
    Drk_ref(1,ceil(fb/8)) = sum(abs(Drktemp(fb:fb+7)).^2);
end
```

4.1.5 Call for the mapping matrix
Table 5 shows a function call of the mapping matrix (MappingMatrix) function. The function has four inputs: The number of harmonics, the echo leakage, the frequency band size and the sampling frequency. The function itself is described in detail in the section of the mapping matrix. Extract of the Matlab code is shown in Table 4.2.

```
MappingMatrix = CalculateFreqBandMapping(H,L,fs/2/M,fs);
```

Table 4.2 Extract of the Matlab code for the call of the mapping matrix

4.1.6 Estimation
The calculation of the estimation is made by using the mapping matrix function which calculates which bands the harmonics are in. For those cases the estimation parameter is updated by a product of the weighting function and the loudspeaker input. An extract of the Matlab code is shown in Table 4.3.

```
for m = 1:M
    temp = 0;
    for i = 1:m
        for j = 1:H
            for l = -L:L
                if (MappingMatrix(m,i,j,l+(L+1)) == 1)
                    temp = temp + WR(m,i,j,l+(L+1))*Xprimek_masked(i);
                end
            end
        end
    end
    Drhatk(m) = temp;
end
```

Table 4.3 Extract of the Matlab code for the calculation of the estimation. M, m, H and L are number of frequency band, fundamental frequency band, number of harmonics and echo leakage respectively
4.1.7 Averaging
To compensate for the effects from the banding, averaging is used. Averaging is calculated by taking the previous spectrum added with the present spectrum and then dividing the sum with two. Extract of the Matlab code is shown in Table 4.4.

```matlab
if k == 1
drk=dr(k:k+79);
else
drk=(dr(k-80:k-1)+dr(k:k+79))/2;
end
```

Table 4.4 Extract of the Matlab code for the averaging calculation

4.1.8 Limiting the weights
This function has the purpose of limiting the weight so that more power than actually is the case for the harmonics is not transferred to the band. In the C implementation the weight limit (WeightLimit) is changed to the relative gain (Rg), which is the ratio between the gain of the fundamental and the harmonics. Extract of the Matlab code is shown in Table 4.5.

```matlab
if WR(m,i,j,l+(L+1)) > WeightLimit
   WR(m,i,j,l+(L+1)) = WeightLimit;
end
```

Table 4.5 Limiting the weights

4.1.9 The MappingMatrix function
The purpose of the MappingMatrix function is to determine what bands the harmonics may end up in for a certain frequency. The MappingMatrix is a three-dimensional matrix and each dimension can be divided into fundamental frequency, band number and number of harmonics. It includes all information about the overtones. The information is stored as ones and zeros in the matrix. In Table 4.6 the Matlab code is included that calculates the MappingMatrix.

```matlab
for i = 1:NbrFreqBands
   for j = 1:NbrHarmonics
      LowestBand = floor(FreqLowLimit(i)*j/Bandwidth);  
      HighestBand = floor(FreqHighLimit(i)*j/Bandwidth);  
      BandNumberIndex = (LowestBand:1:HighestBand)+1;  
      BandNumberIndex = BandNumberIndex(find(BandNumberIndex <= NbrFreqBands));  
      BandNumberIndex = BandNumberIndex(find(BandNumberIndex >= 1));  
      MappingMatrix(BandNumberIndex,i,j) = 1;
   end
end
```

Table 4.6 A part of the Matlab code that calculated the MappingMatrix

Thus, the MappingMatrix can be divided into X two-dimensional slices, where X stands for the number of overtones (harmonics) considered. The first overtone can be considered as an example: A table of the first overtone is shown in figure 4.2. The element which is denoted by one has the meaning that for that
fundamental frequency the overtone can exist in the corresponding band, e.g. A1 in the table is denoted by 1. That means that if the fundamental frequency lies between 0 and 249 Hz the first overtone may only exist in the first band. For the second overtone, see figure X, there are ones in both A1 and A2. This means that if the fundamental frequency lies between 0 and 249 Hz the second overtone may only end up in the first or the second frequency band. If six harmonics is considered there will be six tables like the ones which are shown in figure 4.2 and 4.3. The mathematical explanation of this behavior is the following:

4.1.10 Mathematical explanation
If the second overtone is considered and the fundamental frequency lies somewhere between 250 and 499 Hz there may be a fundamental frequency at 300 Hz in that range which may generate a second overtone at $300 \text{ Hz} \times 2 = 600 \text{ Hz}$. Thus a second overtone may lie in band 3. If instead a fundamental frequency that has a value of 400 Hz is considered, still in the same range, the second overtone may end up in $400 \text{ Hz} \times 2 = 800 \text{ Hz}$. Thus, the second overtone may also end up in the forth band. Compare this result with figure 4.2.

First overtone:

<table>
<thead>
<tr>
<th>Frequency bands</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.2 Table of the mapping matrix for the first overtone
Second overtone:

**Fundamental frequency**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Frequency bands**

- A & 1: 0 - 249 Hz
- B & 2: 250 - 499 Hz
- C & 3: 500 - 749 Hz
- D & 4: 750 - 999 Hz
- E & 5: 1000 - 1249 Hz
- F & 6: 1250 - 1499 Hz
- G & 7: 1500 - 1749 Hz
- H & 8: 1750 - 1999 Hz
- I & 9: 2000 - 2249 Hz
- J & 10: 2250 - 2499 Hz
- K & 11: 2500 - 2749 Hz
- L & 12: 2750 - 2999 Hz
- M & 13: 3000 - 3249 Hz
- N & 14: 3250 - 3499 Hz
- O & 15: 3500 - 3749 Hz
- P & 16: 3750 - 3999 Hz

Figure 4.3 Table of the mapping matrix for the first overtone
4.1.11 Matlab results

4.1.11.1 The input signal
The input signal which was generated in Matlab consists of a fundamental and a number of harmonics. By letting the fundamental frequency and the number of harmonics vary, the performance of the HDRES algorithm could be tested. The amplitude of the harmonics varies randomly but the amplitude for harmonics higher than three is significantly lower than the others. This was done to create a more realistic input signal. An example of the input signal can be observed in figure 4.4.

![Figure 4.4 The input signal created for testing of the HDRES algorithm in Matlab](image)

4.1.11.2 Estimation
To be able to cancel the harmonic distortion of the loudspeaker signal it must be estimated. In figure 4.5 and 4.6 the true residual echo and the estimated residual echo is presented. A comparison between the two figures shows that they are similar except for the first frequency band. The reason why the first band is not estimated is that there are no harmonics there for this case. Thus, the estimation of the harmonic distortion seems to be fairly good, at least for this kind of input signal. This result is important for the continuing implementations in C since the estimation part is one of the major parts that are going to be implemented in the AEC structure.
4.1.11.3 The error function
To be able to rate the performance of the estimation the error signal can be used. The error function specifies the difference between the true signal and the estimate. Thus, the lower value of the error function the better is the estimation. The figure 4.7 shows a plot of the error function for the true residual echo and its estimation. The error function is made for one chosen band.
Figure 4.7 The error function, gives the difference between the true residual echo and its estimation
4.2 C- Implementation
The need to convert the HDRES algorithm to C means replacing the harmonic nonlinear modeling part of residual echo control module in the current speech enhancement program implemented in C by the residual echo estimation part of the HDRES. The integration will make the harmonic distortion estimation code of the speech enhancement dynamic which automatically adapts itself according to the true residual echo. On the other hand, HDRES gets the benefit of using the already implemented linear echo canceler, noise estimator, time to frequency conversion, frequency banding and gain calculation from the speech enhancement program.

4.2.1 Signal Processing Library (SPL)
SPL is a library implemented by ST-Ericsson. It consists of different data types; functions which are common for signal processing operations such as FFT and IFFT. It also has basic mathematical and logical operations that can be applied to variables of data types defined in SPL. Most of the variables and functions used in the implemented AEC are from SPL instead of the standard C library.

Numeric data types in SPL are defined as structures which contain two variables; Q and v. A scalar number n is represented as

\[ n = v \times 2^Q \]

where v is 16 or 32 bit size.

For floating point implementation Q can vary, while in fixed point implementation Q is not allowed to vary. Mathematical operations on these numbers should be done carefully if standard C library functions are going to be used. SPL functions defined for appropriate data types can handle these issues by themselves.

4.2.2 Inputs to HDRES algorithm in C
Every computation in HDRES algorithm that is going to be implemented in C is in frequency domain and in frequency bands, so every input should be in frequency domain and frequency bands before the estimation begins. To make the C code more readable, three GET functions are defined to be called whenever the inputs are needed. The three inputs are:

4.2.2.1 Reference signal
Loudspeaker input signal after time to frequency domain conversion and then frequency banding is used as an input. This is the signal that is used as a regression variable for HDRES. It is available in the required format in REC block.

4.2.2.2 Residual Echo
The difference between linear echo estimate and microphone input signal will contain the residual echo, background noise and near end speech. This signal is one of the inputs for HDRES and is available as an output from Echo subtraction block.
4.2.2.1 Noise estimate
Background noise estimate is the third input for HDRES. It will be used to remove background noise from the output of echo subtraction block so that the algorithm will estimate residual echo only, instead of their sum. Noise estimation is done in noise reduction block. Hence the estimate is extracted from this block.

4.2.3 HDRES algorithm in C
The HDRES function definition is given below:

```c
void HDRES(SPL_vint32Q_t* SPL_Drhatk, double* Xprimek1, double* Ek1, double* Nk1, HDRES_DATA* HData)
```

The function has five arguments where SPL_Drhatk is used as output of the function which holds residual echo estimate. Xprimek1 is the loudspeaker input signal. Ek1 is the difference between the echo estimate and microphone input signal. Nk1 is the noise estimate and HData contains previous values of variables that will be used in the algorithm.

Data type of the first argument and last arguments are pointers to structures defined by the key word struct in C programming while the rest are pointers to double.

The body of the HDRES algorithm in C is very similar to the MATLAB except some syntax changes.

The major differences were:
- Variables should be declared and initialized before they are used in the case of C which was not the case in MATLAB.
- Copying values into vectors should be done by loops in C where a simple equal sign was enough in MATLAB if the vectors have equal size.

Residual echo estimation code in C is given in Table 4.7 to compare it with the MATLAB code given above.

```c
for (m = 0; m < M; m++)
    temp = 0;
    for (i = 0; i < m; i++)
        for (j = 0; j < H; j++)
            if (MappingMatrix[m][i][j] == 1)
                temp = temp + currentWR[m][i][j]*Xprimek_masked[i];
    Drhatk[m] = temp;
```

Table 4.7 Residual echo estimation

The weight updating code in C as shown in Table 4.8 is not allowed to update weights all the time. The algorithm is allowed to update if the weight is less than the possible limit provided in vector Rg (Relative gain). Otherwise the maximum possible value given in Rg will be set.

Relative gain is set by customers by measuring the ratio of power of overtones to power of the fundamental frequency. Implementing this condition check will guarantee that no power is transferred to a band by the loudspeaker more than is physically possible.
Table 4.8 Updating of the weights

4.2.4 Location of HDRES
The HDRES algorithm is defined as a function in a header file and stored along with other header files of the speech enhancement implementation. It is called when estimating harmonic nonlinearity caused by loudspeaker in REC.

HDRES is logically located in REC (Residual Echo Control) block as shown in figure 4.8; specifically in the place where residual echo estimation is done. Nonlinearity in loudspeaker is modelled as harmonic nonlinearity and spectral leakage. The harmonic nonlinear computed by HDRES and spectral leakage are summed and then Gain computation is carried out.

```c
for (i=0; i < m; i++){
    if (Xprimek_masked[i] > 0) {
        for (j=0; j < H; j++){
            if (MappingMatrix[m][i][j] == 1){
                WR[m][i][j] = currentWR[m][i][j] +
                (mu/currentPk[i])*Xprimek[i]*errk[m];
                if (abs(WR[m][i][j])>abs(Rg[j])){
                    WR[m][i][j] = Rg[j];
                }
            }
        }
    }
}
```

Figure 4.8 Block diagram of acoustic echo canceller
4.2.5 Output of HDRES

Output of HDRES in C is estimated residual echo in frequency domain and frequency banded. Moreover the datatype should be converted to SPL; a code that converts residual echo estimate stored in double vector Drkhatk to SPL type is given in Table 4.9.

```c
SPL_vint32Q_t* SPL_Drhatk;
SPL_Drhatk->Q = 0;
SPL_Drhatk->L = 16;
SPL_Drhatk->v_p = Drhatk;
```

Table 4.9 Double to SPL type conversion
CHAPTER 5 RESULTS
5.1 Introduction
In summary, the article about an algorithm called “Harmonic Distortion Residual Echo Cancelation” written by Diego A. Bendersky, Jack W. Stokes and Henrique S. Malvar was chosen for further investigation. The algorithm was first implemented in Matlab where it was tested and improved. When the Matlab implemented algorithm showed promising results it was converted into C. The C-program was then included in the AEC structure along the static method to cancel the nonlinear echo. To evaluate the performance of the HDRES algorithm it was running through the simulation computer program that was included in the AEC structure. A number of different measurements were set as an input to the simulation program. The measurements were captured from cell phones which were documented to have problems with nonlinear echo. The measurements were taken from the loudspeaker and the microphone input on the cell phones. From the simulation program three different outputs have been generated; the output from the linear AEC (which is the same as the input to the HDRES algorithm and the static nonlinear echo canceller), the output of the HDRES algorithm and the static method. The HDRES is first compared to the output of the linear AEC to distinguish how much of the nonlinear echo that is cancelled by the HDRES algorithm. The comparison will be made for both hand held and hands free mode. At this point it is only the echo reduction that is considered. This result can be seen as the rate of the performance of the algorithm to cancel nonlinear echo. The next comparison is made between the HDRES and the static nonlinear echo canceller. This is made to evaluate the performance of the two methods in the aspect of cancelling nonlinear echo. Except for cancellation of nonlinear echo the handling of double talk is of interest for the evaluation. The HDRES handling of the double talk situation will be compared to the output of the linear AEC and the static acoustic echo canceller.

5.2 Results for echo subtraction
To give a general idea how the simulation is performed a signal plot of the loudspeaker and the microphone input is shown in figure 5.1 and 5.2 respectively. The signal which is captured from the loudspeaker contains a one minute long signal of speech with different amplitudes. The microphone input signal is a result of the sound from the loudspeaker travelling through the acoustic room before it is captured by the microphone. These sounds that are captured by the microphone are not desirable since they will generate an echo of the speech coming from the far-end communicator. Thus, the far-end communicator will hear himself in his mobile phone. For this reason the expectation of a good echo canceller would be that it cancels all acoustic echoes. Thus, the signal from the microphone input will be zero.

Note that in the result part only a few chosen inputs/outputs are shown. For a more complete evaluation the reader is referred to the appendix C where more and magnified plots are presented.
5.2.1 Hand held mode

Figure 5.1 Illustration of the loudspeaker input signal. It contains speech with different amplitudes

Figure 5.2 Illustration of the microphone input signal

Figure 5.3 Illustration of the output of the linear AEC

As can be seen in figure 5.3 not all echoes are cancelled by the linear AEC. The idea is to let the nonlinear echo canceller, in this case the HDRES algorithm or the static method, take care of the residual echo. In figure 5.4 the output of the HDRES is shown. In this figure almost all residual echo is gone which means that the result fulfils the expectations.

If the result of the HDRES algorithm is compared to the static method which is shown in figure 5.5, the result is that they are almost similar regarding the amount of cancelled residual echo.
Figure 5.4 Illustration of the output of the HDRES algorithm

Figure 5.5 Illustration of the output of the static method output

To have a better look at the result a magnified view of the output of the linear AEC, the HDRES and the static method is shown in figures 5.6, 5.7 and 5.8 respectively. In this comparison it is clearer how much of the residual echo from the linear AEC is reduced. It is also visible that the static method has a better performance in cancelling residual echo than the HDRES.

Figure 5.6 Illustration of the output of the linear AEC magnified
5.2.2 Hands free mode case

For the hands free mode the same comparison as for the hand held mode is made. The loudspeaker input, microphone input, AEC output, HDRES output and the Static method output are illustrated in figures 5.9, 5.10, 5.11, 5.12 and 5.13 respectively.
Figure 5.10 Illustration of the microphone input signal

Figure 5.11 Illustration of the output of the linear AEC

Figure 5.12 Illustration of the output of HDRES

Figure 5.13 Illustration of the static output
The results from the hands free mode are similar to the results for the hand held mode. Thus, almost all the residual echo is cancelled. The comparison between the output from the HDRES model and the static method gives a similar result. If the plots for the HDRES output and the static model output are magnified the result will be like in figure 5.14 and figure 5.15. In this comparison it is more clear that the static model gives a better result in cancelling the residual echo, although the difference is small.

![Figure 5.14 Illustration of the HDRES output magnified](image)

![Figure 5.15 Illustration of the static output magnified](image)

### 5.2.3 Results for the double talk situation
For this comparison a signal which contains double talk is considered. Thus, there are both acoustic echo from the loudspeaker and near-end speech present at the same time. In figure 5.16 the output from the linear AEC is illustrated. It is containing the mostly near-end speech but also some residual echo. The purpose of the residual echo canceller is to take care of the residual echo without cancelling the near-end speech. As mentioned earlier, adaptive methods are sensitive for double talk, and so is the HDRES algorithm. This is quite clear if the red marked areas in the plots are compared to each other. The area is containing near-end speech mixed with residual echo which gives the result that the HDRES simply takes away everything, as illustrated in figure 5.17. The static method, which is illustrated in figure 5.18, has a better performance in this case.
Figure 5.16 Illustration of the output of the linear AEC

Figure 5.17 Illustration of the output of the HDRES

Figure 5.18 Illustration of the output of the static method
**Performance measurement:**

The most common method to measure performance of acoustic echo canceller is Echo Return Loss Enhancement (ERLE).

ERLE shows how much of the echo captured by the microphone is cancelled by the acoustic echo canceller in dB scale. It is defined as the ratio of residual echo power to echo power at the microphone [11].

$$ERLE = -10\log\left(\frac{E[e^2]}{E[y^2]}\right)$$

where $e$ and $y$ are residual echo and microphone input signals respectively. A good AEC is expected to have a high value of ERLE. ERLE is used only for single talk scenarios.

Since this thesis focuses on nonlinear acoustic echo cancellers, measuring their performance irrespectively of the linear AEC used before them is possible if the ERLE is modified to Nonlinear Echo Return Loss Enhancement (NERLE) defined as the ratio between power of the output from nonlinear echo canceller and power of the residual echo (output of the linear AEC). Hence the following NERLE definition is used to measure the performances of the HDRES algorithm and static method [11]:

$$NERLE = -10\log\left(\frac{E[r^2]}{E[e^2]}\right)$$

where $r$ is output of the nonlinear echo canceller and $e$ is the residual echo.

**Hands free Mode**

Performances of the HDRES algorithm and the static method when measured by NERLE are shown in Figures 5.19 and 5.20 respectively. Both methods have high NERLE values showing their nonlinear echo cancelling performance [11].
Hand held Mode

Figures 5.21 and 5.22 show NERLE measurement of the HDRES and the static method respectively. The two algorithms have higher NERLE values in hands free mode than in hand held mode which implies that they perform well for hands free mode. This is because nonlinear echo is more prominent in hands free mode than in hand held mode [11].
Figure 5.22 NERLE of the static method in hand held mode (single talk)
6.1 Conclusion
This thesis is considering the fact that an adaptive method is needed to cancel residual nonlinear echo. The question that was asked in the problem statement was:

- Is there any adaptive method to cancel nonlinear acoustic echo in a way that is suitable for mobile phones?

To answer this question an adaptive algorithm named HDRES, short for Harmonic Distortion Residual Echo Suppression been implemented in an operating acoustic echo cancelling system and has been tested.

The project was initiated by a number of measurements being performed on three different mobile phones that were documented to have problems with nonlinear acoustic echo. The measurements were analysed to characterize the problem of nonlinearities produced by mobile phone loudspeakers. When the problem was identified an investigation was started in order to look for already published articles that were discussing nonlinear acoustic echo cancellation. From this wide field of different concepts three articles were chosen for a closer investigation. From those articles one method was chosen to become implemented and tested.

The result of the implementation of the chosen method was an algorithm that showed good performance in cancelling nonlinear echo. The simulation testing that was done showed that almost all residual echoes were cancelled. However, the sensitivity to doubletalk of the method that was stated in the article was shown to be correct. At some parts of the signal where doubletalk was presented, the near-end speech was reduced or even completely erased. Some attempts were made to minimize the sensitivity of the doubletalk, e.g. weight limitation, but with no results.

When the HDRES is compared to the static nonlinear canceller, it can be said that the HDRES has a similar performance to the static method in cancelling nonlinear echo. However, the performance in handling doubletalk was significantly better for the static method.

To answer the question, it might be possible to use an operating adaptive algorithm to cancel the nonlinear echo in an AEC system, which has both low computational complexity and fast convergence. However, the prerequisite for such a system to work would be that a doubletalk detector is implemented along with the adaptive algorithm. That way the doubletalk situation could be found and the adaptation of the algorithm could be stopped. Thus, the major part of the speech would be saved.
6.2 Further work
The problem of echo in general and nonlinear echo in particular is significant in the ever-increasing development of hands free communication devices. As the trend in the technology is to make everything wireless and portable, it is evident that this problem will get worse in the future due to cheap loudspeakers. In this thesis, ways of cancelling nonlinear echo are presented. Moreover the selected algorithm for implementation has cancelled most of the nonlinear echo. But further improvements can be done to handle the following:

- Double talk
- Other types of non linear distortion such as spectral leakage in addition to harmonic distortion.
- Minimization of near-end speech distortion

Finally the complexity of the algorithm is not tested by running it on a cell phone, which should be done before heading to use it on real applications.
References


[8] Klippel, W. “Loudspeaker Nonlinearities – Causes, Parameters, Symptoms” Dresten, Germany


Appendix A Matlab Code

The Matlab code of the Harmonic Distortion Residual Echo Suppression algorithm.

```matlab
% Initialization
MIN_GAIN = 0.01;          % Minimum Gain allowed
fs = 8000;                 % sample rate
N = 256;                   % N point FFT size
M = 16;                    % Number of frequency bands for 8kHz sampling frequency
H = 6;                     % Number of harmonics including fundamental band mapping
L = 0;                     % Echo leakage window (Denoted by K on the paper)
mu = 0.07;                 % step size
beta = 1;                  % tunes aggressiveness of the Gk(m) algorithm
alpha = 0.5;               % controls amount of smoothing
rho = 0.25;                % power estimate smoothing
Sk = zeros(1,M);           % Speech in the near end
Nk = zeros(1,M);           % Noise in the near end
Nks = zeros(2,M);          % Noise (two rows needed to store the current and previous noise)
Drk = zeros(1,M);          % Reference residual echo signal
Drks = zeros(2,M);         % Smoothed estimation of residual echo signal
Ek = zeros(1,M);           % Output of the linear AEC
Eks = zeros(2,M)+eps;      % Smoothed update of output of the linear AEC
Xk = zeros(1,M);           % frequency banded input signal in frequency domain
Pk = zeros(2,M)+eps;       % Average power of speaker input signal
WR = zeros(M,M,H,2*L+1);    % Weight for Residual echo estimation
ThresholdPowerRxdBovl = -30;
ThresholdPowerRx = 10^(ThresholdPowerRxdBovl/10)*(32768)^2*N*N*8/2;
Blocksize = 80;
MaxValue = 0;
WeightLimit = 0.15;

% Initialization: Input signals
f = 536;                   % fundamental frequency
num_harmonics = 6;         % number of harmonics
w0 = 2*pi*f/fs;
n = (0:10000);
fvec = (0:128-1)*8000/256;
Dr = 0;                    % dr starts with the value 0
A_org = 1*(2^(15)-1);     % = 32767
min_value1 = 0.5;         % min random value 1
max_value1 = 1;           % max random value 1
min_value2 = 0;           % min random value 2
max_value2 = 0.5;         % max random value 2
ref=(1:128)*8000/256;

% Initialization: Estimated residual echo
temp = 0;

% Check for harmonics limit
if (f*num_harmonics > fs/2)
    error('To many harmonics considered');
end

% Create the input signal
x = A_org*sin(w0*n);       % Loudspeaker input signal
for j = 1:num_harmonics    % dr=SUM[A*sin(w0*n+P)] (summation of sinusoidal)
```

73
%give a random value between 0.5 - 1 for the 2nd and the 3rd harmonics
A = A_org*(min_value1 + (max_value1-min_value1) * rand(1));

if j >= 3
    %give a random value between 0 - 0.5 for harmonics >= 3
    A = A_org*(min_value2 + (max_value2-min_value2) * rand(1));
end

%generates a phase shift between 0 and 2*pi
Phase=rand(1)*2*pi;

%CREATE THE SUMMATION
b = A*sin((j+1)*w0*n+Phase);
dr = dr+b;
end

dr = dr + rand(1)*x;  %Remember that the residual echo signal includes x to some extent.
dr = dr*1/max(dr)*(2^(15)-1);  %Normalize to amplitude 1.

%----------------------------- COMPUTE THE HDRES ALGORITHM ---------------------------------

MappingMatrix = CalculateFreqBandMapping(H,L,fs/2/M,fs);  %calling a function
for k = 1:blocksize:length(x)-79  %Compute estimated residual echo
    %shift the previous values to 2nd row to put the current values in the 1st
    %one block contains 80 samples, 10ms in 8khz sampling frequency
    Eks(2,:) = Eks(1,:);
    Nks(2,:) = Nks(1,:);
    Drks(2,:) = Drks(1,:);
    Pk(2,:) = Pk(1,:);

    %Current block
    xk = x(k:k+79);
    drk = dr(k:k+79);

    %############ Averageing #################################################################
    %To compensate for the effects from the baning averageing is used.
    %Averaging = (previous spectra + current spectra)/2
    if k == 1
        drk=dr(k:k+79);
    else
        drk=(dr(k-80:k-1)+dr(k:k+79))/2;
    end

Xktemp = fft(xk,N);
Drktemp = fft(drk,N);

%Frequency banding (1 frequency band = 8 frequency bins)
%consider only 1 to N/2 FFT transformed values, to avoid redundancy
for fb = 1:8:N/2
    Xk(ceil(fb/8)) = sum(abs(Xktemp(fb:fb+7)).^2);
    Drk_ref(1,ceil(fb/8)) = sum(abs(Drktemp(fb:fb+7)).^2);
end

Nkhat = abs(zeros(1,M)).^2;  %From Noise floor estimation algorithm
Ek = Nk + Sk + Drk_ref;  %No delay in this case

Xprimek = Xk;

for mm = 1:length(Xprimek)
    if (Xprimek(mm) < ThresholdPowerRx)  %Prevent bands not powerful enough to contribute to Harmonics
        Xprimek_masked(mm) = 0;
    else
        Xprimek_masked(mm) = Xprimek(mm);
    end
end

%POWER SMOOTHING
for m = 1:M
    Pk(1,m) = (1-rho)*Pk(2,m) + rho*((Xprimek(m))^2);  
    if(Pk(1,m)< Minimum_power)
        Pk(1,m) = Minimum_power;
    end
end

for m = 1:M
    %FREQUENCY BAND
    temp = 0;  %Initialization for each f-band
    for i = 1:m  %fundamental frequency band
        for j = 1:H  %number of harmonics
            for l = -L:L  %echo leakage
                if (MappingMatrix(m,i,j,l+(L+1)) == 1)
                    temp = temp + WR(m,i,j,l+(L+1))*Xprimek_masked(i);
                end
            end
        end
    end
    Drhatk(m) = temp;  %Estimated Residual Echo
    if (Drhatk(m) < 0),
        Drks(1,m)= (1-alpha)*Drks(2,m);
    else
        Drks(1,m)= (1-alpha)*Drks(2,m) + alpha*(Drhatk(m));
    end

    %smoothing  %second row in Ek, Drk, Nk, Pk refers to the previous signal
    Eks(1,m) = (1-alpha)*Eks(2,m) + alpha*(Ek(m));
    Nks(1,m) = (1-alpha)*Nks(2,m) + alpha*(Nkhat(m));

    %Gain computation
    Gk(m) = max(max(Eks(1,m)-beta*Drks(1,m),Nks(1,m))/abs(Eks(1,m)),MIN_GAIN);

    %error computation for adaptation of WR
    errk(m) = Ek(m) - Drhatk(m);

    for i=1:m
        if (Xprimek_masked(i) > 0)
            for j=1:H
                if (MappingMatrix(m,i,j == 1)
                    WR(m,i,j) = WR(m,i,j) + (mu/Pk(1,i))*Xprimek(i)*errk(m);
                end
            end
        end
    end
end

%End for each frequency band
The Matlab code of the mapping matrix

```matlab
function [MappingMatrix] = CalculateFreqBandMapping(NbrHarmonics, WindowLength, Bandwidth, SampleRate)
% Calculate the mapping of contributing frequency band.

SampleRateBy2 = SampleRate/2;
NbrFreqBands = SampleRateBy2/Bandwidth;

FreqLowLimit = Bandwidth*(0:1:NbrFreqBands-1);
FreqLowLimit(1) = 30; % Approximative lowest frequency of usage - DEPEND ON FFT RESOLUTION (2 bin)
FreqHighLimit = Bandwidth*(1:1:NbrFreqBands)-1;

% l = 0 - No leakage or search window applied
for i = 1:NbrFreqBands,
    for j = 1:NbrHarmonics;
        LowestBand = floor(FreqLowLimit(i)*j/Bandwidth);
        HighestBand = floor(FreqHighLimit(i)*j/Bandwidth);
        BandNumberIndex = (LowestBand:1:HighestBand)+1;
        BandNumberIndex = BandNumberIndex(find(BandNumberIndex <= NbrFreqBands));
        MappingMatrix(BandNumberIndex, i, j) = 1;
    end
end
```

end %End for each frame
Appendix B Measurement plots

Frequency sweep
White noise
Loudspeaker input. Frequency sweep with a high amplitude, frequency domain
Microphone input. Frequency sweep with a high amplitude, frequency domain.
Loudspeaker input. Frequency sweep with a low amplitude, frequency domain.
Microphone input. Frequency sweep with a low amplitude, frequency domain.
Loudspeaker input. White noise, frequency domain.
Microphone input. White noise, frequency domain
Loudspeaker input. Frequency sweep with a high amplitude, frequency domain.
Microphone input. Frequency sweep with a high amplitude, frequency domain
Loudspeaker input. Frequency sweep with a low amplitude, frequency domain
Microphone input. Frequency sweep with a low amplitude, frequency domain
Measured with an external microphone. Loudspeaker input, frequency sweep, frequency domain.
Measured with an external microphone. Microphone input, frequency sweep, frequency domain.
Appendix C Result plots

Illustration of the loudspeaker input signal. It contains speech with different amplitudes.
Illustration of the output of the linear AEC magnified
Illustration of the output of the HDRES output magnified
Illustration of the output of the static output magnified
Illustration of the loudspeaker input signal. It contains speech with different amplitudes
Illustration of the microphone input signal
Illustration of the output of the linear AEC
Illustration of the HDRES output magnified
Illustration of the output of the linear AEC
Illustration of the output of the HDRES
Illustration of the output of the static method