ISSN 0280–5316
ISRN LUTFD2/TFRT--5618--SE

Digital Loudspeaker Equalization

Andrej Petef
Jimmie Landerman

Department of Automatic Control
Lund Institute of Technology
May 1999

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 Lund Sweden

Document name
Master Thesis

Date of issue
May 1999

Document Number
ISRN LUTFD2/TFRT--5618--SE

Author(s)
Andrej Petef and Jimmie Landerman

Supervisor
Björn Wittenmark

Sponsoring organisation

Title and subtitle
Digital Loudspeaker Equalization. (Digital frekvensutjämning av högtalare).

Abstract

In this thesis we present different methods for improving the performance of a loudspeaker using digital signal processing. Three different approaches on DSP equalizing techniques are compared and evaluated both with computer simulations and by subjective listening tests. General considerations on equalizing as well as through background to the different methods are presented. Gained knowledge on loudspeaker measurements, inverse systems and DSP hardware is also presented.

Two different loudspeakers in a free-field environment are compared. Computations were based on measured impulse response and issues like computational efficiency and error measure were considered. The study is limited to linear magnitude and phase equalization.

The results are discussed and thoroughly presented with plots, and further possibilities with loudspeaker signal processing are presented. An example of a completely digital reproduction system is also given as an example of future audio systems.

It is shown that considerable improvements of the loudspeaker performance can be achieved using FIR filtering techniques. Both measurements and subjective evaluation shows satisfying results, but also points out the careful steps that should be taken when deciding what to actually compensate for.

A goal for the thesis have been to give the reader a good understanding of the whole equalizing process, from loudspeaker measurement to the equalized result.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
67

Recipient's notes

Security classification

Table of contents

1. Introduction

The speaker is definitely the weakest link in an audio reproduction chain, and to verify that by measurements is straightforward. By correcting the amplitude and phase response of the speaker we should expect some measurable and audible improvements.

The possibility of doing this digitally will give us methods for doing much more accurate corrections and to make the filters easily changeable. We can also use filter types that were never possible to realize with analog components. We will in this study analyze and compare three different filtering methods and implement these in DSP real-time hardware to verify the results. Our main purposes for the project have been to investigate and to gain knowledge about all the parts included in such a correction system, and to draw conclutions on what kind of filtering should be used. The report concentrates on linear filtering of a loudspeaker considered as a linear and time-invariant system (LTI-system), through compensation for nonlinear distortion could be made using nonlinear techniques.

Though the entire path from source to listener is important, we will here concentrate only on correcting the loudspeaker in a free field situation. The filters are aimed to be the inverse transfer function of the loudspeaker and could be used in the digital audio-data stream or in the analog low-level signal path.

The technique is very new to the audio industry, and is so far only implemented in a handful loudspeakers on the hi-fi market, all very expensive, and in some stand-alone boxes on the professional market. This is though believed to be the future of high quality audio and a widely expanding field in audio engineering.

Similar works have been presented in a few papers, like [1], [2], [3] dealing with adaptive systems, [4] concentrating on *warped* filters, [5] dealing with recursive filters.

First, we will give an overview of some loudspeaker characteristics and discuss the general equalization of a loudspeaker. We look at the generating of the loudspeakers response and the degree and type of correction that should be done.

In chapter 3 we look through some gained knowledge on inverse systems and the inversion of non-minimum phase systems.

In chapters 4, 5, and 6 the background to the three filter types is analyzed and the implementation and results for each method are discussed.

We present a summary and take a look at the future possibilities of the technique in chapter 8.

In Appendix we take a look at the DSP hardware used, and the necessary programming is presented. We provide some considerations on measurement techniques, and the used *Matlab* scripts and assembler programming is listed.

2. Speaker Equalization

We should start with putting up some obvious goals for a system that would equalize the response of a speaker:

- Obtain flat amplitude and linear phase response from the loudspeaker.
- Increase the degree of freedom in loudspeaker design. When designing a speaker we must consider many requirements such as the control of amplitude frequency characteristics, directivity, acoustic power output, and nonlinear distortion. The purpose is to free us from at least the amplitude frequency characteristics.
- Design a system whose frequency characteristics could be changed at will by the listener.

Every speaker can be considered as a *multi-mode vibrator*, and will have specific points of resonance and therefor the speaker output will have amplitude and phase variations at these frequencies. The speaker will also have phase-shifts along with frequency. It is virtually impossible to compose an accurate mathematical model of a speaker from physical principles, due to the complexity of those principles involved. Therefor the inversion of a loudspeaker *model* is impossible, and we can only make some general assumptions about the system. We are left with the possibility to do measurements of the speaker's performance, and this way try to work out a way to compensate for its flaws.

To be able to pre-process the signal, we must first have a good view of the speaker's characteristics, and different ways of achieving this is presented in the *Measurements* appendix. We will use the speaker impulse response and from this analyze its frequency response with the discrete fourier transform (DFT).

Generation of the impulse and frequency response

We have worked with two different speakers called the small speaker and the big speaker. They are described in appendix B. The impulse response of the small speaker, measured as described in appendix A,can be seen below.

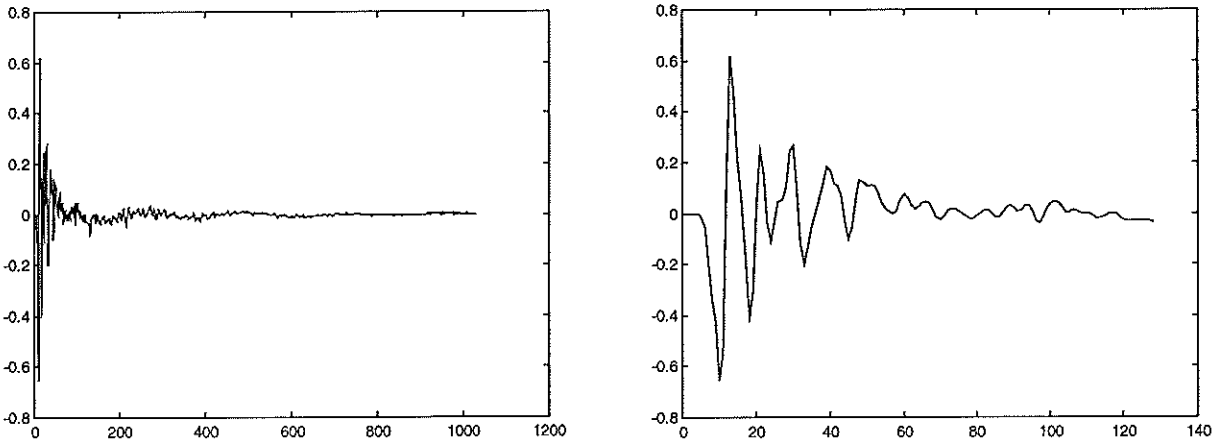


Figure 2.1: Impulse response for small speaker shown with 1024 and 128 samples.

Because of the relatively high noise level in the measurements, long impulse responses are not particularly useful, and to obtain accurate information about low-frequency response is not possible. The wavelength of 20Hz is represented by 2200 samples, and it's obvious we can't get frequency resolution in the lower bass region with the shorter impulse responses we're using.

The corresponding frequency response is shown in figure 2.2.

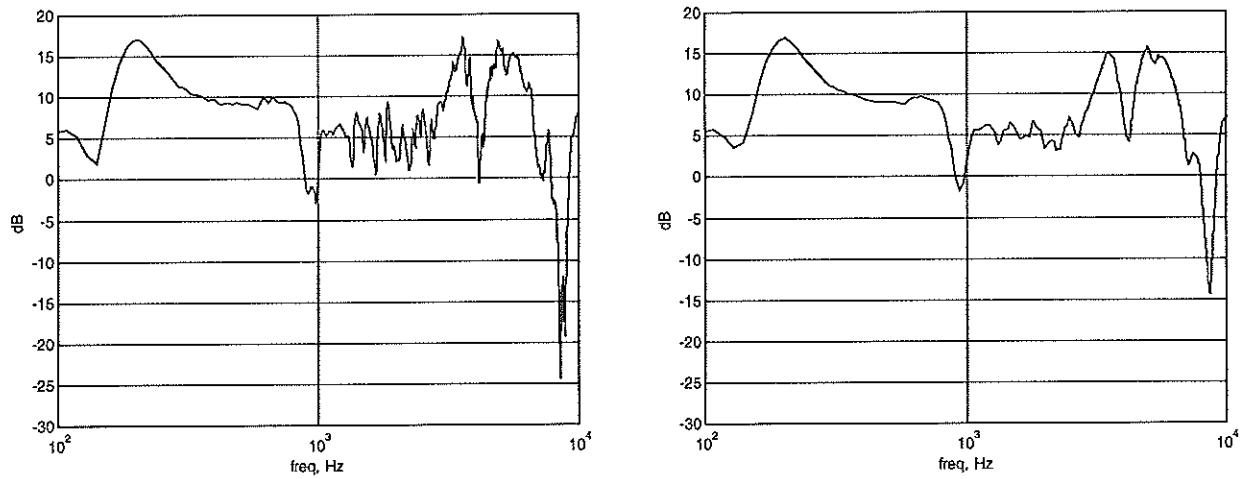


Figure 2.2: Frequency response for small speaker with 4096 point FFT. To the left we have the unsmoothed response, and to the right with 1/8-octave smoothing.

This impulse response is indeed a near field measurement to suppress the acoustical influence of early room reflections. We see that we have great variations in amplitude over the whole frequency band, and a couple of sharp notches. The raw frequency response is somewhat difficult to analyze, and the smoothed one is generally used.

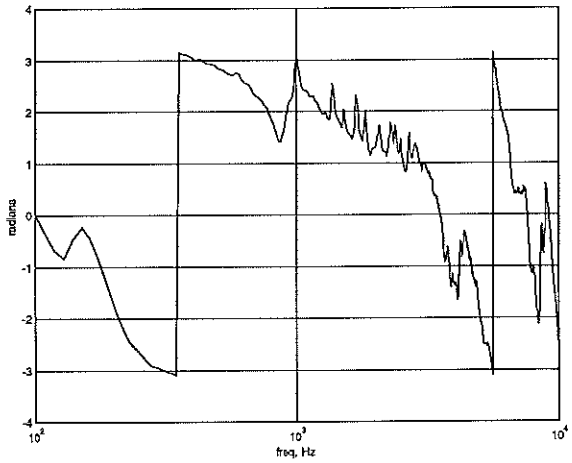


Figure 2.3: The phase response of the small speaker.

The phase response of the speaker is showed to the left. It is generally hard to analyze the phase plot because of noise and delay in the system. An example of a linear phase plot is showed in figure 2.5.

The responses of the bigger 3-way speaker are shown below in figure 2.4.

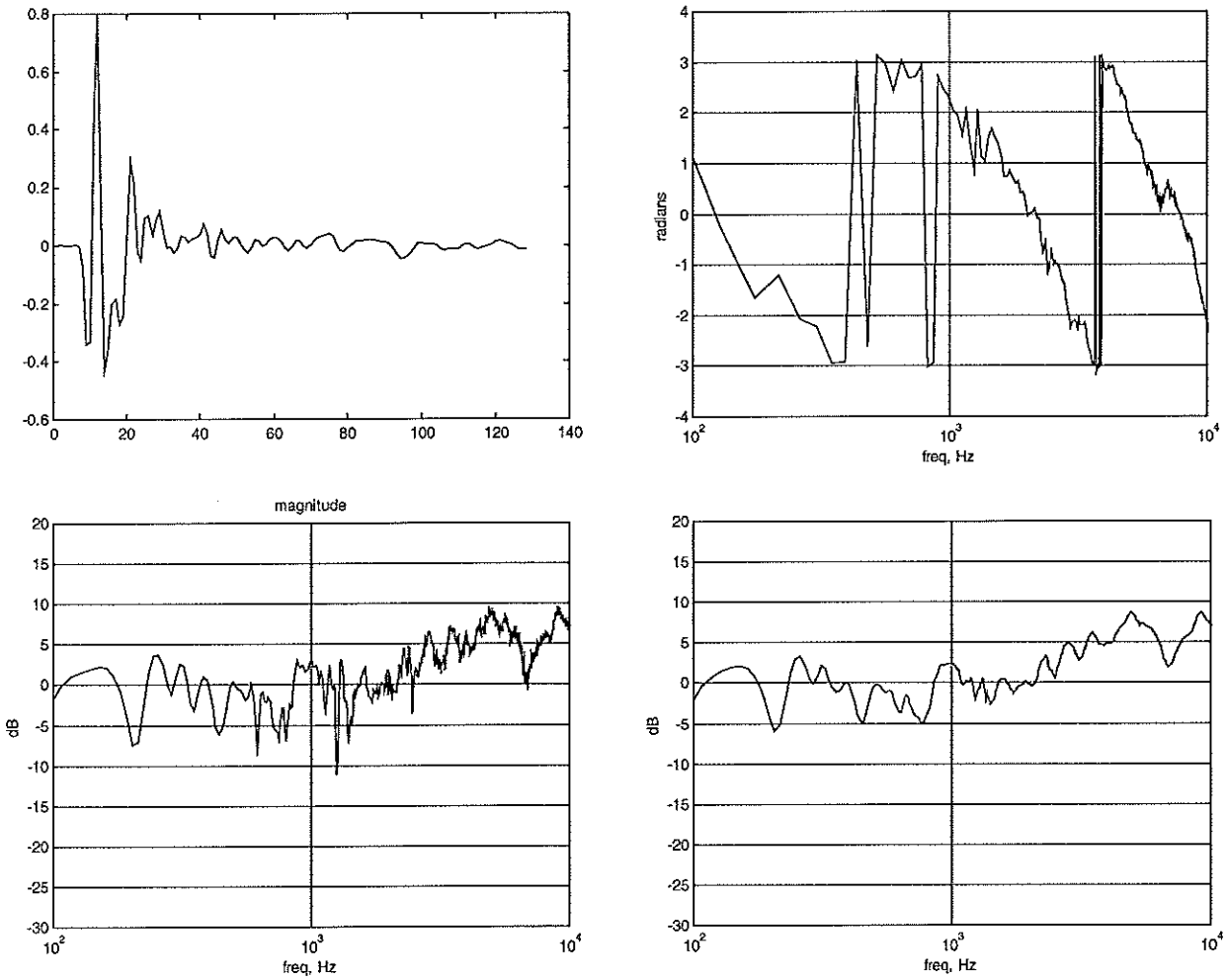


Figure 2.4: Impulse response, phase response, amplitude response and smoothed amplitude response for big speaker. 4096 point FFT and 1/8-octave smoothing.

In figure 2.4 we see a more complicated impulse response that could arise from bad alignement of the drivers and from the complex cross-over filter, among other things.

Degree of equalization

One of the first things we need to consider is the degree to which we need to equalize the system. Different types of amplitude variation affects the perceived sound quality to various degrees, and a perfectly flat amplitude response for a single point in space may not always be the optimal solution.

It is known from experience that sharp notches in the response introduce less coloration than peaks do. Consequently the filter should seek to accurately remove peaks, but can be less accurate about correcting sharp notches without significantly affecting the perceived performance.

Also, if the frequency characteristic of the speaker has sharp dips at certain frequencies, the amount of compensation becomes very great at these frequencies. This could cause a power amplifier to saturate, and to avoid this the input signal must be decreased. Then the signal-to-noise ratio of the system would decrease considerably.

Sharp dips in near-field measurements are also likely to arise from problems when pressure-

waves are out of phase and thus extinguish each other. This dip will probably vary both with distance and angle from speaker, and correction for these deviations could make the result even worse elsewhere.

As we listen over a wider area in space (or probably in our living room) we should always make a tradeoff between the accuracy of the correction at one point and the listening area over which it is valid. Making several measurements in relevant positions and weighting those together would be a preferable method. In this project though we have several practical problems with such a setup, and since we are more interested in the comparison of different methods we will only do single point measurements.

Another aspect is that we hardly have any reliable low-frequency response under 100Hz, since the dynamic range of the measurement system is poor and we also have to truncate the response to avoid room-interactions. A 1024 sample response gives us a frequency resolution of 43Hz, and we would need more than 2200 samples to get 20Hz resolution. Thus we should here limit our frequency analysis to above 100Hz. Trying to correct the lack of bass extension for a small speaker could also be a fast way of overheating and damaging the driver.

Since the speakers, at least the small one, have no significant output above 10kHz, and since the real-time system we have for evaluation is also limited to frequencies below that, we simplify the system compared to standard digital audio systems and restrict our analysis to below 10kHz.

We could therefor aim our correction towards a target function with flat amplitude response and linear phase, i.e a bandpass filter which response is shown below in figure 2.5.

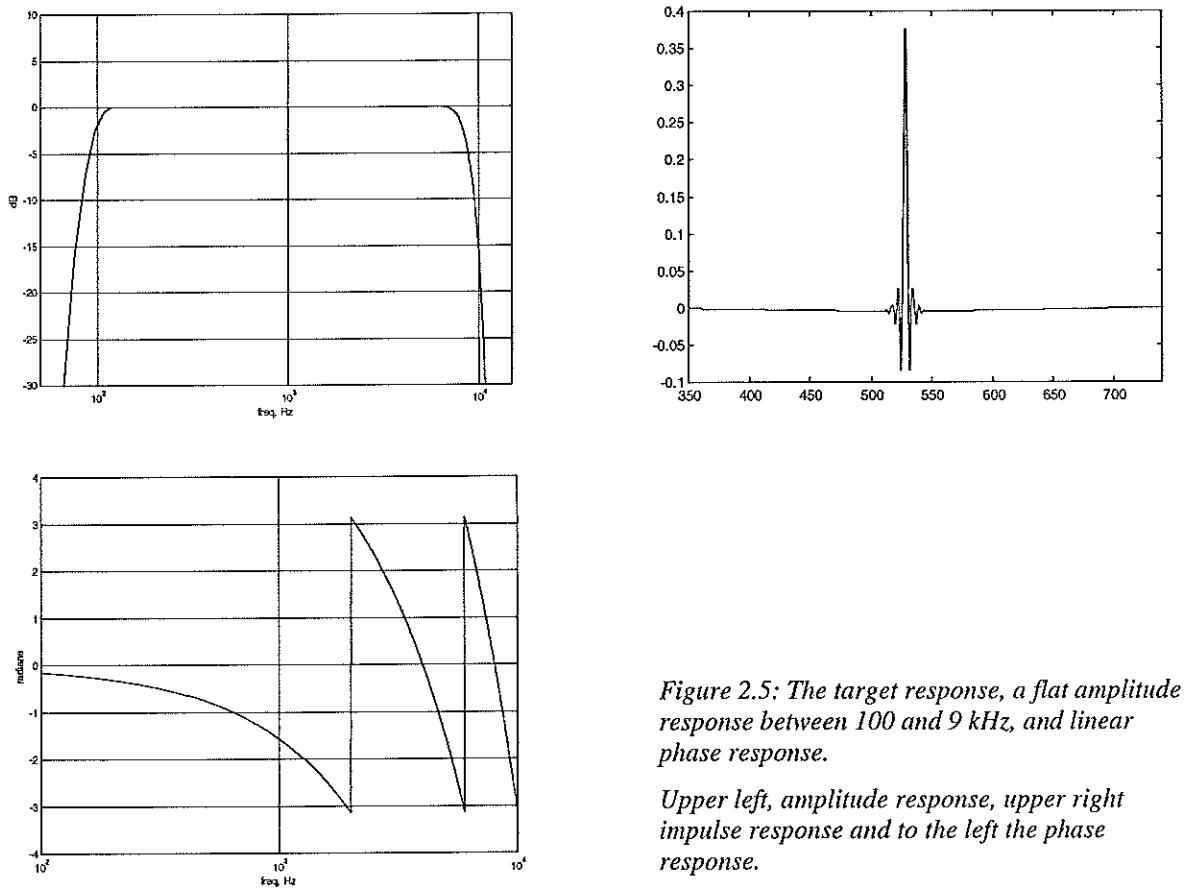


Figure 2.5: The target response, a flat amplitude response between 100 and 9 kHz, and linear phase response.

Upper left, amplitude response, upper right impulse response and to the left the phase response.

Different Methods

These amplitude and phase variations can be compensated for, and is to some degree being done, in the analog domain by speaker designers today. There are systems that have a low-level compensating network, incorporating feedback signals from drivers etc. Often is amplitude and phase equalizing done in the cross-over network in the speaker.

Far better would be to do it with digital signal processing, which would produce a much more flexible and accurate result.

Previous work that has been done in the field include [1],where they have been using a combination of FIR and IIR filters for compensating sepatately for the minimum phase and excess phase parts. In [2] a frequency inversion technique have been realized, [3] is realizing an adaptive system witch produces a flat frequency response and linear phase response in the listening position. [4] does a basic study on warped IIR and FIR filters and [5] is concentrating on recursive IIR filters for amplitude equalizing. All reports, exept [3], are dealing with equalizing of the loudspeaker in free field conditions.

An investigation of what seems to be possible solutions gives us three obvious methods to evaluate:

- Frequency Inversion.
- Least Squares with an offline solution.
- Adaptive LMS algorithm with a possible online adaptation.

3. Design of a Filter

3.1 Inverse Systems

The design of a digital filter for correcting a system *H(z)* is a search for the inverse characteristics of the system to be corrected. The inverse is then used as a filter cascaded with the input signal and would theoretically produce a transparent system with a perfect impulse response. A system with system function *H(z)* has the corresponding inverse system function $H_i(z)$ which is defined to be the system with system function such that if it is cascaded with *H(z)*, the overall system function will be unity:

$$G(z) = H(z)H_i(z) = 1 \tag{3.1}$$

$$H_i(z) = \frac{1}{H(z)} \tag{3.2}$$

In the time domain:

$$g(n) = h(n) * h_i(n) = \delta(n) \tag{3.3}$$

Any linear time invariant (LTI) system can be written in the following form:

$$H(z) = \frac{a_0}{b_0} \frac{\prod_{k=0}^{M}(1 - c_k z^{-1})}{\prod_{k=0}^{N}(1 - d_k z^{-1})} \tag{3.4}$$

$$H_i(z) = \frac{b_0}{a_0} \frac{\prod_{k=0}^{N}(1 - d_k z^{-1})}{\prod_{k=0}^{M}(1 - c_k z^{-1})} \tag{3.5}$$

Where the c_k's are the nonzero zeros and the d_k's the nonzero poles of the system. The inverse of the transfer function *H(z)* is as the interchanging of the poles with the zeros, and vice versa.

If *H(z)* is stable and causal, as a loudspeaker is, all of the poles must be located inside the unit circle (otherwise the response would not be stable). But there are no such restrictions about the zeros, which can be located both inside, outside and on the unit circle. Since the inverse system is an interchanging of the zeros with the poles of the original system, it means the inverse will be stable and causal only for systems which have all their zeros inside the unit circle. Such systems, with a stable inverse, are called minimum phase systems.

3.2 MINIMUM PHASE SYSTEMS

A minimum phase system has some properties worth mentioning. For all systems $H(z)$ with a causal and stable impulse response that have the same magnitude response as a minimum phase system $H_{min}(z)$, the partial energy of the impulse response are delayed the least for the minimum phase system.

$$\left|H_{\min}(e^{j\omega})\right| = \left|H(e^{j\omega})\right| \tag{3.6}$$

$$\sum_{k=0}^{n}\left|h_{\min}(k)\right|^2 \geq \sum_{k=0}^{n}\left|h(k)\right|^2 \tag{3.7}$$

As n reaches infinity all of the systems with the same magnitude response have equal partial energy.

The minimum phase system also have the minimum group delay for all frequencies.

$$grd[H_{\min}(e^{j\omega})] < grd[H(e^{j\omega})] \tag{3.8}$$

Since of all possible pole-zero systems with the same magnitude response, there is only one minimum phase system it seems reasonable that the phase response of a minimum phase system can be totally derived from the magnitude response. There is a unique relationship between the magnitude response and phase response of a minimum phase system that is derived from the discrete Hilbert transform relationship.

$$\arg[X(e^{j\omega})] = -\frac{1}{2\pi}\wp\int_{-\pi}^{\pi}\ln\left|X(e^{j\omega})\right|\cot\left(\frac{\omega-\theta}{2}\right)d\theta \tag{3.9}$$

Where $\wp$ denotes the Cauchy principal value of the integral that follows.

Obviously, there is no problem to make a causal inverse of a minimum phase system, unfortunately loudspeakers are normally not minimum phase. The cross-over networks and the design with multiple drivers positioned at different places on the loudspeaker cabinet generally results in non-minimum phase systems. Loudspeakers with just a single driver that lack cross-over networks could more easily be minimum phase system, but doesn't necessarily have to.

3.3 NON-MINIMUM PHASE SYSTEMS

A non-minimum phase system with no zeros located on the unit circle can always be expressed as a minimum phase system cascaded by an allpass system with constant amplitude response:

$$H(z) = H_{\min}(z)\cdot H_{ap}(z) \tag{3.10}$$

The minimum phase part is obtained by reflecting all zeros that is located outside the unit circle as their conjugate reciprocals inside the unit circle.

If zeros of H(z) which is located outside the unit circle is denoted c_{J+1} to c_M the minimum phase part and the allpass part will be as follows:

$$H(z)=\underbrace{\frac{a_0}{b_0}\frac{\prod_{k=0}^{J}(1-c_k z^{-1})\cdot\prod_{k=J+1}^{M}(1-\frac{1}{c_k^*}z^{-1})}{\prod_{k=0}^{N}(1-d_k z^{-1})}}_{H_{\min}(z)}\cdot\underbrace{\frac{\prod_{k=J+1}^{M}(1-c_k z^{-1})}{\prod_{k=J+1}^{M}(1-\frac{1}{c_k^*}z^{-1})}}_{H_{ap}(z)} \tag{3.11}$$

By inverting the minimum phase part $H_{min}(z)$, the result will be an inverse $1/H_{min}(z)$ that cascaded with the system $H(z)$ have a flat magnitude spectrum but with a phase response as $H_{ap}(z)$:

$$G(z)=H_{\min}(z)\cdot\frac{1}{H_{\min}(z)}\cdot H_{ap}(z)=1\cdot H_{ap}(z) \tag{3.12}$$

This method to derive a filter is used in our Inverse Minimum phase filter described in chapter 4.3.

Another way to express a non-minimum phase system is to split it in a maximum phase part and a minimum phase part. A maximum phase system has all its zeros located outside the unit circle.

$$H(z)=H_{\min}(z)\cdot H_{\max}(z) \tag{3.13}$$

If zeros of H(z) that are located outside the unit circle are denoted c_{J+1} to c_M the minimum phase part and the maximum phase part will be as follows:

$$H(z)=\underbrace{\frac{a_0}{b_0}\frac{\prod_{k=0}^{J}(1-c_k z^{-1})}{\prod_{k=0}^{N}(1-d_k z^{-1})}}_{H_{\min}}\cdot\underbrace{\prod_{k=J+1}^{M}(1-c_k z^{-1})}_{H_{\max}} \tag{3.14}$$

We can now make a separate inversion of the minimum phase part $H_{min}(z)$ and one of the maximum phase part $H_{max}(z)$. $H_{i\text{-}max}$ is the inverse of $H_{min}(z)$ and is causal and stable. The inverse of $H_{max}(z)$ results in a noncausal stable inverse $H_{i\text{-}max}(z)$.

$H_{i\text{-}max}$ can be expressed by being expanded in power of z using long division:

$$H_{i-\max}(z)=\frac{1}{H_{\max}(z)}=\frac{1}{\prod_{k=J+1}^{M}(1-c_k z^{-1})}=a_0 z+a_1 z^2+a_2 z^3\ldots \tag{3.15}$$

The expanded serie will have infinite length which means we have to truncate the sequence to wanted length L to be able to introduce a delay that will make this filter causal. The result is an estimate of $H_{i\text{-}max}$ called $\hat{H}_{i-\max}$ with finite length L:

$$\hat{H}_{i-\max}(z)=a_0 z+a_1 z^2+\ldots+a_{L-1}z^L \tag{3.16}$$

By multiplying this estimate with a pure delay with L sample, z^{-L} the inverse is made causal and the resulting filter will be:

$$\hat{H}_i(z)=z^{-L}\cdot H_{i-\min}(z)\cdot\hat{H}_{i-\max}(z) \tag{3.17}$$

This is a causal estimate to an inverse of the system $H(z)$. The distortion compared to a perfect

inverse, comes from the introduced delay and the truncating of the infinite sequence. In chapter 4.2 we have derived a filter, called Inverse Filter, using this method

A question may arise; What if the system, we are trying to find the inverse of, contains zeros located on the unit circle? This is not an uncommon fact, as example most common digital filters have zeros located exactly on the unit circle. In that case a stable inverse can not be made, neither causal nor non-causal, cause the inverse would then include poles on the unit circle. This can easily be understood by the fact that a zero on the unit circle means a dip to zero in the magnitude response at the corresponding frequency. The magnitude of the inverse have at that frequency a magnitude that multiplied to zero should be one, and no such number exists. A loudspeaker have dips close to zero in the magnitude response at very low frequencies and very high frequencies. Since we are using a bandpass targetfunction, we overcome that problem.

3.4 CONCLUSIONS

We have seen that it is not possible to design an exact stable and causal inverse of a non-minimum phase system, but by accepting some distortion of the equalized system $G(z)$ a filter can be designed. Two ways of doing this were described, basically to introduce a delay (and some smaller artifacts due to the truncating of an infinite sequence) or by accepting a nonlinear phase response of the equalized system.

Both of the above listed options should be considered. A slight delay of the system is normally not a problem, if not the delay is so long that it will be noticeable in systems when, for example, picture and sound is synchronized. A nonlinear phase is much less audible than a distorted magnintude response.

4. Frequency inversion

With frequency inversion we mean methods to find the inverse characteristics of a system in the frequency domain by use of the discrete fourier transform (DFT) and its inverse (IDFT).

This is as close as we get to analytically find the inverse with the z-transform, which would be to hard because of the problem to factorize a 1023:th-order polynomial, which corresponds to the 1024 samples long FIR system we estimate the speaker with.

The DFT corresponds to the Fourier transform by samples, equally spaced in frequency, of the continuous Fourier transform.

Fourier transform:

$$H(e^{j\omega}) = \underset{z=e^{j\omega}}{H(z)} = \sum_{-\infty}^{\infty} x(n) \cdot e^{-j\omega n} \tag{4.1}$$

Discrete Fourier transform:

$$X(k) = \sum_{n=0}^{N-1} x(k) \cdot e^{-j2\pi kn/N} \quad k = 0,1,2,...N-1 \tag{4.2}$$

Inverse Discrete Fourier transform:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{j2\pi kn/N} \quad n = 0,1,2,...N-1 \tag{4.3}$$

As already mentioned we use a target function instead of making a real inverse, because of the limited bandwidth of the speaker.

There is two obvious ways to design a target function to the frequency inversion methods. Doing it in the time domain and then take the DFT of that or directly in the frequency domain. We have used a linear phased bandpass FIR filter designed in the time domain, because it was easy to design and the same filter can be used for all equalization method we have used in this report. We did some tests with bandpass filters designed in the frequency domain, but no good results were obtained, because it's hard to just design a response in the frequency domain "by hand", it will give "discontinues" in the amplitude response.

The bandpass filter were designed in the time domain as a Hanning windowed, linear phase FIR filter with cutfrequencies at 100 Hz and 9 kHz. The response of the filter can be seen in figure 2.5

By taking the Fourier transform of the speakers impulse response *x(n)* and the defined target function *d(n)*, the Fourier transform of the filter *f(n)* is:

$$H_{filt}(e^{j\omega}) = \frac{D(e^{j\omega})}{X(e^{j\omega})} \tag{4.4}$$

4.2 Inverse Filter

A nice property of the IDFT is that it will not produce an unstable sequence if not the DFT have some infinity elements i.e not any poles located on the unit circle, as can be seen from the formula of the IDFT which is a sum over a finite number of elements.

Any noncausal part will be wrapped around in the timewindow that is defined by the IDFT, n=[0...N-1] to the end of the sequence. Since the IDFT produces a periodic sequence of length N, the sequence can be rotated which is equal to a shift in time (pure delay). Because of this we

don't have to care about splitting the system in a maximum phase and a minimum phase part. The inverse of the maximum phase part will be non causal and wrapped around the time window.

Problems arise because the inverse of a FIR system is always a IIR system. By taking the DFT of a sequence $x(n)$ we automatically estimate the system as a FIR system or a "periodic FIR system", because the DFT is taken during a finite time.

When calculating the inverse in the frequency domain and then take the IDFT of the inverse, the result is a new FIR system. The length of this system will have N samples i.e. just as many samples as the DFT had. The IIR system, which arise from the inverse of a FIR system (which was an estimation of IIR in the first place!), will beacause of the DFT be estimated as a FIR system. This can be choosen to have arbitrarily length by padding the original sequence with zeros.

The estimation of a IIR system as a FIR system is not a problem if the number of samples of the FIR system is so that the infinite response have declined close to zero. For our small speaker the impulse response has declined after 1024 sample (see figure 2.1) .

Figure 4.1 and 4.2 is an example of the inversion with DFT/IDFT of some simple signals, one minimum phase, one maximum phase and one mixed phase. As can be guessed from the figure, the mixed phase signal in figure 4.2 is a convolution of the minimum phase signal and the maximum phase signal in figure 4.1. The minimum and maximum phase signal is only two samples long but needs at least 10 sample to estimate its inverse because of the IIR structure of the inverse.

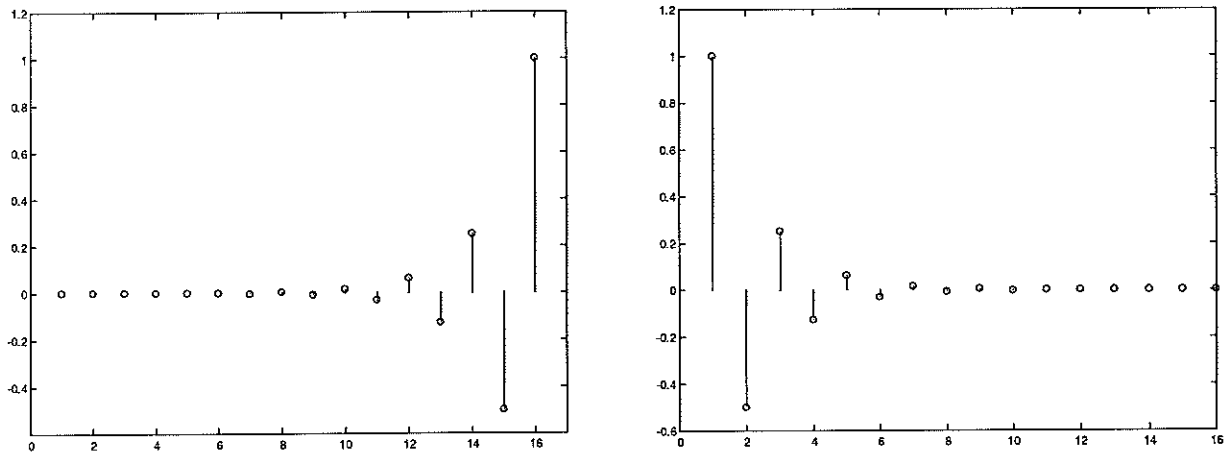


Figure 4.1: A maximum (left) [0.5 1] and a minimum phase (right) [1 0.5] FIR systems inverse by IDFT (1/DFT).

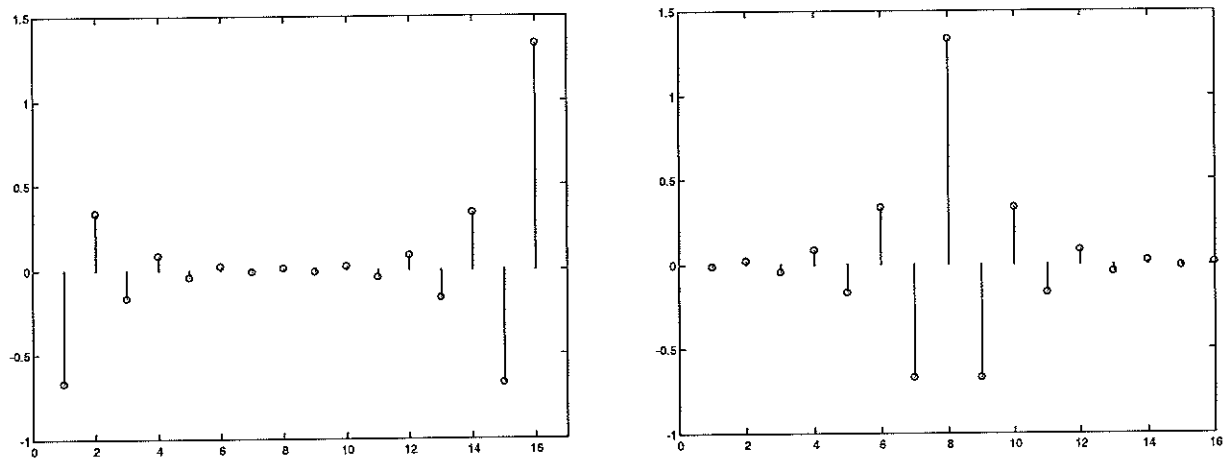


Figure 4.2: A mixed phase FIR system [0.5 1.25 0.5] inverted to the left and then rotated to obtain the best result (right).

Our method of designing this kind of filter, which we call "inverse filter", is shown as a flowchart in figure 4.3 there *x(n)* is the loudspeaker estimated as a FIR system. *d(n)* is the desired response and *f(n)* the ready to use filter. The rotation of the result should be done so the amplitude in the beginning and in the end of the time window have declined and is close to zero, as seen in figure 4.2.

We can design a filter with arbitrarily length by either taking the discrete fourier transform of exactly as many sample of *x(n)*, as we want our resulting FIR filter to be. Or take many (more) samples padded with zeros and then window the long resulting filter to wanted length. Trying both methods, the conclusion was made that the second alternative is generally the best and gives more control about what is happening.

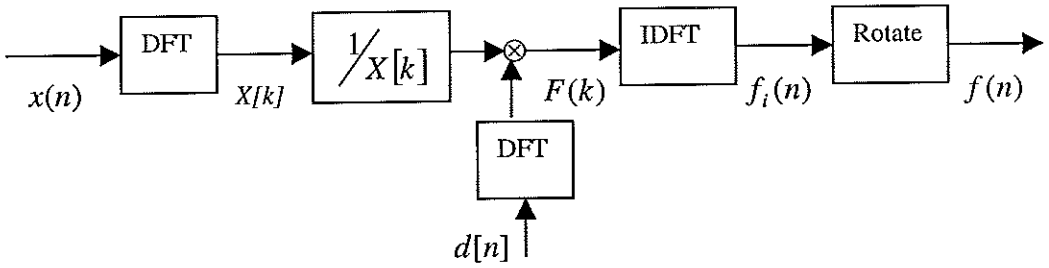


Figure 4.3: Flowchart to our inverse filter design

4.3 *INVERSE MINIMUM PHASE FILTER*

To design a filter that is the inverse of the minimum phase part with the same magnitude function as the FIR-system we have estimated the loudspeaker with, we must somehow derive the minimum phase part of the signal. This can be done with homomorphic filtering [6].

Transformation of a signal into its cepstrum is called homomorphic transformation. The cepstrum corresponding to a sequence $x(n)$ is the sequence $\hat{x}(n)$ whose z-transform is the logarithm of the z-transform of $x(n)$. If using the more practical fourier transform, the cepstrum of $x(n)$ is:

$$\hat{x}(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[\log\left|X\left(e^{j\omega}\right)\right| + j\arg\left(X\left(e^{j\omega}\right)\right)\right]e^{j\omega n}d\omega \qquad (4.5)$$

However to derive the minimum phase part we just need the real cepstrum of the sequence, which is calculated as the inverse fourier transform of the magnitude response only:

$$c_x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\log\left|X\left(e^{j\omega}\right)\right|e^{j\omega n}d\omega \qquad (4.6)$$

The complex cepstrum of the minimum phase part of $x(n)$ is calculated as seen in figure 4.4.

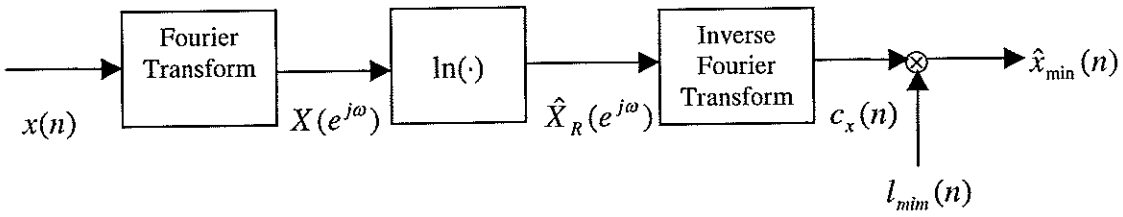


Figure 4.4 : Minimum phase complex cepstrum

and from the minimum phase complex cepstrum, the searched mimimum phase part of $x(n)$ can be found:

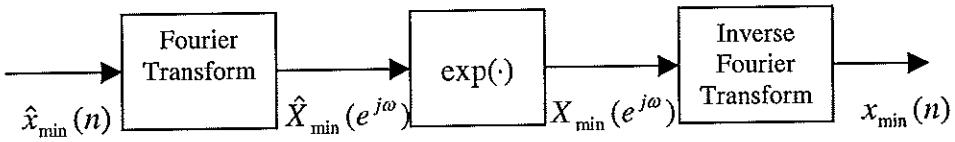


Figure 4.5 : Minimum phase part of x(n).

The $l_{mim}(n)$ sequence is coming from the following relationship between the real and complex cepstrum:

$$c_x(n) = \frac{\hat{x}(n) + \hat{x}(-n)}{2} \qquad (4.7)$$

if $\hat{x}(n)$ is causal, then:

$$\hat{x}(n) = c_x(n)l_{mim}(n) \quad \text{if} \quad l_{mim}(n) = 2u(n) - \delta(n) = [1\ 2\ 2\ldots] \qquad (4.8)$$

Calculating this is an easy task in *Matlab*, using the DFT. Since we calculating the filter in the frequency domain we directly use $X_{\min}(e^{j\omega})$ as:

$$H_{filt}(e^{j\omega}) = \frac{D(e^{j\omega})}{X_{\min}(e^{j\omega})} \tag{4.9}$$

4.4 RESULTS

We have constructed filters with the two methods just described for two different speakers, see Appendix B. The first one, which the filters will be evaluated on, is the small speaker with a single driver unit and without cross-over networks. FIR filters with lengths of 256 and 1024 tap have been the main case, even though both shorter and longer filters have been tried.

Both the minimum and maximum phase filters were designed with 4096 points for later being truncated down to the wanted size.

As can be seen in figure 4.6 both of the filters end up with the most energy in the end of the time window. This shouldn't happen to the minimum phase inverse, if it weren't because of the targetfunction we have used.

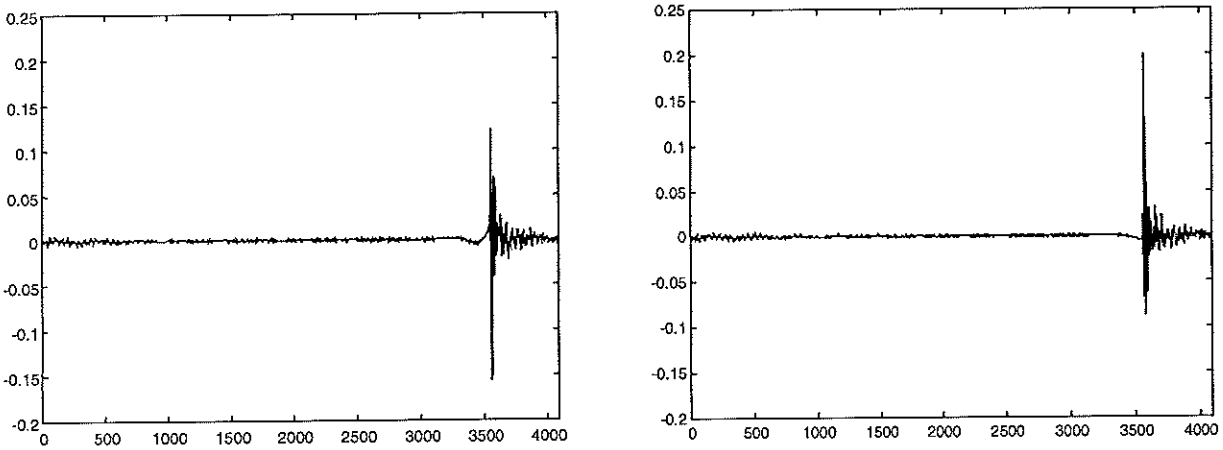


Figure 4.6: To the left is the inverse filter and to the right the inverse minimum phase filter for the small speaker.

So both filters have to be rotated, which is possible because of the circular properties of the discrete fourier transform. We also truncate them to 1024 samples, which is the longest filter we will evaluate

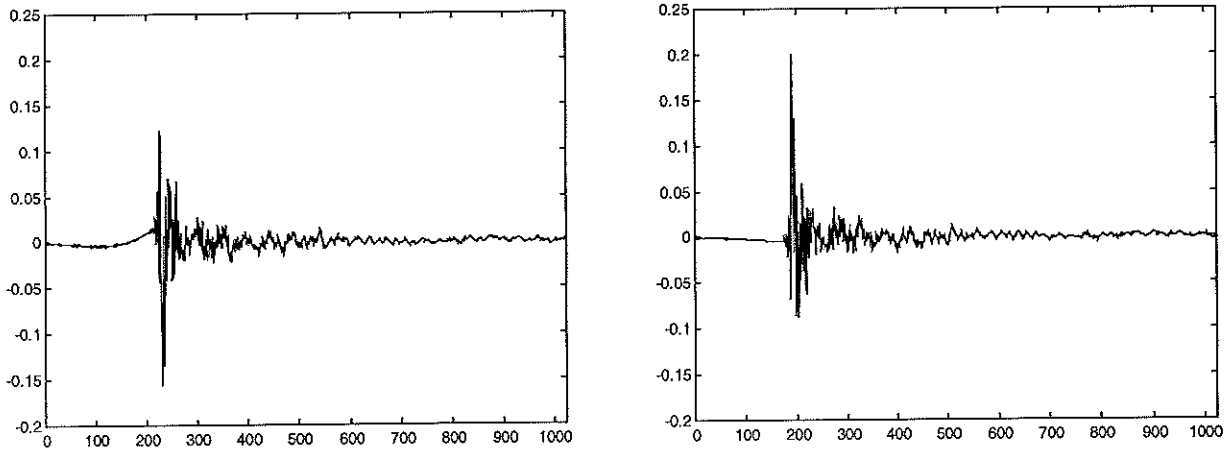


Figure 4.7: Here are the filters rotated and truncated to 1024 samples.

There is no big difference to window this 1024 taps filters (figure 4.7) with anything else than a rectangular window. This is because the amplitude of the filters in the end of the timewindow is already close to the background noise in our equipment, so there will not be any abrupt ending in the convoluted signal.

The shorter 256 tap filter, will of course end with larger amplitude and could thus be a more interesting case for windowing. We have tried some different windows, mainly Bartlett, exponential, and Hanning. The results for the different filters were similar. To use an asymmetric exponential or asymmetric Hanning window seemed to give slightly better result than the triangular bartlett window and definitely better results than just truncating, i.e. using a rectangular window. Here is asymmetric Hanning window used shown in figure 4.9.

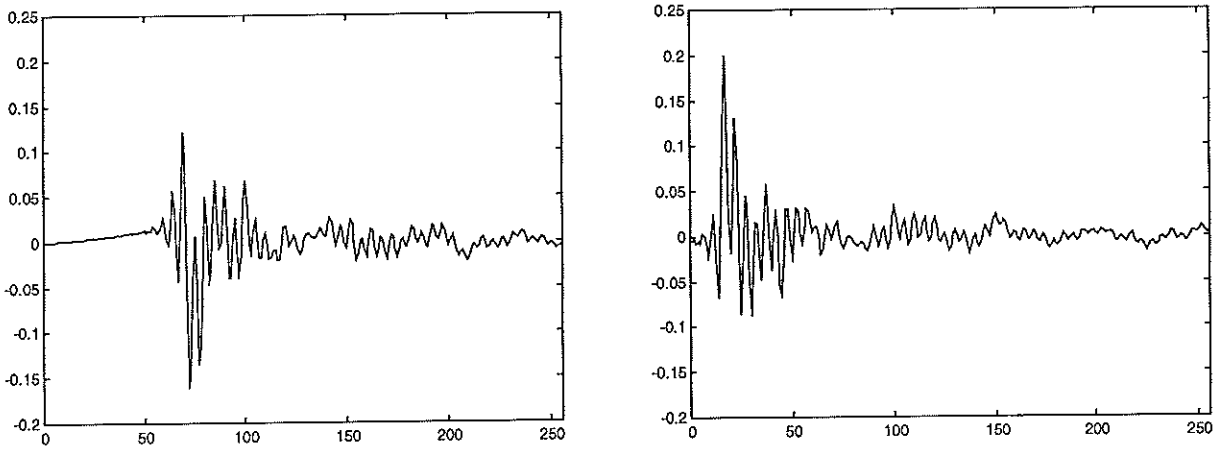


Figure 4.8: The shorter 256 tap filters have a higher amplitude in the end which encourage the use of other windows than rectangular shown here.

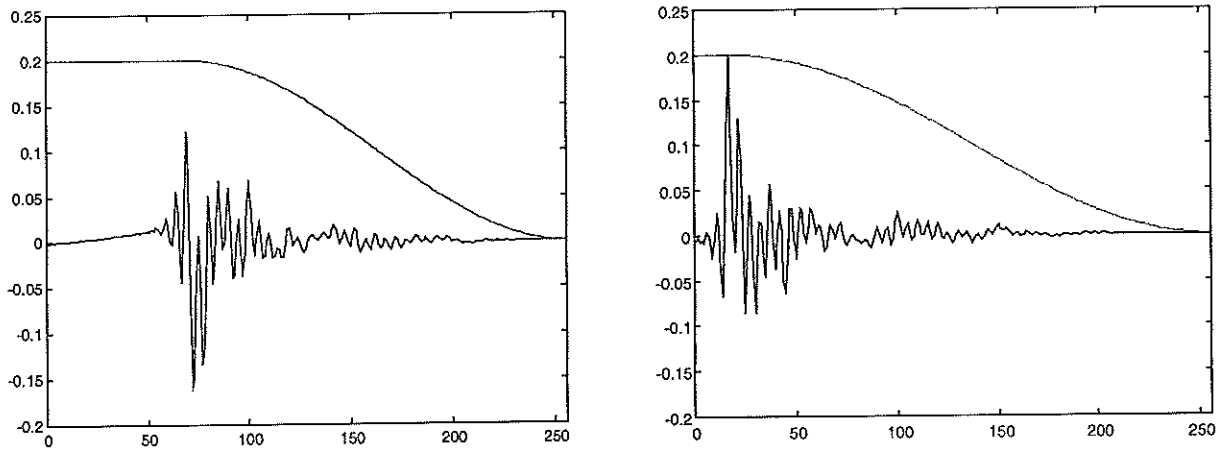


Figure 4.9: The filters windowed with an asymmetric Hanning window (dotted).

The filters and the magnitude function of the small speaker, should be mirrors of each other. This can be seen in figure 4.10 there the 1024 tap filter is compared with the speaker. Observe that the inverse filter and the inverse minimum phase filter have the same magnitude function.

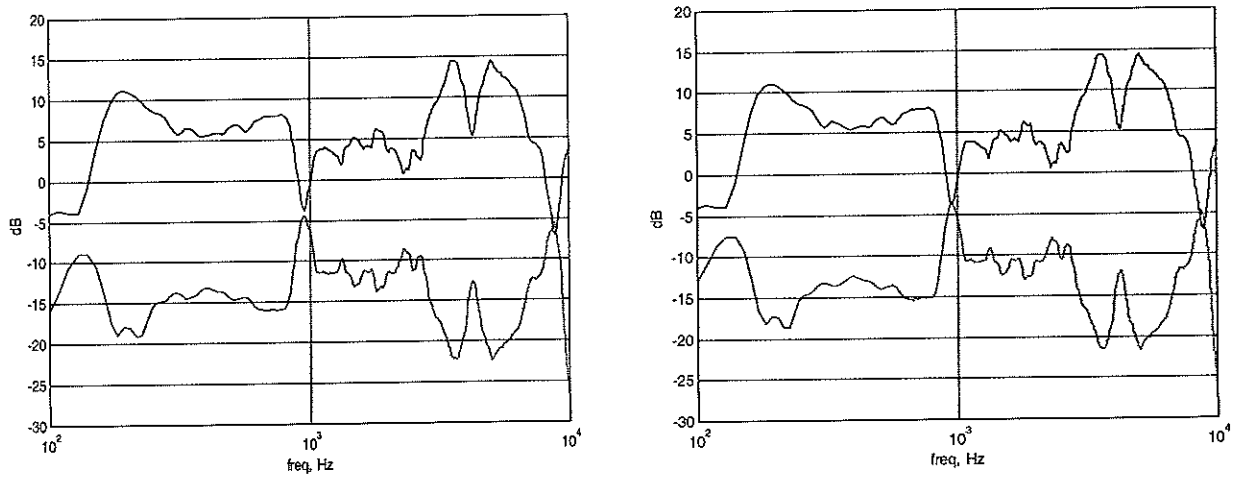


Figure 4.10: The speaker and the 1024 tap filters magnitude frequency. (Filters dotted) Inverse filter to the left and inverse minimum phase filter to the right.

By numerical convolution of the constructed filter and the sampled impulse response of the speaker we can see how good the filter will work theoretically. By measuring the response from the speaker fed with the filter we get the practical result. To be able to make real listening tests with music we have to sample some music and convolve the music signal with the filter. This can be done on-line in a signal processor, or off-line numerically (Matlab).

Figure 4.11 and 4.12 shows the inverse filters theoretical and practical result. As can be seen in the graphs the theoretical and practical result differs very little. The main difference comes from the noise in the practical measuring. This result shows that the speaker could be successfully estimated as a linear time invariant FIR system. Had the practical and theoretical results showed big difference, the speaker had been non linear and much harder to deal with.

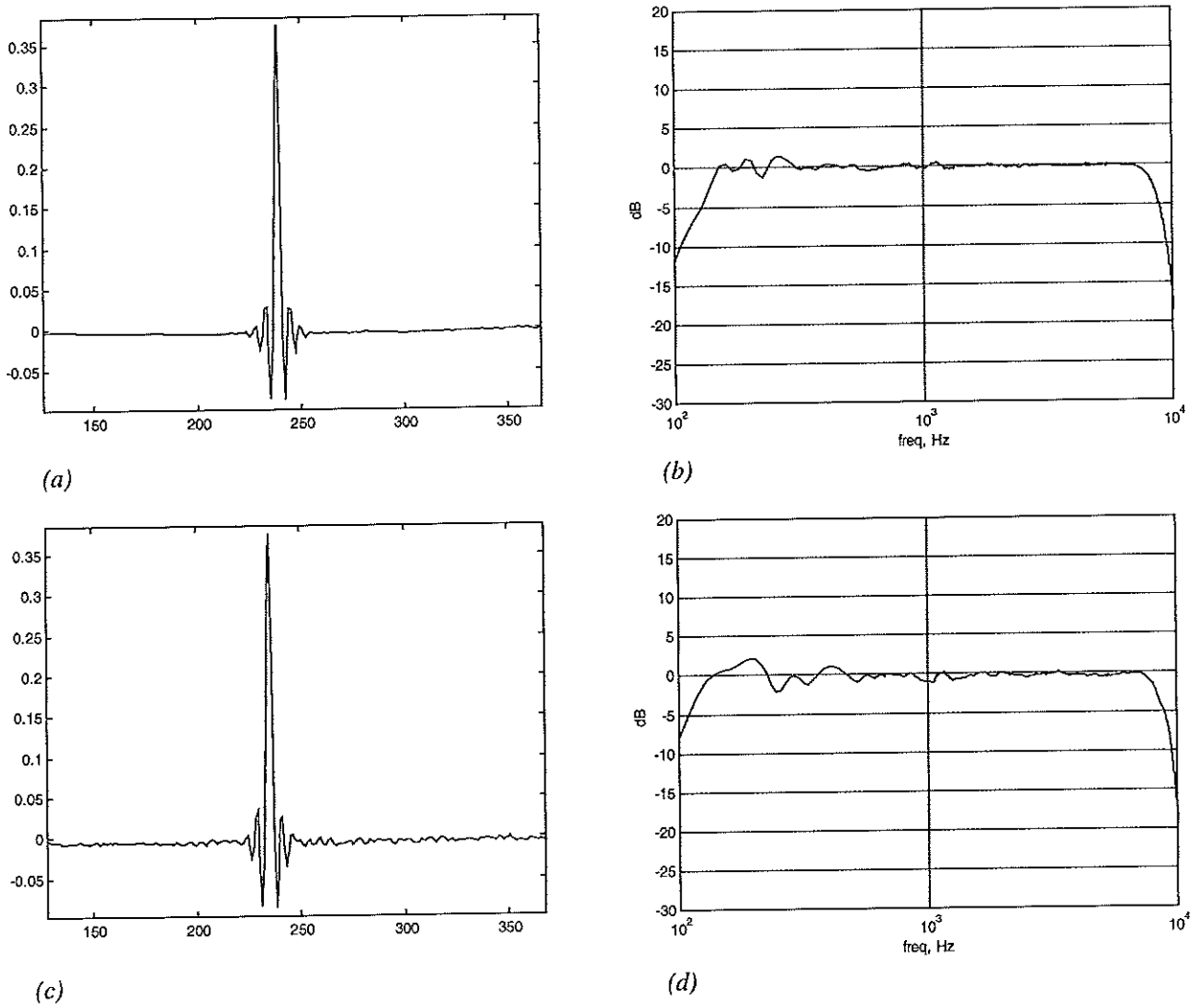
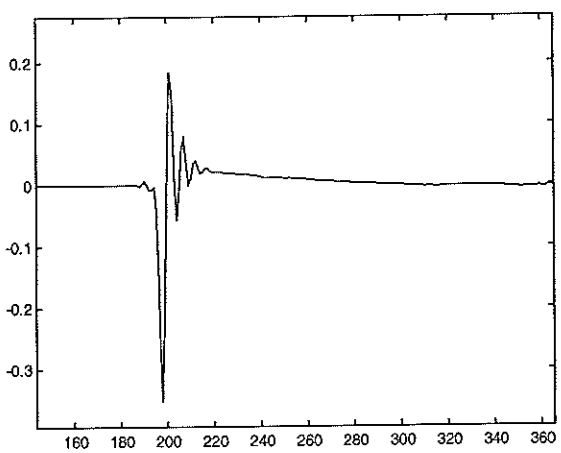
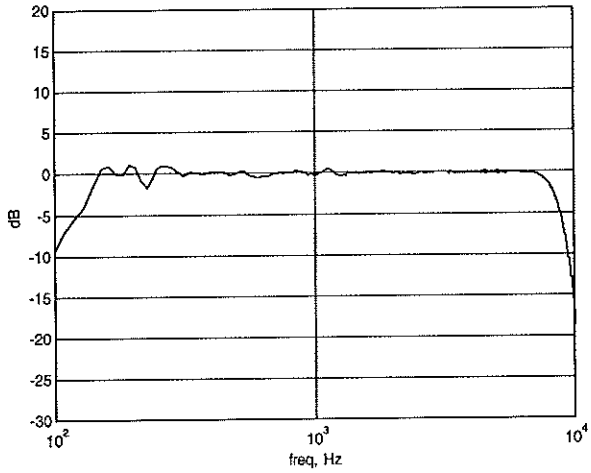


Figure 4.11: The impulse response, a and c, and the magnitude response, b and d of the small loudspeaker equalized with the 1024 tap Inverse filter. Simulated, (a) and (b), and measured, (c) and (d).

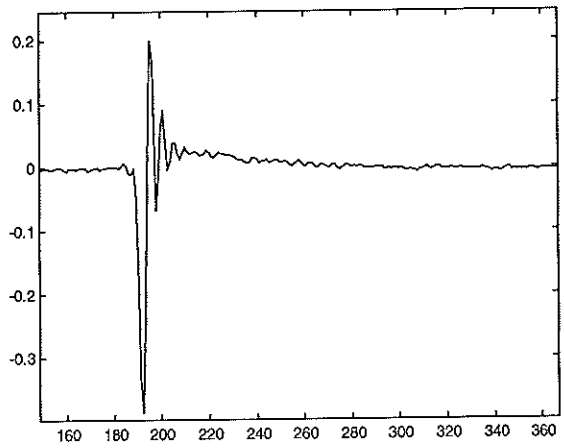
Also the results for the minimum phase inverse are very good. It is also very easy to see the ability of the minimum phase filter to correct the magnitude function just as well as the inverse filter (or even slightly better). But the response viewed in the time domain, shows big difference from the optimal bandpass respons we aim at. Still the response is much better than the original (see fig 2.1) which had much wider response with high amplitude.



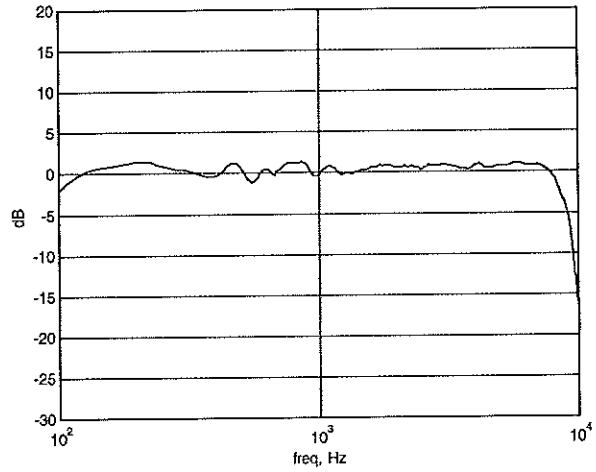
(a)



(b)

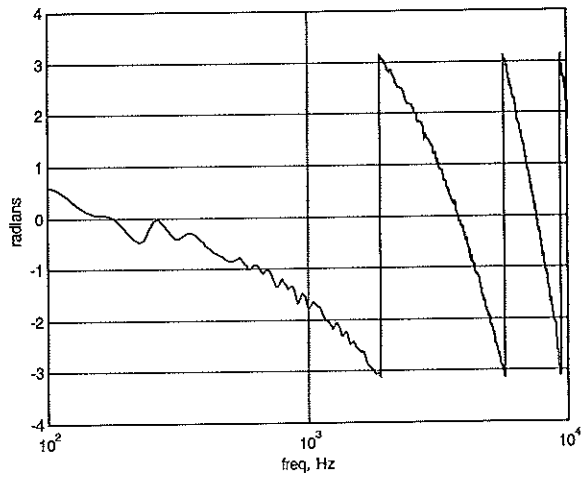


(c)



(d)

Figure 4.12: The impulse response, (a) and (c), and the magnitude response, (b) and (d) of the small loudspeaker equalized with the 1024 tap Inverse minimum phase filter. Simulated, (a) and (b), and measured, (c) and (d).



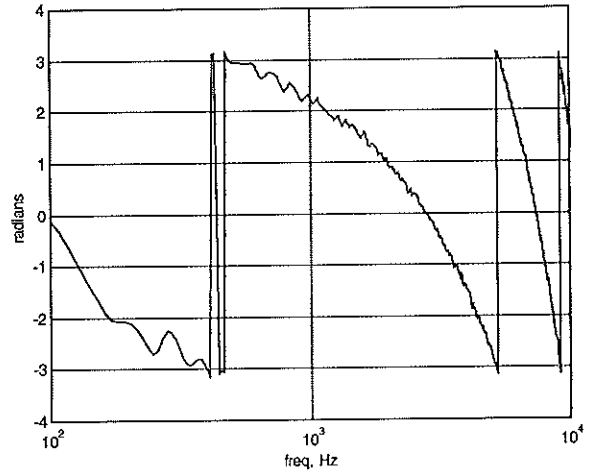


Figure 4.13: The phase response of the inverse filter (left) system and the inverse minimum phase filter (right)

The phase response from the two different equalized systems can bee seen in figure 4.13, even though it is hard to plot an accurate phase response it is fairly easy to see the bigger difference from the optimal phase response of the inverse minimum phase filter. Especially in the lower frequencies 100 to 500 Hz.

The phase response from the two different equalized systems can bee seen in figure 4.13, even though it is hard to plot an accurate phase response it is fairly easy to see the bigger difference from the optimal phase response of the inverse minimum phase filter. Especially in the lower frequencies 100 to 500 Hz.

1024 tap filters is rather long, even though it is not at all impossible to implement with a decent signal processor. A shorter filter is however always preferred to be able to use cheaper computer power. The results we got with the 256 tap filters show that it is too short to handle the job well. In figure 4.14 the response of the equalized speaker is shown with filters windowed with rectangular window. The length of the filter can easily be spotted in these time domain plots. The signal seems pretty good until it abruptly increases in amplitude.

In figure 4.15 the corresponding plots for the assymetric hanning windowed filters are shown. These lack the abrupt amplitude increase after the filters length. Instead they show a general higher amplitude level in time. However the magnitude functions plots in figure 4.16 shows that the hanning windowed filters have a slightly better magnitude function. They do have a dip around 1 kHz, but less ripple than the rectangular windowed filters.

It is also shown that the inverse minimum phase filter give better results. The inverse filter lack some of the bass that is preset in the inverse minimum phase function and in our target function.

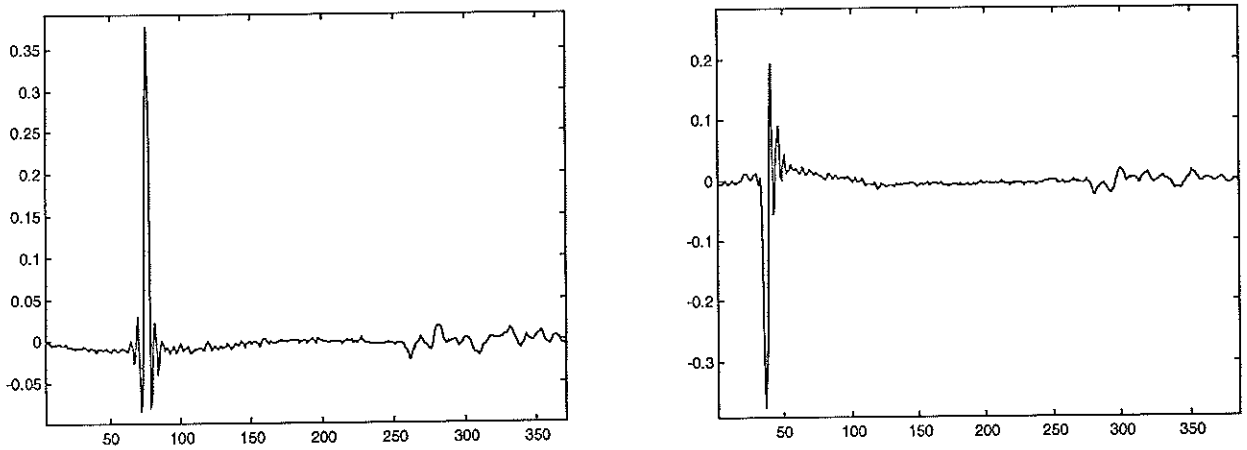


Figure 4.14 Response of 256 tap filters, inverse filter to the left and inverse minimum phase filter to the right.

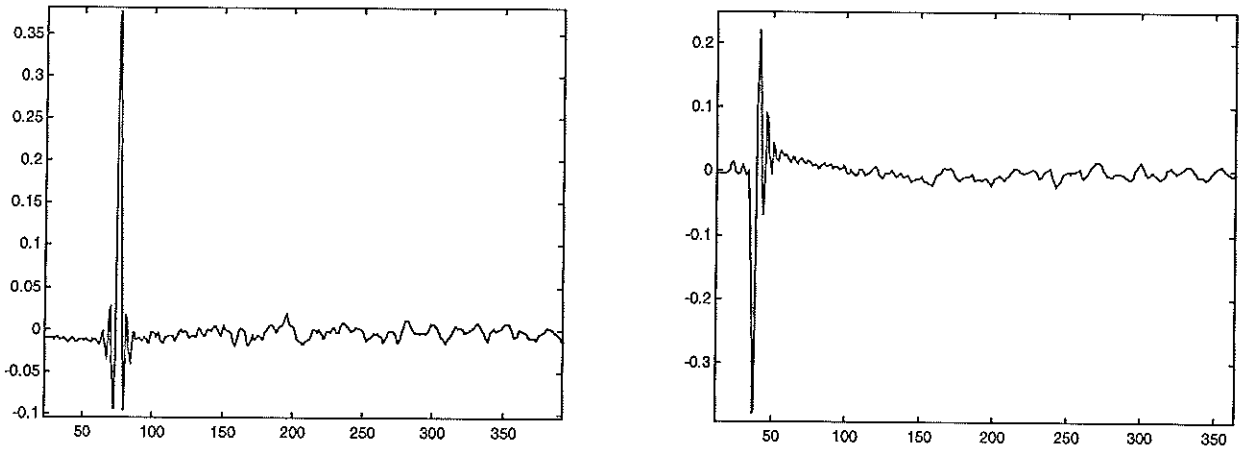


Figure 4.15: Response of 256 tap hanning windowed filters, inverse filter to the left and inverse minimum phase filter to the right.

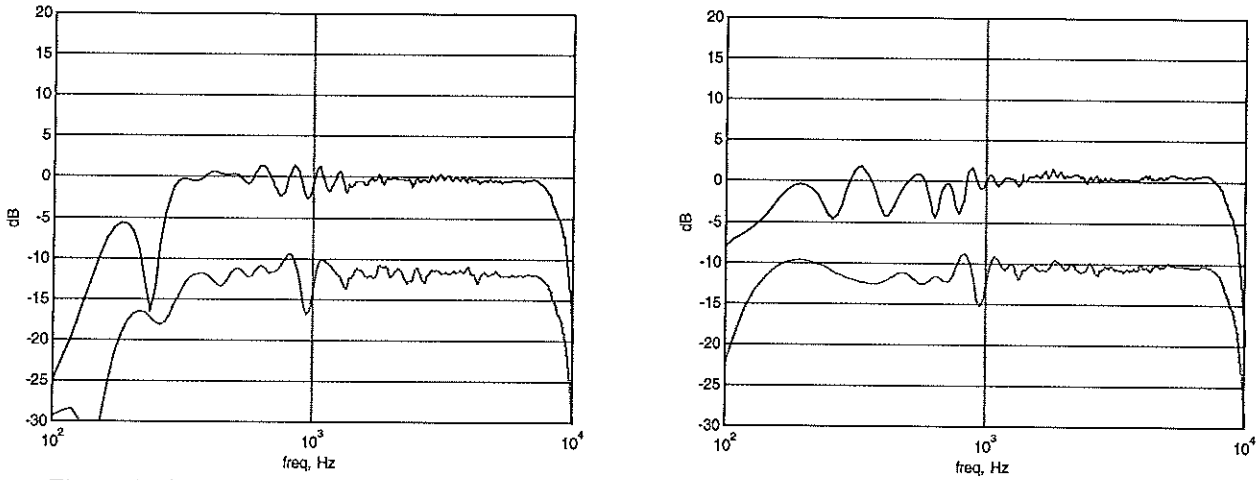


Figure 4.16 Magnitude function of the equalized speaker. 256 tap filters used and inverse filters to the left and inverse minimum phase filters to the right. Hanning windowed filters dotted.

For the large speaker, that have not been tested as thorough as the small one, we here show and evaluate the results of using 1024 tap filters designed as described in chapter 4.1 and 4.2. The constructed filters is shown in figure 4.17

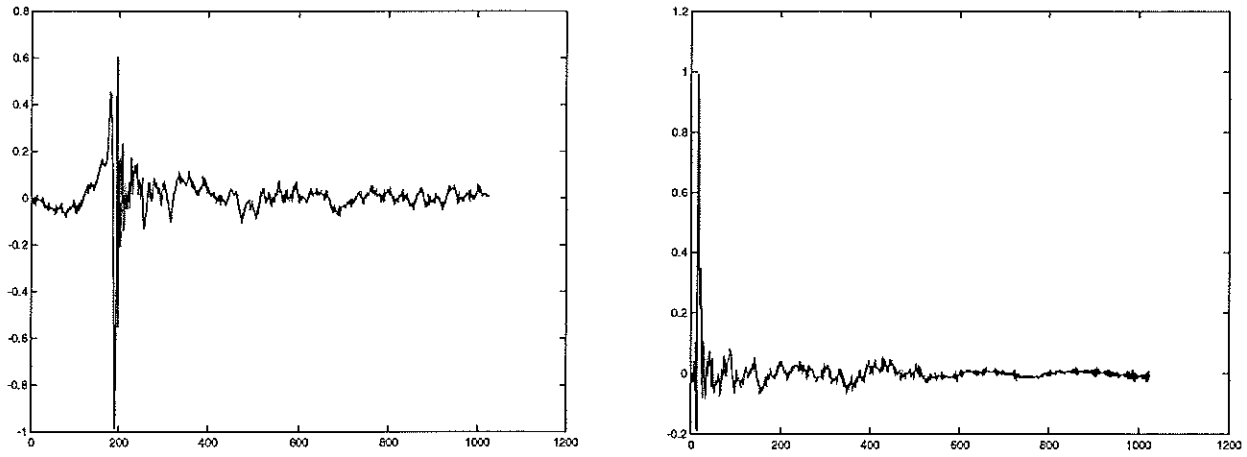


Figure 4.17: The 1024 tap filters constructed to the large speaker. Inverse filter to the left and inverse minimum phase filter to the right.

We can see that the inverse filter have a rather high amplitude in the end. This is because of the more complex impulse response of the big speaker (compared to the small one). The use of multiple drivers positioned on different places and cross-over networks introduce delays and phase distortion between the different frequencies that gives the speaker a more complex response and "less minimum phase structure". Since the inverse minimum phase filter only corrects the magnitude function and "assume" that the phase response is minimum phase, we expect a better filter from this method given the same inverse filter have a rather high amplitude in the end. This is because of the more complex length (1024 tap).

It is clearly seen in figure 4.17 that the inverse minimum phase filters amplitude has declined more than the inverse filter.

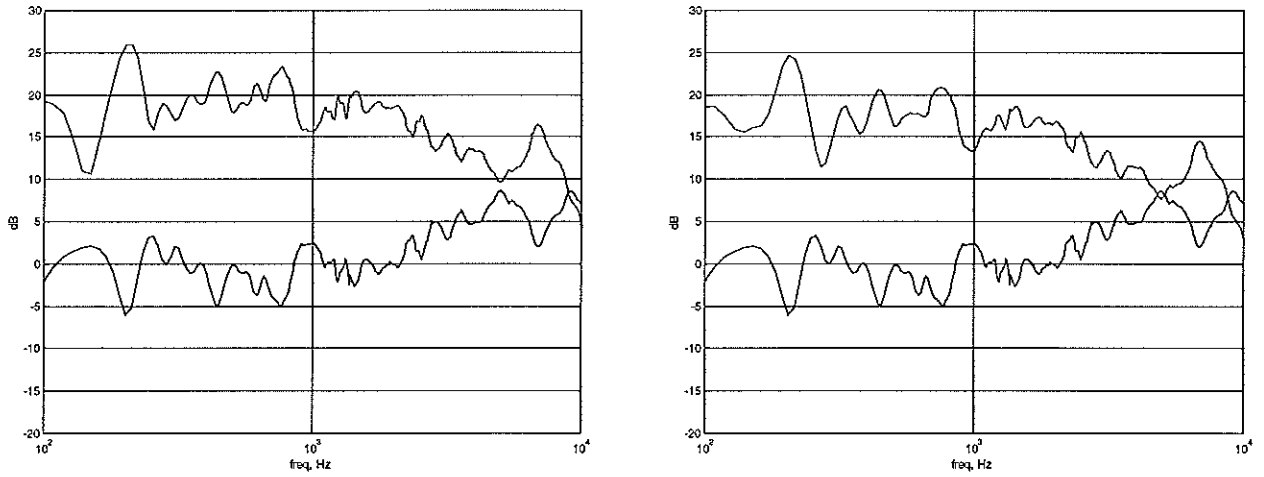


Figure 4.18 The magnitude function of the speaker and the two filters (dotted). Inverse filter to the left and inverse minimum phase filter to the right.

In figure 4.18 the magnitude function of the two filters is plotted together with the magnitudefunction of the speaker. Both filters seems to do well, but the inverse filter have slightly better magnitue function. Why is unclear.

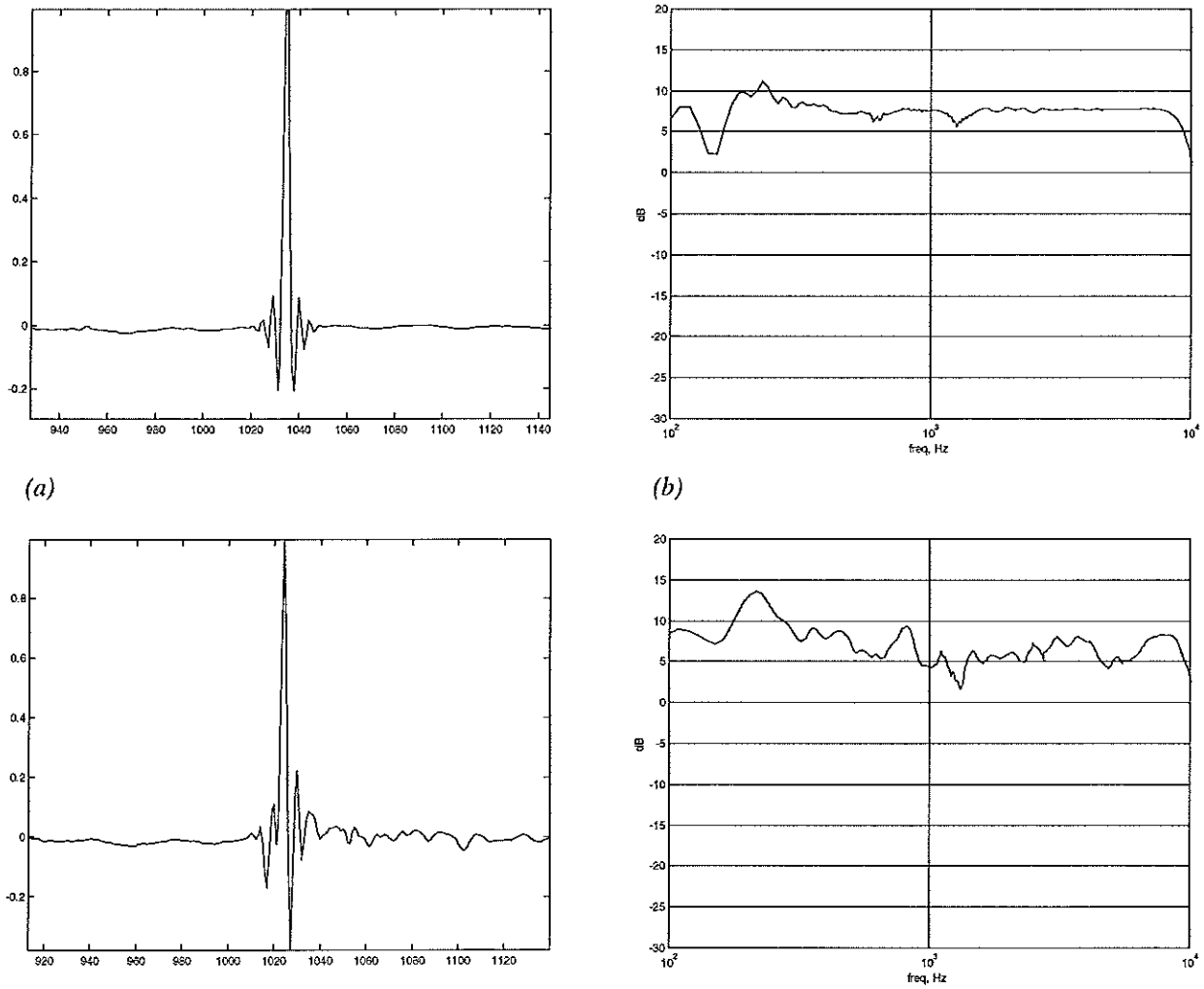


Figure 4.19: The impulse response, (a) and (c), and the magnitude response, (b) and (d) of the large loudspeaker equalized with the 1024 tap Inverse filter. Simulated, (a) and (b), and measured, (c) and (d).

In figure 4.19 the simulated and practical measured impulse reponse and magnitude frequency of the equalized system is plotted. The simulated and measured response is rather different and this is probably caused by that the speaker does not behave as we assume, i.e. the speaker is not well estimated with a linear system. This combined with the to short filter is our explanation of the poor result.

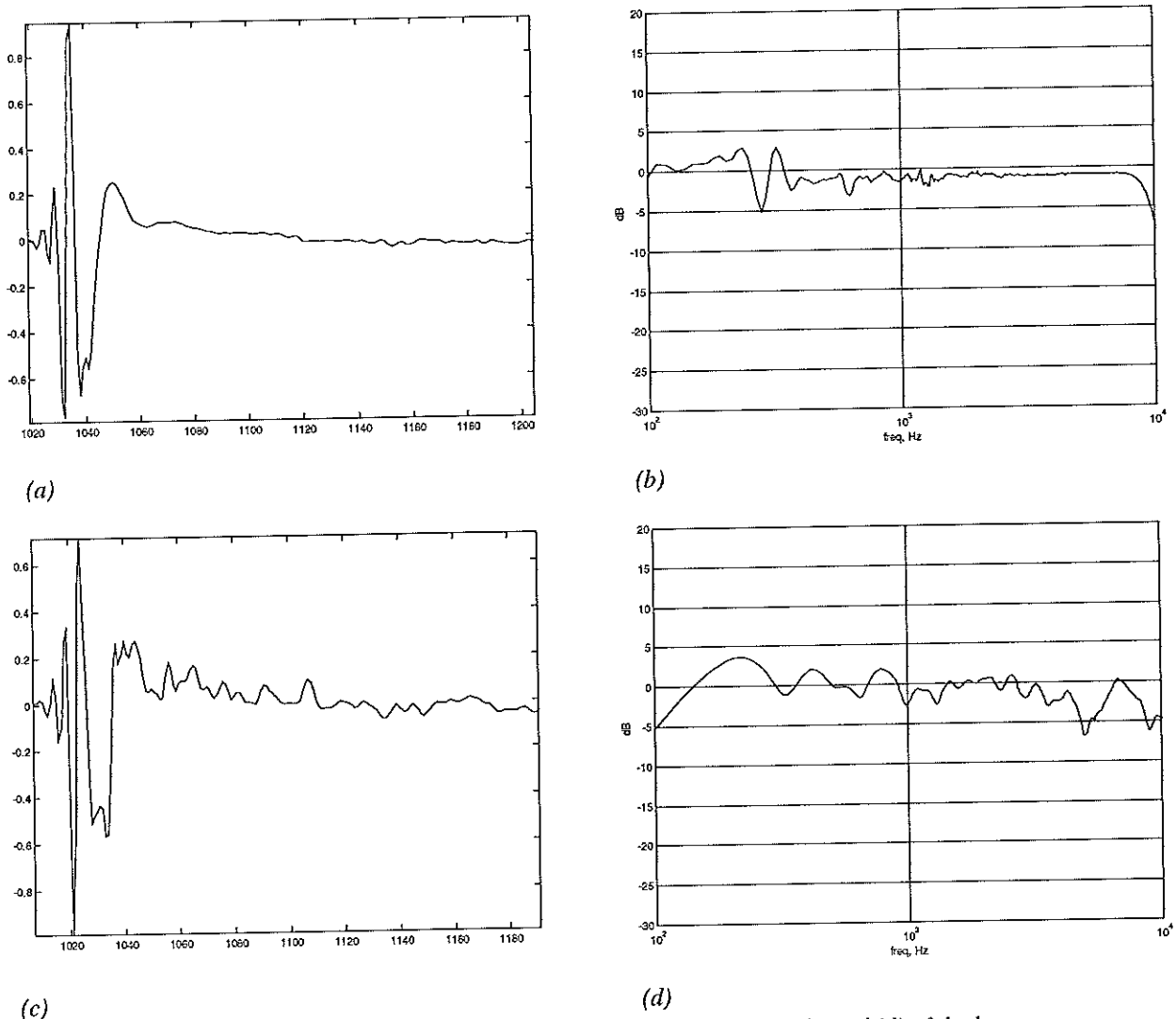


Figure 4.20 The impulse response, (a) and (c), and the magnitude response, (b) and (d) of the large loudspeaker equalized with the 1024 tap Inverse minimum phase filter. Simulated, (a) and (b), and measured, (c) and (d).

Figure 4.20 shows the results from the inverse minimum phase filter shows some strange irregular peaks and dips. Again this is probably because non linear properties of the speaker.

4.6 CONCLUSIONS OF FREQUENCY INVERSE METHODS

The frequency inversion methods were succesfully used to equalize the speakers. The results of equalizing the small speaker was really good. The measurements show a great improvement in the magnitude response and for the inverse filter a nearly perfect impulse response as expected. The difference between the inverse minimum phase filter and the inverse filter with 1024 taps could not be heard in our listening condition. The shorter filters, 256 taps, did not sound equally good. We liked the shorter inverse minimum phase filter better than the inverse filter, because of its better low frequency characteristics.

The improvement of the large speaker did not sound or measure as good as the small speaker, probably because it have non-linear properties.

5. Least Squares

5.1 Least Squares Background

The Least Squares method has achieved widespread applications and is applied on common LTI systems. We will develop the concepts of least squares minimization with a view to our application of loudspeaker equalization. The majority of the work presented in this chapter is modified from the explanation provided in [7].

The intention with the Least Square design is to design a filter that is optimally close to the ideal result in some sense. The input signal to the filter is to be filtered by a linear filter with impulse response $f(n)$ and produce an output as close as possible to the desired one. The performance of the filter is observed through the error $e(n)$.

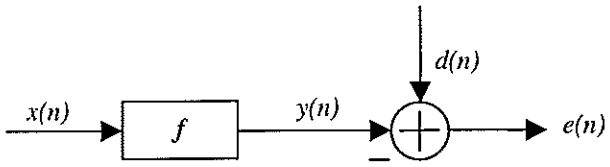


Figure 5.1: The LS filter model.

For random input signals we can define an error-function $J=E(e^2(n))$, where E is the ensemble expectation value. The goal will be to minimize J, the mean squared error, over all n.

Now referring to figure 5.1,

$$e(n)=d(n)-\mathbf{f}_n\cdot\mathbf{x}_n \tag{5.1}$$

where $\mathbf{f}_n$ is the filter vector and $\mathbf{x}_n$ is the input vector of samples.

To minimize the error we differentiate the error-function J with respect to each coefficient f_i and equate the results to zero;

$$\frac{\delta J}{\delta f_i}=\frac{\delta E\{e^2(n)\}}{\delta f_i}=2\cdot E\left\{e(n)\frac{\delta e(n)}{\delta f_i}\right\} \tag{5.2}$$

And since

$$\frac{\delta e(n)}{\delta f_i}=\frac{\delta\left(d(n)-\sum_j f(j)\cdot x(n-j)\right)}{\delta f_i}=-x(n-i) \tag{5.3}$$

This gives us

$$\frac{\delta J}{\delta f_i}=2\cdot\left(\sum_i R(n-i,n-j)\cdot f(i)-g(n-j,n)\right) \tag{5.4}$$

Here R is the auto-correlation matrix of the input signal and g is the cross-correlation vector between the input $x(n)$ and the desired signal $d(n)$, defined as follows:

Autocorrelation:

$$R(n-i,n-j) = E\{x(n-i), x(n-j)\} \tag{5.5}$$

Crosscorrelation:

$$g(n-j,n) = E\{x(n-j)\cdot d(n)\} \tag{5.6}$$

This gives us a set of linear algebraic equations known as the *normal equations*, which can be written as

$$\sum_i R(n-i,n-j)f(i) = g(n-j,n) \tag{5.7}$$

Here we can set some constraints on the equations based on the physical properties of the system; For this case the signals involved are deterministic, i.e. the speakers impulse response, and this gives us a slightly simpler autocorrelation and crosscorrelation function. The filter is also bounded to be causal and of finite length, which means the solution cannot be obtained by simple spectral division, and will generally only allow us to find an approximate solution to the filter. The normal equations becomes

$$\sum_{i=0}^{L-1} r(j-i)f(i) = g(j) \quad 0 \le j \le L-1 \tag{5.8}$$

where L is the length of the filter. The solution of these equations gives us the optimal filter, or the *least squares filter*, and can be simpler written in vector form as

$$\mathbf{R}\cdot\mathbf{f} = \mathbf{g} \tag{5.9}$$

or better

$$\mathbf{f} = \mathbf{R}^{-1}\cdot\mathbf{g} \tag{5.10}$$

where R is the autocorrelation matrix of the input signal, g is the crosscorrelation vector and f is the filter vector.

Inverse filtering

Concentrating on the less general case of *inverse filtering*, we get a slightly different model and equations:

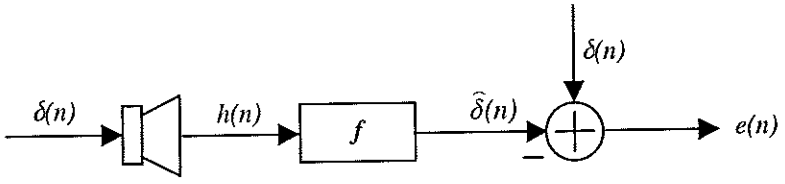


Figure 5.2: The LS filter model for inverse filtering.

Suppose that the loudspeaker in figure above is exited with unit impulse $\delta(n)$, then the filter will have an input that is the impulse response of the loudspeaker, and the desired signal will be the unit impulse.

We can simplify the autocorrelation to

$$r(l) = \sum_{n=-\infty}^{\infty} h(n) \cdot h(n+l) \tag{5.11}$$

And the crosscorrelation becomes

$$g(l) = \sum_{n=-\infty}^{\infty} \delta(n) \cdot h(n-l) = h(-l) \tag{5.12}$$

The finite causal filter will give the normal equations of the form

$$\mathbf{f} = \mathbf{R}^{-1} \cdot \mathbf{g} \tag{5.13}$$

In this case the autocorrelation matrix **R** will be built upon the impulse response and the crosscorrelation **g** simply becomes the impulse response shifted backwards into the first elements of the correlation matrix.

Using delay

The use of delay to compensate for the time the signal takes through the system will greatly improve the inversion of this kind of non-minimum phase system. The modified least squares model will be to delay the desired signal *d(n)* with an arbitrary delay *m*. The model will look like:

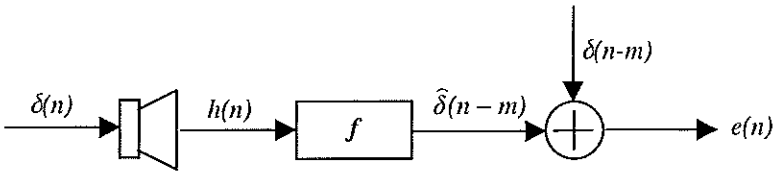


Figure 5.3: The LS filter model for inverse filtering with introduced delay.

The autocorrelation r(l) will be unchanged, but the crosscorrelation will look like

$$g(l) = \sum_{n=-\infty}^{\infty} \delta(n-m) \cdot h(n-l) = h(m-l), \tag{5.14}$$

which means it has the speaker impulse response shifted the necessary amount of delay into the vector. Finally, the normal equations become

$$\mathbf{R} \cdot \mathbf{f} = \mathbf{g} = [h(m) \quad h(m-1) \quad \ldots\ldots \quad h(0) \quad 0 \quad 0 \quad \ldots\ldots \quad 0] \tag{5.15}$$

The delay in the system is in this particular case somewhat 2ms, and the best way to determine the delay is to calculate filters with different delays and chose the one with the smallest error.

This type of delay is no problem in a home audio playback system, but a long delay could be a problem if used for *live* performance situations.

Computation

There are two stages in the computation of the filter $\mathbf{f}_{opt}$, the computation of the autocorrelation of the input samples and the solution of the normal equations, which contains a computationally heavy matrix inversion. The computational load of the inversion of the autocorrelation matrix is with the Levinson-Durbin algorithm proportional to L^2, with L being the length of the impulse

response. For off-line filter design an adequate length of the impulse response could typically be a thousand samples, and the autocorrelation matrix will then be a 1000 * 1000 Toeplitz matrix. The complexity of this is far too much for any real time system to update and invert on a sample by sample basis and is for that reason suited for offline processing. To design a 1024 tap least squares filter on a Pentium 200Mhz would take an hour of processing.

The finite least squares inverse filter will be minimum phase, independent of the phase structure of the input. Consequently, the inversion is most effective when the input sequence is minimum phase.

A nonminimum phase system can not be completely deconvolved using a stable causal inverse filter. Such a filter can effectively flatten the spectrum, but will leave a residual phase distortion, the all pass component. But by introducing delay in the system we can produce a flat spectrum and a *linear* phase curve.

Analysis of the least squares algorithm

The least squares as it is presented here has some obvious advantages over the inversion algorithm presented earlier:

A feedback loop is integrated to the technique, which compares the input and output of the system and drives the coefficients in such a direction as to minimize the resultant error.

A performance measure may be calculated in the form of the least squares error. This may be used to judge (and hopefully improve) the performance of the algorithm.

The derivation of the filter is based on sound mathematical theory, and is open to systematic development of the algorithm using recognized mathematical and scientific techniques.

Some obvious drawbacks though are evident:

The computational complexity of the algorithm shown is very high.

Whilst the algorithm here is reasonably simple, the filter is based in the time domain and as such it is very difficult to se how this system operates in the frequency domain, which is where we would like to analyze our speaker characteristic.

5.2 *RESULTS*

In figure 5.4 to 5.6 we see some of the filters that has been derived for the small speaker, the simulated results and measured system responses from the whole setup.

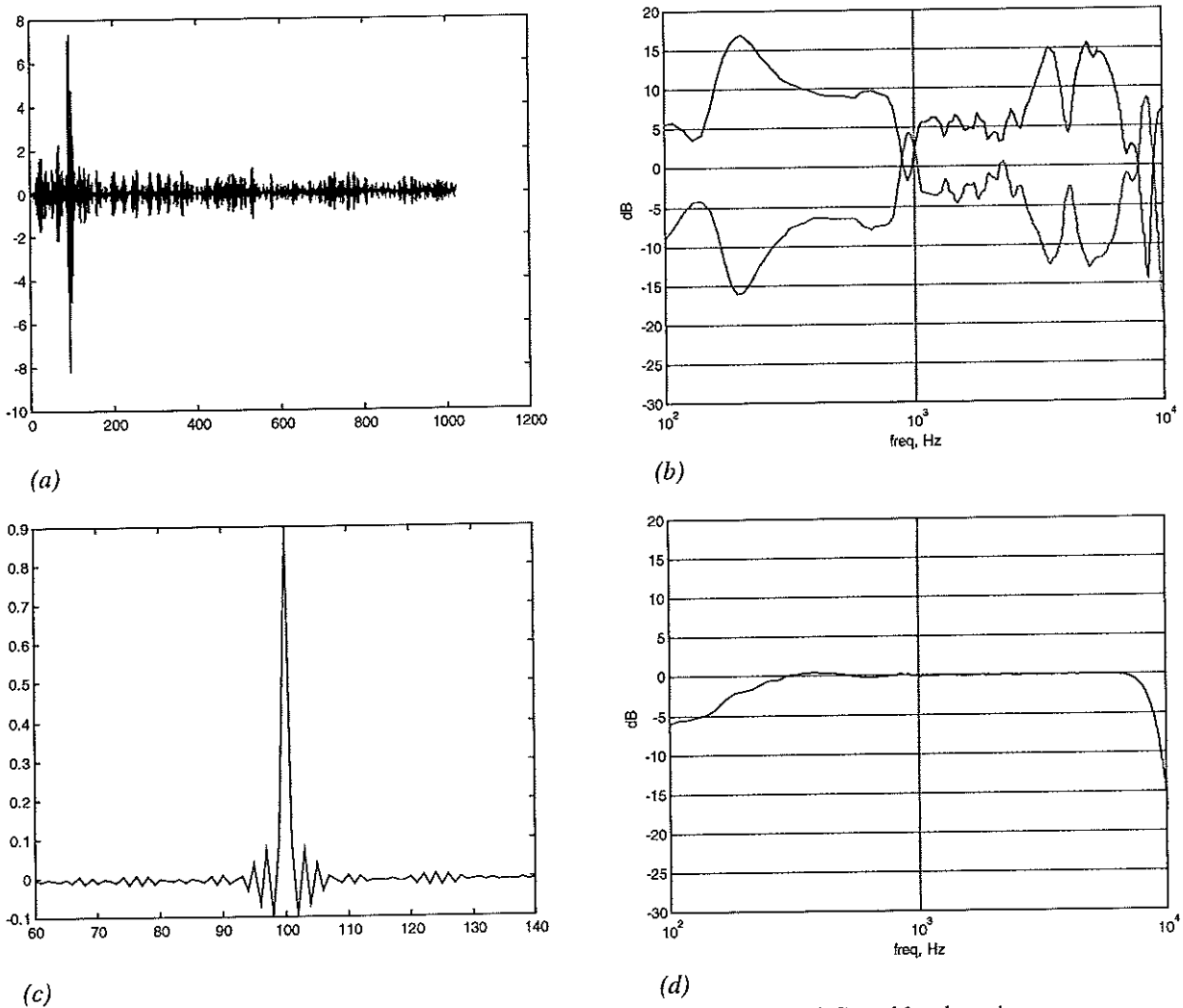
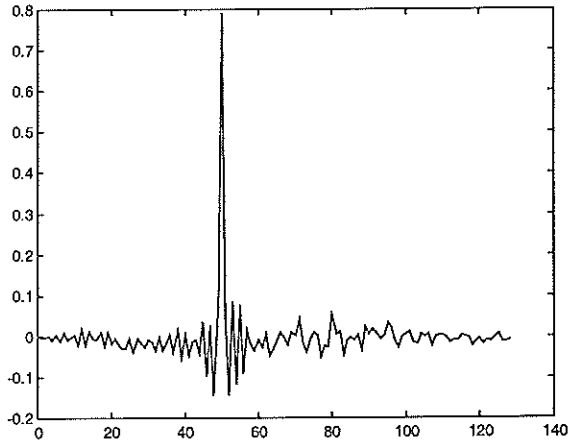
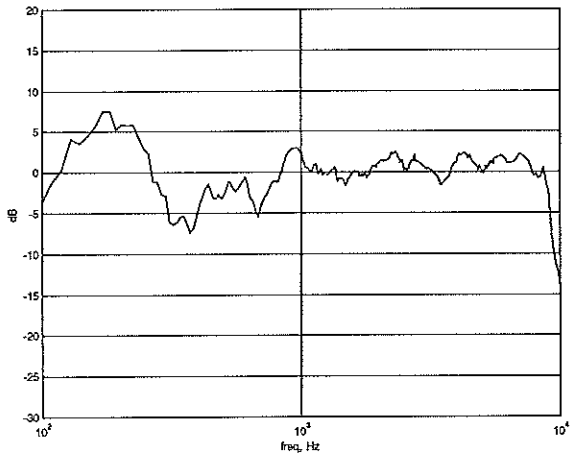


Figure 5.4: (a) 1024 tap filter, (b) simulated frequency response for filter (solid) and loudspeaker response (dotted), (c) simulated impulse response, (d) simulated frequency response.

As we see in figure 5.4 we have a long filter that in the simulations corrects the amplitude perfectly, except for some low frequency level. The simulated impulse response is very close to a perfect low-pass filter response. The system should definitely be able to improve the speaker performance under normal listening conditions.

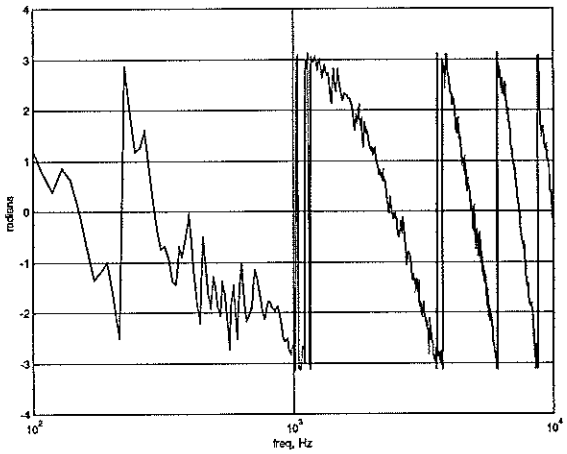


(a)



(b)

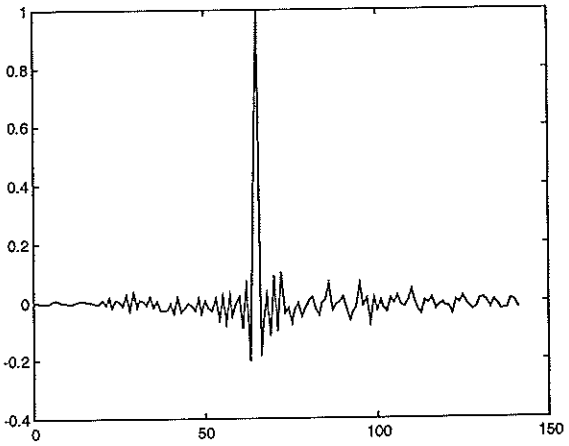
The measured response also has some obvious deviations from flat magnitude for the low frequencies, and some difficulties can also be seen in the phase response. Probably a big part of the deviatons is due to complicated measurement conditions. The responses are clearly improved compared with the original though.



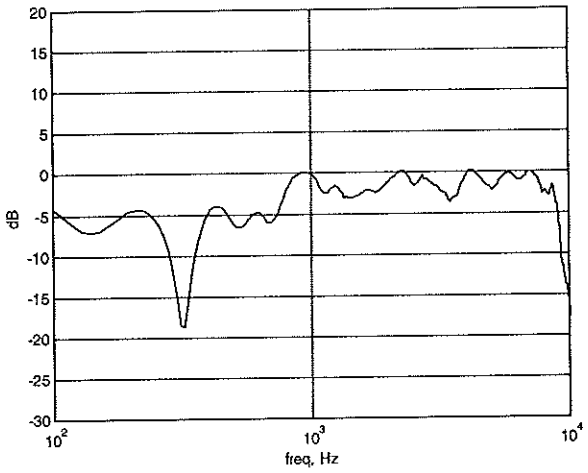
(c)

Figure 5.5: The 1024 tap filter: (a) measured impulse response, (b) measured frequency response, (c) measured phase response.

In figure 5.6 below we see the small speakers 256 tap response.

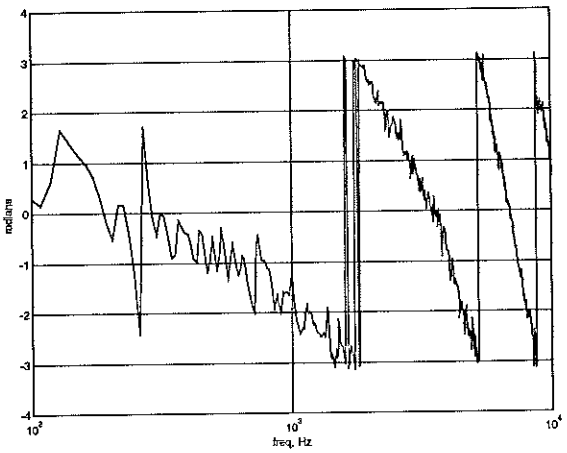


(a)



(b)

As we see, the 256 tap filter in this case equalizes the speaker as good as the 1024 tap filter do, and the magnitude is mostly within 5dB. The sharp dip in the 256 tap filter is a result of reflections in the measurement setup. A good impulse response is an obvious result.



(c)

Figure 5.6: The 256 tap filter for the small speaker: (a) measured impulse response, (b) measured frequency response(truncated filter dotted), (c) measured phase response.

In figure 5.7 below we have the measurements of the big speaker.

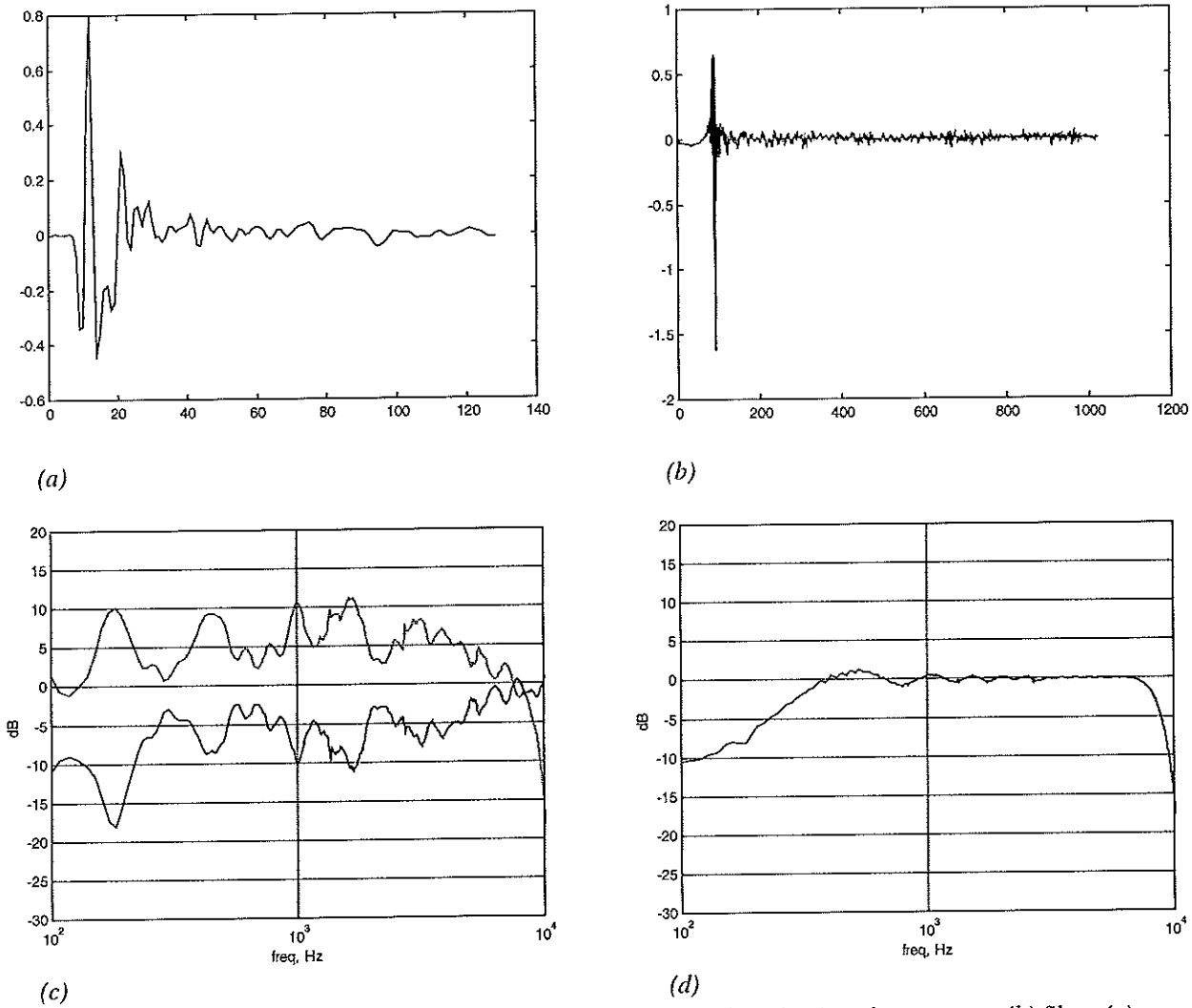
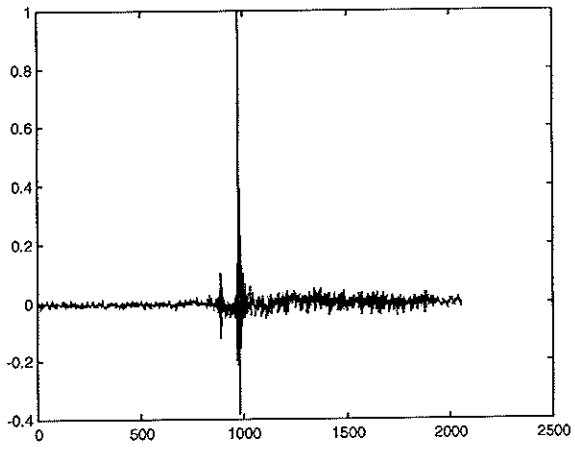
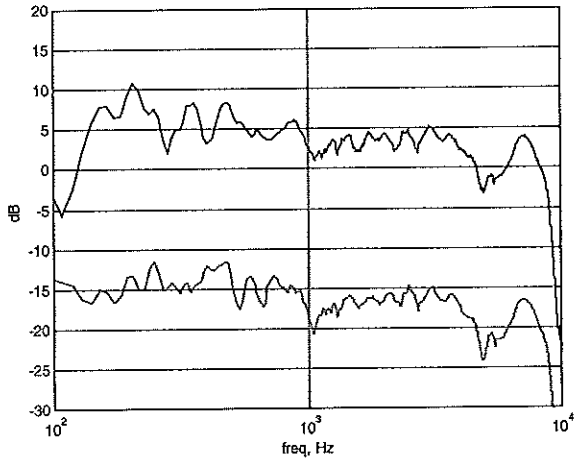


Figure 5.7: Big speaker measurements with 1024 tap filter: (a) loudspeaker impulse response, (b) filter, (c) filter and loudspeaker frequency responses, (d)simulated system frequency response

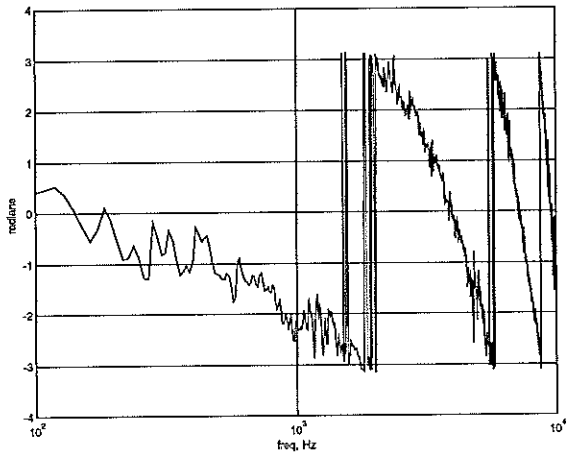


(a)

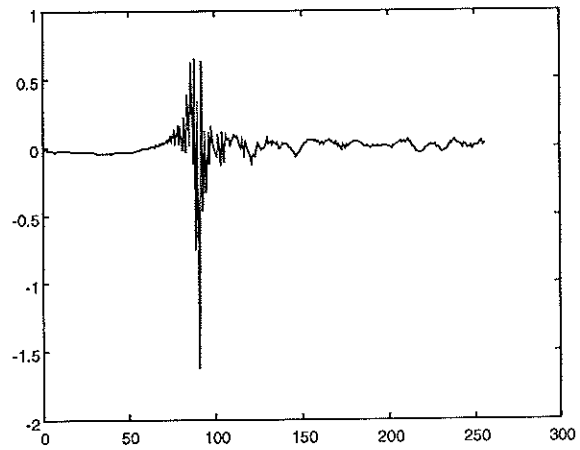


(b)

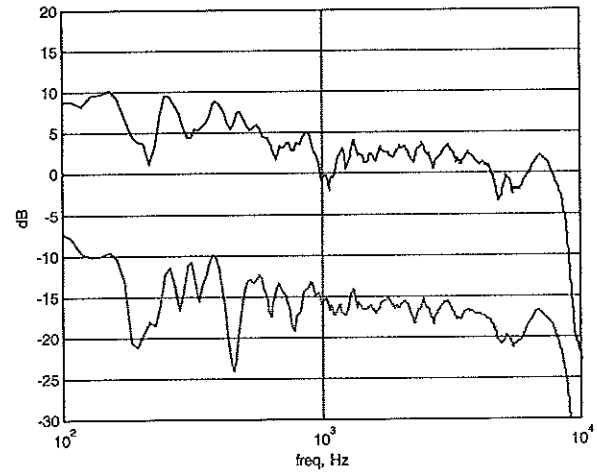
The responses in figure 5.8 for the both filter lengths are clearly improvements of the original response. The magnitude response is equalized to within half the previous deviation on the dB scale. The two filter lengths are quite similar in magnitude response, but the impulse response is better for the longer filter.



(c)



(d)



(e)

Figure 5.8: Big speaker: 1024 tap filter: (a) system impulse response, (b) system frequency response (dotted curve truncated) (c) system phase response. 256 tap filter: (d) system impulse response, (e) system frequency response (dotted curve truncated).

5.3 *Conclusions of the Least Squares method*

We have seen that an inverse could be calculated off-line using the time-domain based Least Squares method, also shown in Appendix D. The method is quite heavy computationally due to the inversion of the auto-correlation matrix and is in this case only possible off-line. The use of delay is important to be able to equalize a non-minimum phase system, and the delay time is decided by "trial and error". It is also quite complicated to se how the method works in the frequency-domain due to its time-domain base. We can clearly improve the loudspeaker response with this method, though we get some obvious problems with the bigger speaker and its non-minimum phase properties. The method seems to not be as accurate as the *inverse method*, though.

6. LMS

6.1 LMS BACKGROUND

Since the transfer characteristics of a loudspeaker will change very little with time, the ability of the filter to adapt to changes in the characteristics might seem somewhat unnecessary. The previous systems, in chapter 4 and 5, with predetermined filter coefficients though have some shortcomings. They require both very careful measurements of the speaker characteristics, and we also have to do the calculations of the coefficient from these measurements. The labor cost for this could seem somewhat high and also a lack of versatility could call for an adaptive solution.

With an adaptive system we could compensate not only for changes in the speaker itself, but also for changes in the sound path between speaker and the microphone. For example, after installing the sound system in a completely new environment the system can be made to automatically adapt itself to an optimal response.

Therefor the design of such a *smart* loudspeaker seems to be a tempting idea for designers recent years. Some work, mainly research studies, has been done on this topic, such as[3].

A very important issue here, however, will be to chose the right reference point, the point in which we are supposed to listen to the sound and where we place our microphone. It could be a single spot or maybe an average over several spots in the listening area. Averaging is quite a hard topic in this case since we are both working in real-time and the time domain. We can only try to choose the right reference point, and try choosing it away from any obvious flaws that appear in just one spot.

An adaptive system is not necessarily bounded to the restriction of linear systems as with Least Squares and should be able to produce a result that is closer to flat amplitude and linear phase response compared with the original.

Various computational methods exist for getting closer to the optimal filter. They can be carried out on a sample by sample basis or by analyzing blocks of data, and either in the time or frequency domains. Also any of the usual digital filter structures can be implemented in adaptive form, but for ease of implementation and analyzing we will use a simple FIR filter and implement the well-known *LMS-algorithm [3][7][8].*

Adaptive LMS algorithms

The real-time adaptive system will look like the setup below:

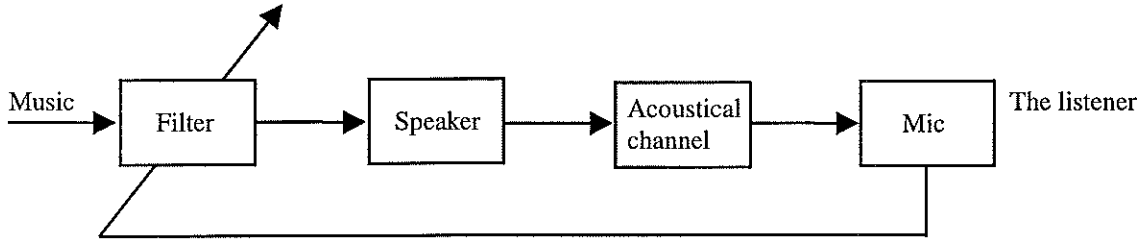


Figure 6.1: The adaptive filter arrangement as implemented in a real environment.

While this is an illustrative model of the system we shall redefine the model somewhat to be able to give a more theoretical background of the adaptive algorithm.

In the design of Least Squares FIR of the previous chapter the inverse filter was designed on the basis of a known speaker character $h(n)$. The filter coefficients were determined off-line, and once the filter is determined they remain fixed. Adaptive filters working in real-time on the other

hand are time-varying, and they have filter coefficients whose values are adjusted for each incoming sample to minimize the measure of an error-signal, *e(n)*. Thus, the algorithm will *adapt* the coefficients towards a filter that generates the least possible error. The main benefit of this technique is that such a filter is able to change along with the statistical properties of the channel or the signal.

We will for our model see the path through the speaker, acoustical channel and microphone as one single channel and call its impulse response *h(n)*. This is the system we want to equalize.

We redraw the system and introduce some more definitions:

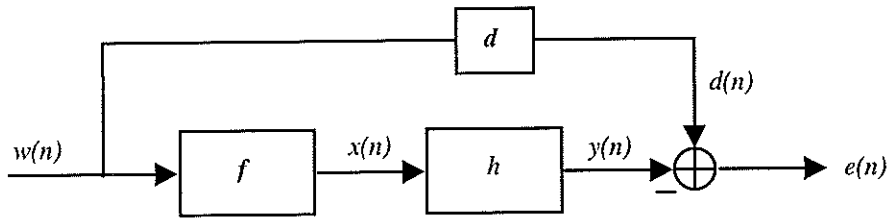


Figure 6.2: The equalization model of the LMS system.

Here we have *w(n)* as the input to the filter, *x(n)* is the input signal filtered through the inverse filter, and *y(n)* is the estimate of the input signal. *d(n)* is the desired signal, in this case the delayed input signal filtered through the target function. If the input is white noise, this signal should have the desired band-pass characteristics suitable for the speaker.

It should be obvious how the model is supposed to work. We will have the input filtered through the filter *f* and again through the speaker system *h*, and the output *y(n)* will be compared with the delayed desired signal *d(n)*. The error-signal *e(n)* will be used by the algorithm to update the filter coefficients. The filter thus consists of two parts, the FIR filter that processes the stream of data, and the algorithm that adjust the coefficients.

During adaptation this system is preferably excited by a broadband signal *w(n)*, such as white noise, or could even be exited with a music signal. If such a music signal is "white enough" we will have a fairly even adaptation rate at all frequencies.

Analyzing the LMS algorithm

Since the adaptive filter is based on the Least Squares algorithm, the optimal filter coefficients could of course be calculated directly from

$$\mathbf{f} = \mathbf{R}^{-1} \cdot \mathbf{g} \ , \tag{6.1}$$

where we still have **R** as the input auto-correlation matrix and **g** as the crosscorrelation vector between the desired signal and the input. But since this is a too big computational load to do in real time, especially if the filter length is large, we have to consider an *iterative* solution. The goal is still to minimize the error-function,

$$J = E\{e(n)^2\}, \tag{6.2}$$

the least squares minimization of the error *e(n)*. This could be accomplished by starting with an initial guess of $\mathbf{f}_0$, and from there, with every iteration, getting better estimates of $\mathbf{f}_n$. The iterative approach could even be useful in obtaining the *fixed* least squares solution since the matrix **R** could be singular.

The iterative update of the filter coefficients has the form:

$$\mathbf{f}_{n+1} = \mathbf{f}_n + \frac{\alpha}{2} \cdot \mathbf{p}_n \ , \tag{6.3}$$

where $\mathbf{p}_n$ is a vector which defines the search direction. As we may have noted, the quadratic function J can be thought of as an L-dimensional bowl-shaped surface with one single minimum describing the optimal value of $\mathbf{f}$. And since J_n is a function of $\mathbf{f}_n$, $J_{\min} = J(\mathbf{f}_{opt})$, the interpretation of $\mathbf{p}_n$ is the negative gradient of J, $\nabla J_n = \delta J / \delta \mathbf{f}_n$. At each iteration we move in the direction opposite to the gradient of J, and by a distant proportional to magnitude that gradient, hence the name of the method, *steepest descent*.

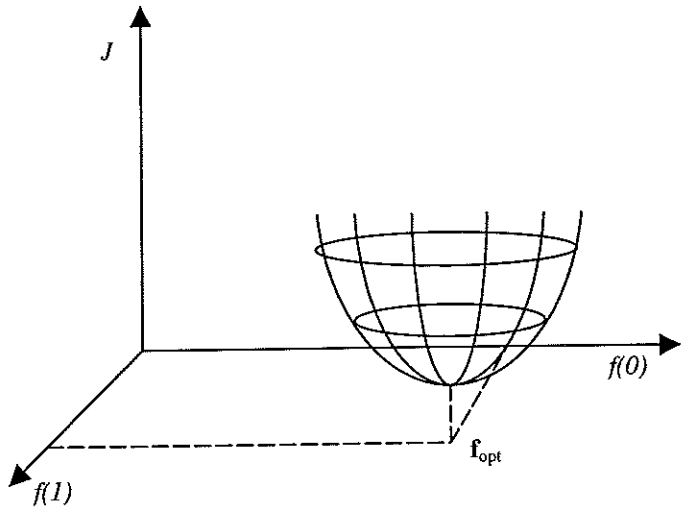


Figure 6.3: The quadratic performance surface of errror-function J in the two-dimensional case, with the two filter coefficients f(0) and f(1).

The theory , see [7], tells us that

$$\nabla J_n = 2 \cdot \mathbf{R} \cdot \mathbf{f}_n - \mathbf{g} \tag{6.4}$$

Our iteration formula becomes

$$\mathbf{f}_{n+1} = \mathbf{f}_n + \frac{\alpha}{2} \cdot (-\nabla J_n) = \mathbf{f}_n - \alpha(\mathbf{R} \cdot \mathbf{f}_n - \mathbf{g}) \tag{6.5}$$

The equation above works only for a stationary input, that is when **R** and **g** don't change with time. We could calculate the values of **R** and **g** and use this equation as it is, but we still want to find the optimal solution when the parameters of the input signal change with time. The equation

$$\mathbf{f}_{n+1} = \mathbf{f}_n - \alpha(\mathbf{R}_n \cdot \mathbf{f}_n - \mathbf{g}_n) \tag{6.6}$$

would do the job, but the computational load is still very high. We could instead replace ∇J_n by an estimate $\nabla \hat{J}_n$.

$$\nabla J_n = \frac{\delta J}{\delta \mathbf{f}} = \frac{\delta E\{e(n)^2\}}{\delta \mathbf{f}_n} \tag{6.7}$$

$$\nabla \hat{J}_n = \frac{\delta \hat{J}}{\delta \mathbf{f}} = 2 \cdot e(n) \cdot \frac{\delta e(n)}{\delta \mathbf{f}_n} \tag{6.8}$$

We have now replaced the mean-square-error $J = E\{e(n)^2\}$ with the momentary squared error $J = e(n)^2$ in the steepest descent algorithm. We now have an approximate steepest descent algorithm, called *the LMS algorithm*:

$$\mathbf{f}_{n+1} = \mathbf{f}_n - \frac{\alpha}{2} \cdot \delta \hat{J}_n \tag{6.10}$$

The signal $y(n)$ in figure 6.2 is the convolution $x(n) \otimes f(n)$, which in vector form can be written $y(n) = \mathbf{f}_n \otimes \mathbf{x}_n$. Here $\mathbf{x}_n$ is an L element vector with the L most recent sample values of $x(n)$: $\mathbf{x}_n = (x(n), x(n-1), \ldots, x(n-L+1))$.

$\nabla \hat{J}(n)$ could be found through:

$$e(n) = d(n) - y(n) = d(n) - \mathbf{f}_n^t \cdot \mathbf{x}_n \tag{6.11}$$

$$\frac{\delta e(n)}{\delta \mathbf{f}_n} = \frac{\delta\left(d(n) - \mathbf{f}_n^t \cdot \mathbf{x}_n\right)}{\delta \mathbf{f}_n} = -\mathbf{x}_n \tag{6.12}$$

$$\nabla \hat{J}_n = 2 \cdot e(n) \cdot \frac{e(n)}{\mathbf{f}_n} = -2 \cdot e(n) \cdot \mathbf{x}_n \tag{6.13}$$

Equation 6.13 thus can be written as:

$$\mathbf{f}_{n+1} = \mathbf{f}_n - \frac{\alpha}{2} \cdot \nabla \hat{J}_n = \mathbf{f}_n - \frac{\alpha}{2} \cdot (-2 \cdot e(n) \cdot \mathbf{x}_n) = \mathbf{f}_n + \alpha \cdot e(n) \cdot \mathbf{x}_n \tag{6.14}$$

This is the final Least-Mean-Square update algorithm.

The choice of f_0 and α

Since J can be seen as a surface with only one single minimum, the choice of $\mathbf{f}_0$ is not a big issue. We could without problem set $\mathbf{f}_0$ to zero. The choice of the adaptation constant α is far more important. α determines the step size within each iteration, and thus the speed of which $\mathbf{f}_n$ approaches $\mathbf{f}_{opt}$. If α is chosen small, the convergence towards $\mathbf{f}_{opt}$ will be slow, and if α is chosen bigger the filter coefficients will be more dependant of the input samples in $\mathbf{x}_n$, that is, will vary more with the stochastic input $x(n)$. The filter vector will be fairly *noisy* around $\mathbf{f}_{opt}$. And a too big α will result in an unstable algorithm, which doesn't converge towards $\mathbf{f}_{opt}$.

Types of adaptive filters

We have several different methods of making the filter update with time. The most natural way would be to let the recursive filter be adjusted with every new sample value coming into the system. This would cause no abrupt changes to the output signal and the processing load is constant. This is the fastest and most accurate method of adapting the filter, but is sometimes discarded in real-time systems because of the heavy processing involved.

With less processing power available, we could instead be taking *time windows* of the input and output signals, performing calculations to create a new set of coefficients, and update according to the window length. This will introduce a time lag between signal changes and filter changes,

give abrupt changes in the output signal and the processing load will come in bursts. It can however lower the total processing load or more advanced algorithms could be used.

For our purposes of off-line processing the processing time is not crucial, so we've only used updating with every sample.

Music vs. white noise

Music is correlated and thus not contains all frequencies, while white noise by definition is flat through out the spectrum. White noise is the best signal for adapting, and gives the fastest rate of adaptation. If any spectral component is missing, these frequencies will be heavily affected by noise. The adaptation speed of the LMS algorithm also depends on the input power and the constant of adaptation.

Proceedings

We have now seen how the adaptive filter can be used and how the LMS algorithm is derived. In the adaptive calculations of the filter coefficients we have not, though, been able to work in an *on-line* situation, since the hardware (as described in chapter 8) does not allow **two** inputs working at the same time. It is a one-channel input/output system and for the adaptive system we would need one input for the signal $x(n)$ and one for desired signal $d(n)$.

The adaptation is however perfectly possible to do in an *off-line* mode. If the signal that we use for adapting, whether it is white noise or music, is stored on the computer hard-drive, we can calculate the coefficients in a program such as Matlab (see Aappendix C).

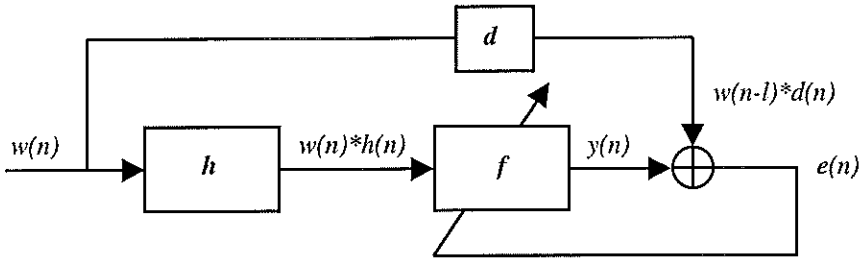


Figure 6.4: An off-line model of the used adaptive system.

With adapting with white noise, we should generate as long sequence of noise as we think would be sufficient for precise adaptation, in our case 40 seconds seemed to be both realizable and sufficient for both speakers. Convolving the noise sequence with the impulse response of the speaker gives us the input to the filter. Delaying and filtering the noise with our target function gives us the desired signal. The adaptation would take a few minutes on a PentiumII 233Mhz computer.

In our application, for filter f to be realizable, the target response filter d must be at least an all-pass with pure delay equal to the acoustic propagation delay between the loudspeaker and the measurement microphone.

Possible limiting factors

The ability to generate an inverse with low mean squared error is somewhat limited by some factors of the inverting system.

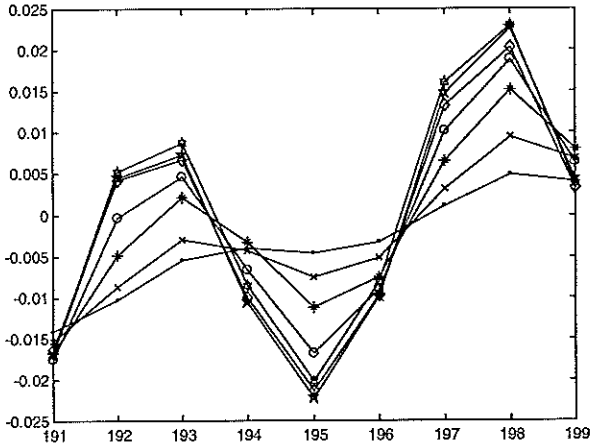
The noise in the system to invert, in this off-line case in measurements of the impulse response, causes the filter coefficient to be "noisy" and not completely accurate. An excellent inverse can be realized with precise adaptation in a less noisy system.

We also here have to introduce the delay to compensate for the sound-path delay, and for the non-

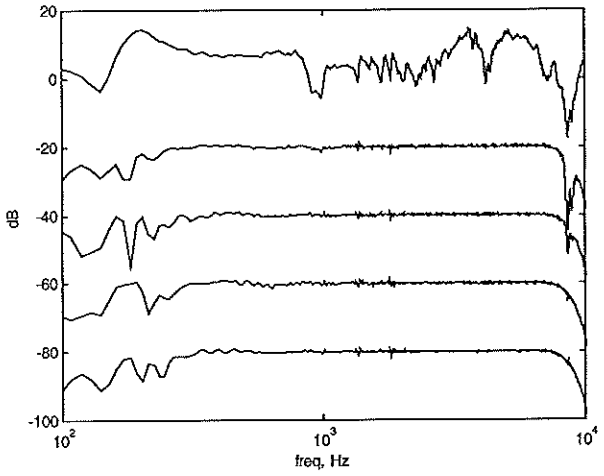
minimum part of the system which would otherwise demand a non-causal filter. The choice of delay time is not critical and 100 samples, about 2.3ms, seems like a good general delay time for our setup and speakers. And we can of course only approximate an infinite impulse response system with our limited FIR filter.

6.2 *Results*

To the right is an illustration of how the filter coefficients approach their final values during the adaptation. At the end there should be little variation in the coefficients, and this would mainly be due to noise in the measurements. We deliberately choose the adaptation constant α to a low 0.0001 to have very stable coefficients in the end.

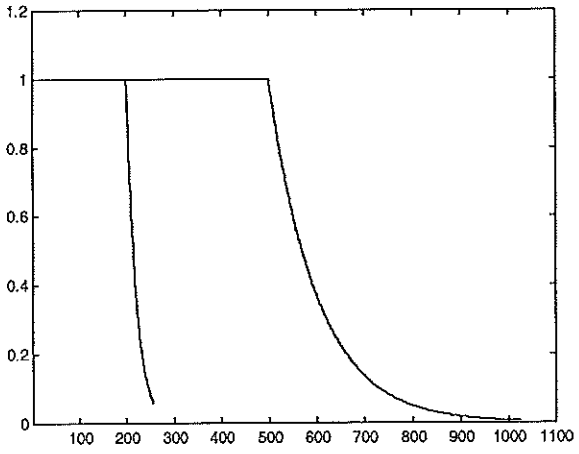


Here we see how the 1024 tap filter adapts the small speaker's frequency response into a very much better amplitude response. The curves are shown for 0, 1, 5, 20 and 40 seconds of adaptation and plotted without smoothing of the frequency curve. It's worth to notice how fast the convergence is within the first second, the first 44100 samples. Still we use 40 seconds of adaptation, since it is quite obvious that the coefficients are still not settled earlier.



There could be some obvious advantages with truncating the filters in that you might get rid of some beginning and end problems if the filter coefficients doesn't begin and end close to zero. The truncating functions shown for the 256 and 1024 tap filter will decrease the end taps of the two filters and remove some distractions in the output.

It is obvious that there are no significant benefits for the longer filter, but for the 256 tap filter it gives slightly less interruptions in the impulse response and reduces some frequency irregularities.



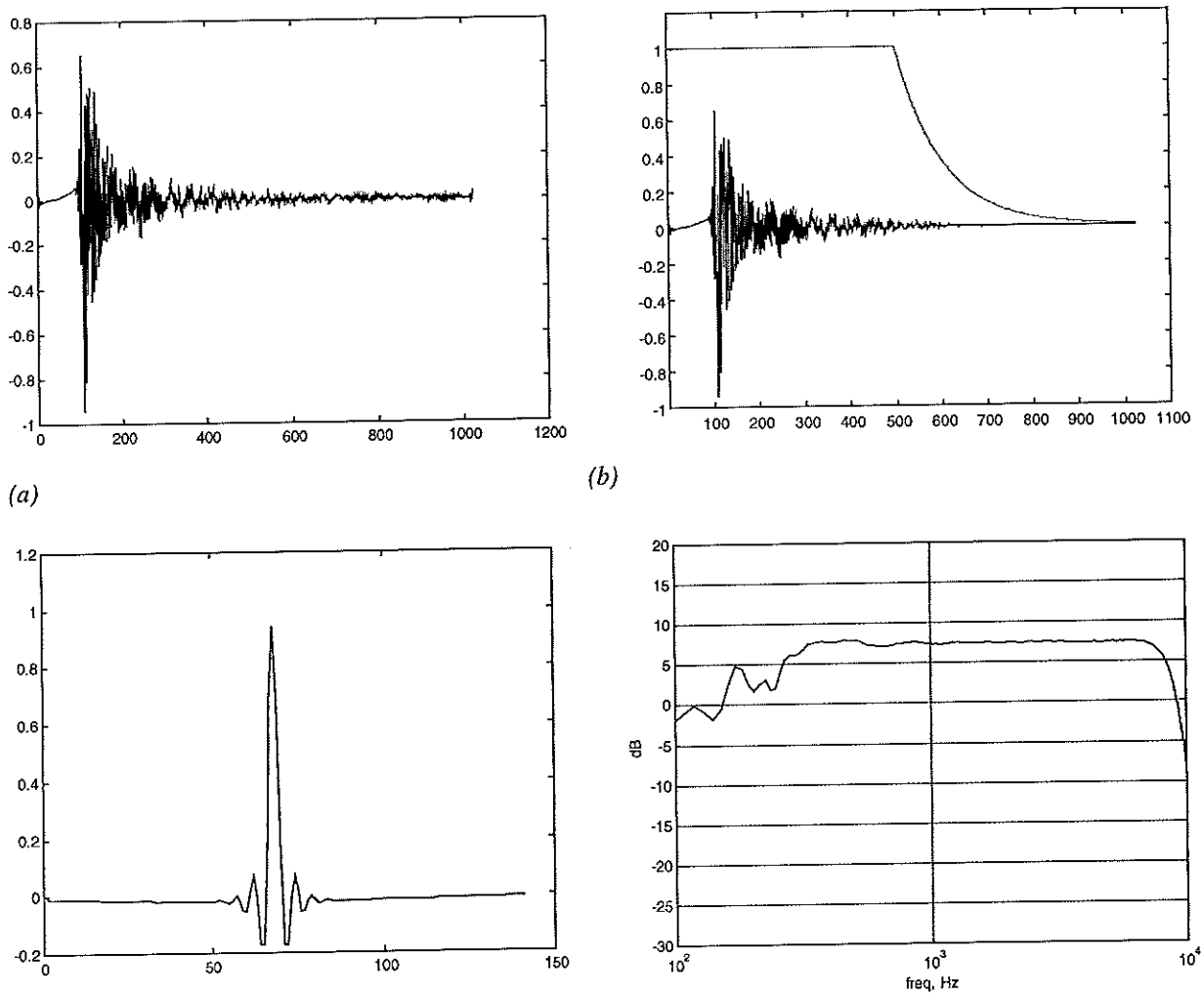
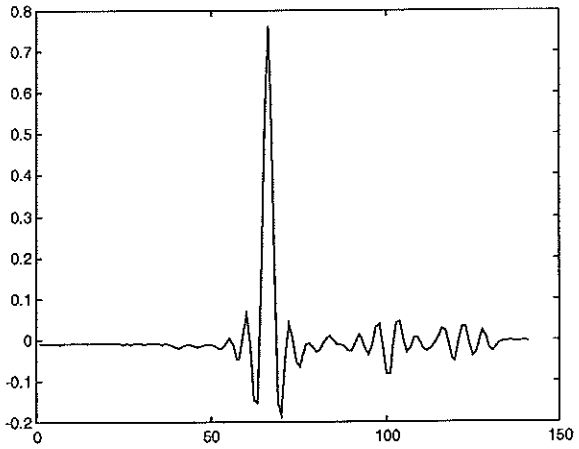
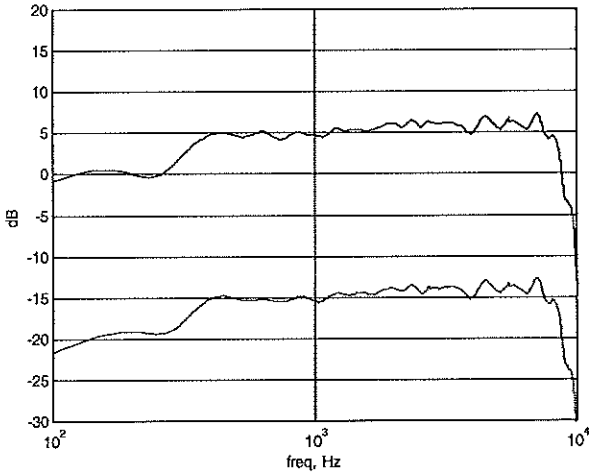


Figure6.5: The 1024 tap filter for the small speaker: (a) the filter, (b) the filter truncated and the corresponding truncating function, (c) simulated impulse response, (d) simulated frequency response.



(a)



(b)

The 1024 tap filter seems fairly long for correcting thr speaker, thogh we have some obvious lack of low frequency information. Truncation seems unnessecary. The Matlab simulated responses seems to be able to perfectly correct the speaker, and the real amplitude response shows a good result. The phase response is close to the ideal.

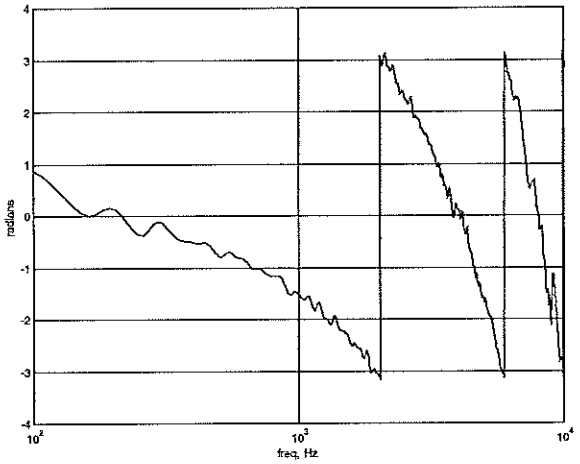
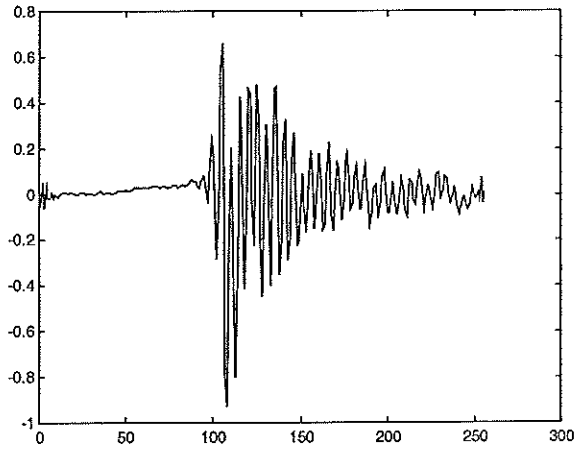
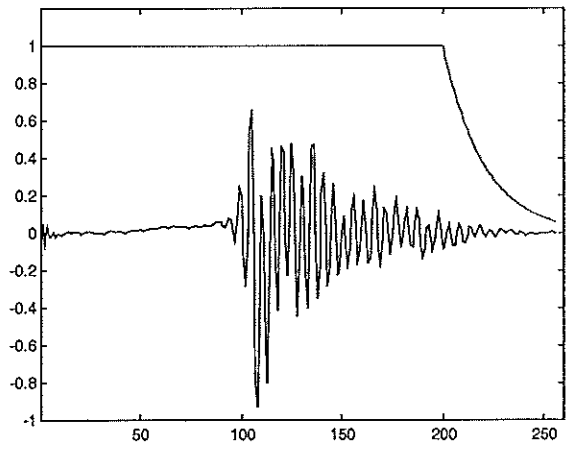


Figure6.6: The 1024 tap system response for the small speaker: (a) measured impulse response, (b) measured frequency response(truncated filter dotted), (c) measured phase response.

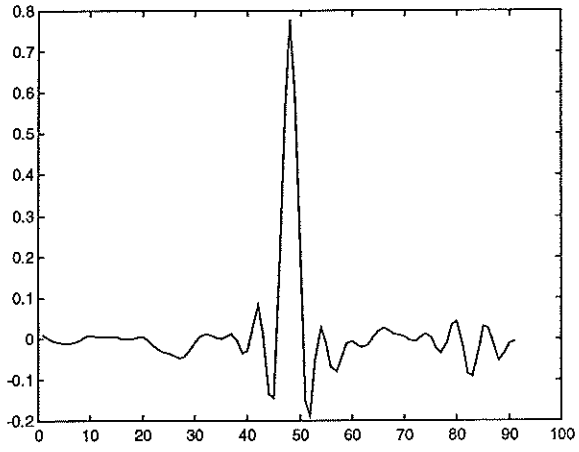
Results for the small speaker with 256 tap filter is shown in figure 6.7 below.



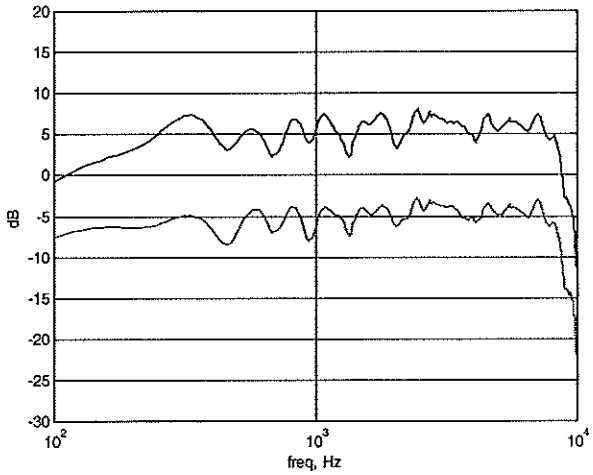
(a)



(b)



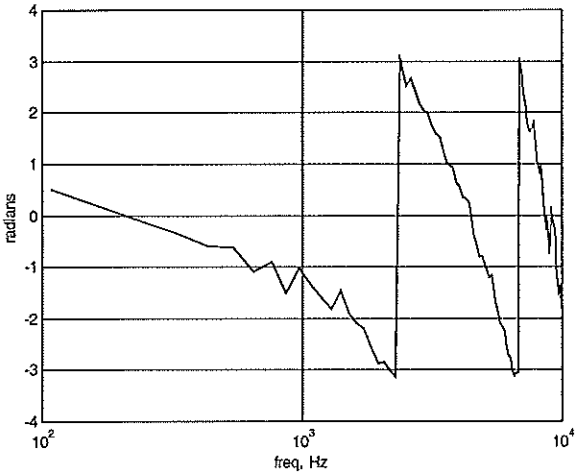
(c)



(d)

The 256 tap filter shown could benefit from truncation, as shown by the time and frequency plots. The longer filter however gives a slightly better amplitude response. Phase response is almost as good as for the longer filter.

Figure 6.7: Tthe 256 tap filter for the small speaker: (a) the filter, (b) the filter truncated and the corresponding truncating function, (c) measured impulse response, (d) measured frequency response(truncated filter dotted), (e) measured phase response.



(e)

The resulting filters and responses for the big speaker is shown in figure 6.8 below.

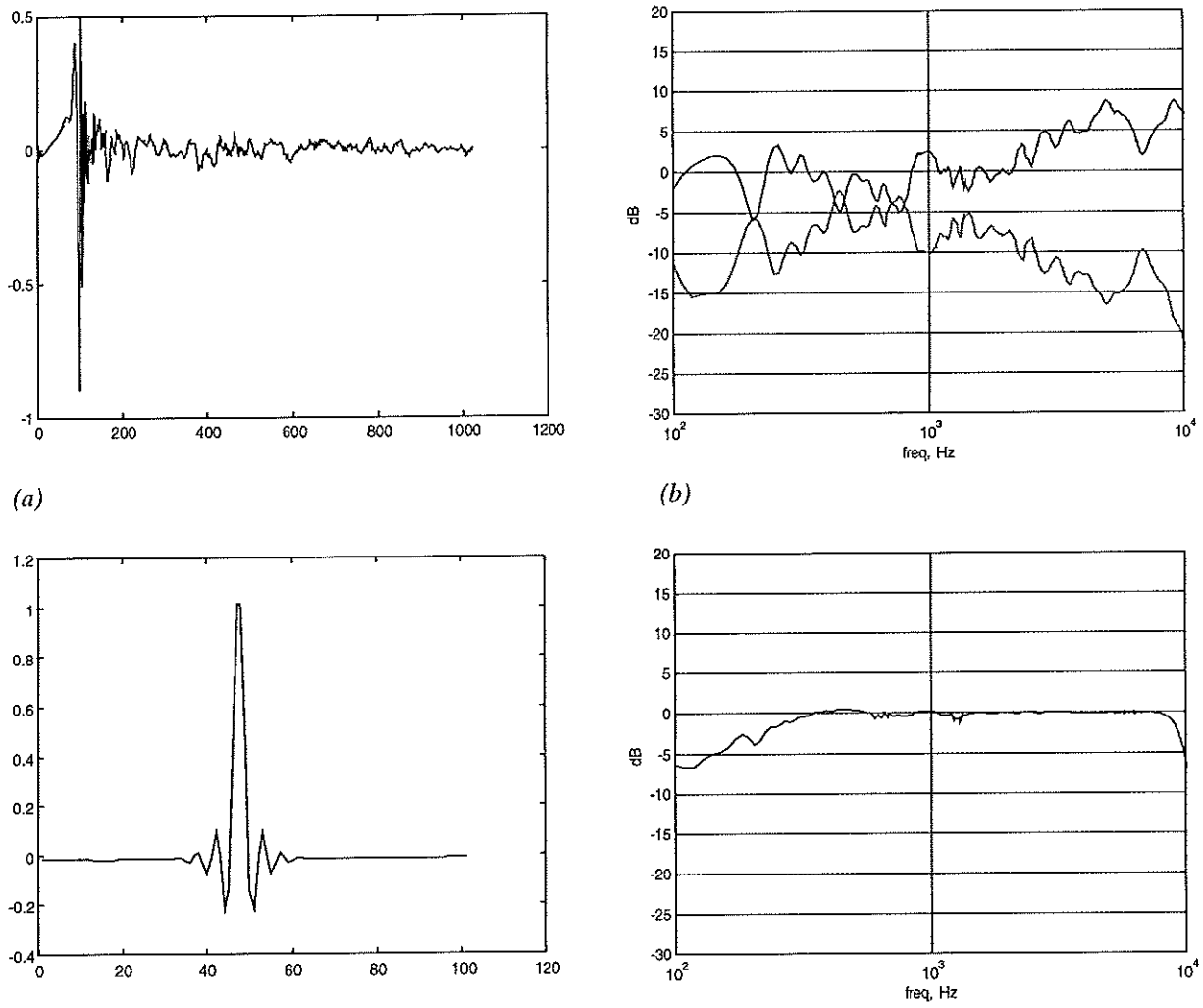
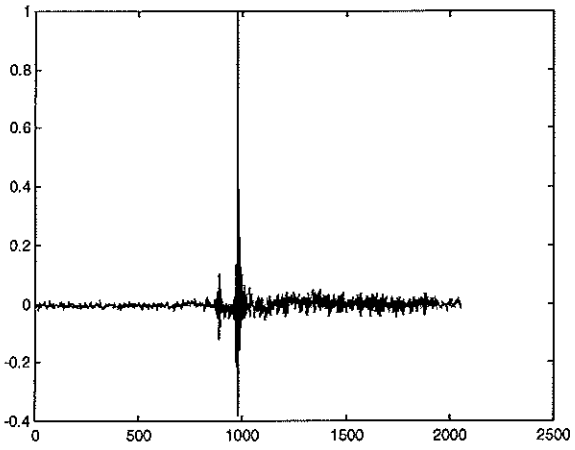
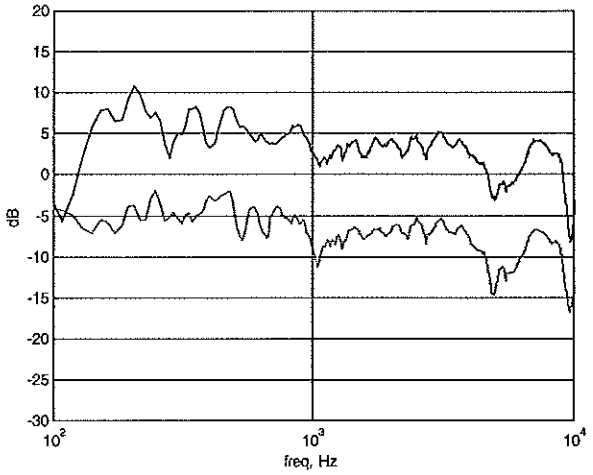


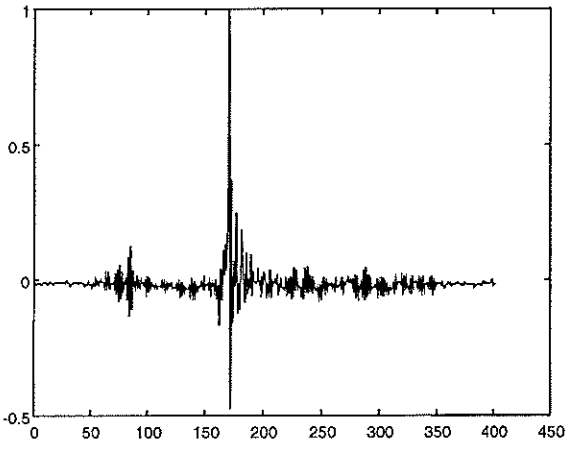
Figure 6.8: The big speaker: (a) the 1024 tap filter, (b) the frequency response of speaker and 1024 tap filter(dotted), (c) 1024 tap simulated impulse response, (d) 1024 tap simulated frequency response.



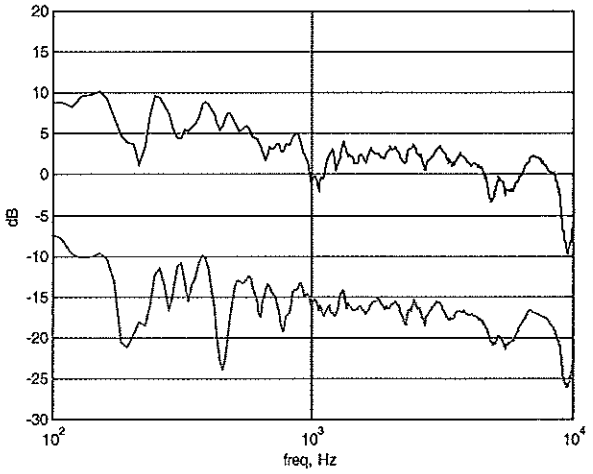
(a)



(b)



(c)



(d)

The 1024 tap filter seems fairly long for, and in simulated plots do the job very well. In the corrected response we see some disturbance with the start of the filter and it is obviously more complicated to correct amplitude and phase in a multi driver system.

The shorter filter is somewhat less accurate, and strangely enough truncating here introduce some frequency dips.

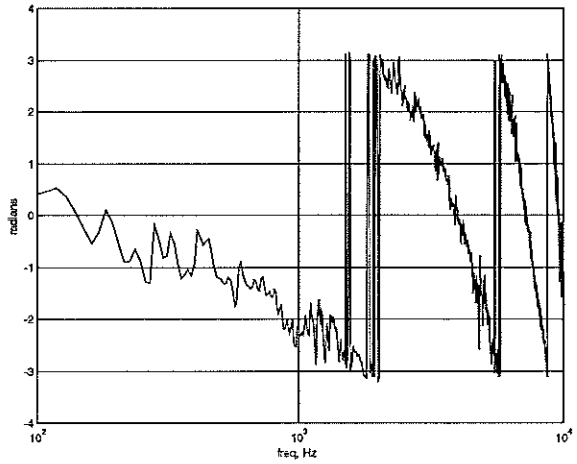


Figure 6.9: The big speaker system response: (a)1024 tap impulse response, (b) 1024 tap frequency response(truncated filter dotted), (c) 256 tap measured impulse response, (d) 256 tap measured frequency response(truncated filter dotted), (e) 1024 tap measured phase response.

6.3 CONCLUTIONS OF THE LMS METHOD

We have seen that the equalizing filter could also be calculated off-line using the adaptive algorithm, shown in Appendix D. This method even shows a better result in equalizing than the Least Squares method, and is after 40 seconds of adaptation converging to a pretty stable solution. Though this method should have some obvious advantages and possibilities if beeing processed in an on line situation, it works fine off-line too. We clearly have some more difficulties with the bigger 3-way speaker and its possible non-linearities and non-minimum phase properties.

7. Acknowledgements

Brett Ninness, Department of Electrical & Computer Engineering, Newcastle, for great supervision, hospitality, getting us started and keeping us going during our stay in Newcastle.

Björn Wittenmark, Department of Automatic Control, Lund, for supervision, cool temper, and making it possible to perform this thesis down under.

All the people at Electrical & Computer Engineering at Newcastle University, Australia, for helping out with all the small and big parts missing.

8. Summary

Great improvements in the loudspeaker response can be achieved with FIR-filters designed with any of our three methods tried. Especially when used under free-field conditions and for on-axis equalization. It is possible to compensate for both amplitude and phase errors with the methods we have used; Frequency Inverse, Least Squares method and Adaptive Least Mean Squares.

The Frequency Inversion method gave us the best results both with measurements and listening and our conclusion is that this could be the prefered method. Using this method we get a more intuitive understanding of how the filter works, because it is designed in the frequency domain.

The Least Squares method showed ease of use, though becomes computationally very heavy for longer filter lengths. It didn't really equalize as well as the other methods.

The Adaptive Least Mean Squares method showed good results, particularly with a single driver system. It has the obvious advantage of being able to adjust the filter parameters along with time if we have any change in our system. This could be practical if we want an easy set up that is able to also equalize the path from the loudspeaker to the listener, and can adjust itself to fit different environments.

Our results must be considered satisfactory, even though we encountered some practical problems along the way. The main goal of improving the performance of the loudspeakers has been met, though the results have shown some minor problems with the response details, especially for more complex loudspeaker systems. The impulse, magnitude and phase responses are significantly improved with all three methods, and the filters should be possible to implement practically with common DSP techniques.

The problems encountered can mostly be associated with non-linearities and poor measurement equipment.

FUTURE POSSIBILITIES

Other types of filters

Numerous other filter types not tried here could perform amplitude and phase corrections of a loudspeaker. An approach with IIR filters have been tried on many occasions, especially for bass equalization where FIR filters become of very high order. Warped filters (WFIR and WIIR) have recently been tried and found to give shorter filters lengths for the same degree of correction [4].

Convolving in hardware

The possibilities of designing an Application Specific Integrated Circuit (ASIC) for audio use would make very long filters possible. Other features like sample converting, manual equalizing could also be implemented.

Digital Cross-overs

The use of digital cross-over filters could offers further improvements. At cross-over frequencies the sound is radiated from two separate sources and preassure waves will be added in or out of phase as a function of off-axis position. Digital filters can be designed to have very steep cut-off slopes, and thus limit this problem. We can also limit resonances and *break-up*'s just outside the drivers working range, wich has to be well attenuated. Previous works on digital cross-overs have been presented in [15].

A "completely" digital reproduction chain

Great improvements can be expected when we apply digital filtering to an existing design, but

even greater possibilities arise if we include digital filters in the initial design of the speaker system. We could use DSP for correction and cross-overs in each frequency band in a 3-way system, using dedicated sampling frequencies.

With the newly developed "all digital" power amplifiers, witch takes a digital signal as input and uses a switching (PWM) output with very good results, we could keep the signal path in the digital domain all the way to the speaker drivers. The switching electronics are far more efficient, doesn't need heatsinks, and could be made far smaller and lighter than the usual amplifier. In this way it should be possible to reduce costs significantly since we could integrate the entire signal processing and amplification into the speaker system, all with dedicated electronics. A figure of a possible "all digital" reproduction system is shown in figure 7.1.

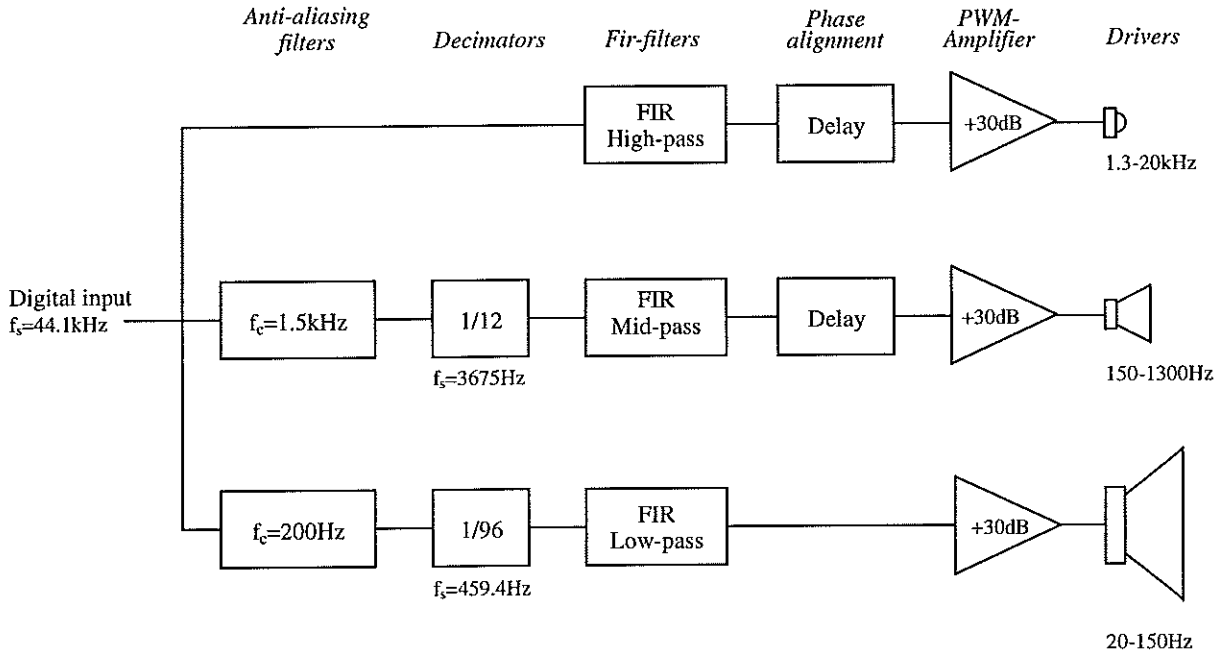


Figure 7.1: An example of a possible "all digital" reproduction system.

9. References

[1] R. Greenfield and M. Hawksford - "Efficient Filter Design for Loudspeaker Equalization", presented at the 86th Convention of the Audio Engineering Soc., p.p. 2764, March 1989.

[2] K. Dalbjörn and K. Åsnes - "Digital Filtering for the Improvement of Loudspeaker Performance in a Listening Environment", Report E 91-02, Department of Applied Acoustics, Chalmers University of Technology, Gothenburg, 1991.

[3] J. Kuriyama and Y. Furukawa - "Adaptive Loudspeaker System", Journal of the Audio Engineering Soc., vol.37, no.11, pp.919-126, 1989.

[4] M. Karjalainen, E. Piirilä, A. Järvinen and J. Huopaniemi - "Comparison of Loudspeaker Equalization Methods Based on DSP techniques", Helsinki University of Technology, Journal of the Audio Engineering Soc., Jan/Feb 1999.

[5] J. Dattorro - "The implementation of Recursive Digital Filters for High-Fidelity Audio", Journal of the Audio Engineering Soc., vol36, no.11, Nov 1988.

[6] A.V. Oppenheim and R.W. Schafer – "Discrete-Time Signal Processing", Prentice Hall, New Jersey, 1989.

[7] P.M. Clarkson - "Optimal And Adaptive Signal Processing", CRC Press, Florida, 1993.

[8] B. Widrow - "Adaptive Signal Processing", Prentice Hall, 1985.

[9] J. S. Lim and Alan V. Oppenheim - "Advanced Topics In Signal Processing", Prentice Hall, Englewood cliffs, NJ, 1988.

[10] van der Ecke - "Discrete-Time Signal Processing", Prentice Hall, Verhoecks, 1989.

[11] U. Zölzer - "Digital Audio Signal Processing", Wiley & Sons, Chichester, 1997.

[12] F. J. Taylor - "Digital Filter Handbook", Marcel Dekker Inc., New York, 1993.

[13] J. D. Turner and A. J. Pretlove - "Acoustics for Engineers", Macmillan Education LTD, London, 1991.

[14] J. G. Proakis and D. G. Manolakis - "Digital Signal Processing: principles, algorithms and applications", Prentice Hall, New Jersey, 1996.

[15] M. Colloms - "High performance loudspeakers", 4:th edition, Pentech Press, 1991.

[16] R. Wilson, G. Adams and J. Scott - "Application of digital filters to loudspeaker cross-over networks", Journal of the Audio Engineering Soc., vol.33, no.33, pp.127-132, 1989.

APPENDIX

APPENDIX A - HARDWARE

Signal processors are specifically designed for repetitive, heavy load calculation purposes. Well suited for real-time processing, these processors are popular for instance for collecting and analysis of analog signals, and are nowadays very common among audio and video processing.

Signal processors come in two basic types: programmable or "hardware coded". The difference is that the programmable are completely software controlled, and the programmer has control over its function. The "hardware coded" are designed for its specific purposes, and you can't change its function with programming.

We have only been involved with the programmable processors.

The signal processor has much less instructions available compared to a general type processor, and it's instead optimized for doing the important instructions very fast. For example the MAC-instruction, Multiply And aCcumulate, computes *two* instructions in the same clock-cycle. That's an essential part of a convolution algorithm, a typical task for the signal processor.

Floating-point or fixed-point

There are two different types of programmable signal processors, floating-point and fixed-point processors, and they differ in their arithmetics:

In fixed-point processors the numbers are represented as real numbers with a fixed precision. It's important to know the size of the numbers so we don't compute into overflow or scale down the number to a poor precision.

In the floating-point processors the numbers are represented as a mantissa and an exponent. With this type of processor we work with maximum precision in the number all of the time, thus we don't have to be that concerned about size of the numbers.

Generally, the floating-point processor is easier to work with. It also offers better dynamic range and precision. For fast development and small series this processor is usually the most convenient type. The fixed-point processor on the other hand is simpler built, takes up less silicone space, consumes less power and is generally cheaper.

Development tools

Earlier the designers of DSP systems were forced to program their chip at an assembler level, but nowadays there are numerous different tools to use during the development phase. There are software tools from programming and debugging at a very low level, to pure DSP operative systems on a very high level to make the programming of the chip as easy as possible.

On the lowest level there are simple tools as assemblers, linkers and debuggers. With knowledge and skills you could get the best performance from your chips on this level.

Next level involves programming the chips in a high level language as C and use compilers to generate the code. The code won't be as efficient as on the lower level, but on the other hand, high level development is usually much faster.

Even easier is it on the next level when using libraries with existing code for special purposes as FFT and filters.

You could also choose to develop in a "block diagram environment". The building blocks could be existing functions that only have to be graphically connected on the screen.

The Texas TMS320 processor family

In the project we've been using a DSP evaluation kit from Texas Instruments. It uses the Texas Instruments TMS320C31 signal processor, which is a general-purpose floating-point processor. The chip is quite "hardware intense", and implements many functions in hardware that otherwise would have had to be implemented in software. This makes the processor quite powerful, and the processing performance is maximum 50 MFLOPS.

The processor is actually capable of 25 MIPS, but since we have the possibility to do only double instructions like the MAC-instruction, we can ideally still do 50 MFLOPS.

The processor has several good features to increase performance:

- The CPU has independent multiplier and ALU and many flexible registers.
- Good memory handling, where four memory positions, including on chip or off-chip memory, could be accessed on the same clock cycle.
- 7 internal buses to fetch and store instructions and data efficiently.
- Two DMA controllers, *Direct Memory Access*, lets the processor access memory without the CPU being involved.
- Flexible ports that let you communicate easily with the surroundings.

Just like many other modern processors, the TMS320C3x family processors are also pipelined. It has five pipeline steps, and the throughput could ideally be increased by five times.

Circular buffer

Circular buffer is a frequently used method when calculating correlations and convolutions. It's a memory area where you access the memory positions in order, and incrementing the last reference will set the pointer to the first position again. When convolving it can bee seen as a "sliding window" which shows the last samples of in-data. When the new one is written in the buffer, the oldest one is written over, and you always have the most resent samples in the buffer.

Assembler

The programming of the processor for the real-time evaluation is all done in assembler-code. This involves some extensive initializing of the peripheral circuits on the DSP board and the processor itself before we can start processing the incoming samples. We need to arrange the memory locations and locate code, initializee the serial port, set up timers and registers. The sampling/reconstruction circuit also needs to be initialized to sample at its maximum precision, 14bits and 20kHz, and do the correct tramsmissions of data.

During processing the program recieves the incoming sample and converts it into floating point data. The sample is then put in the circular buffer in the "oldest" position. We have pointers to the circular buffer with samples and to the list of coefficiens, and we apply the MAC(mutiply-and-accumulate) instruction on the whole set of coefficients. We then convert the output sample back to integer and send it to the D/A converter and wait for another interrupt from a new sample.

The DSP evaluation kit also provide the useful and simple DSK Assembler and the DSK Debugger that we have been using for generating and fixing the code.

The fact that the real-time system only allows for sampling at a maximum of 20kHz sampling means that all the filters created for the common CD Audio standard of 44100kHz sampling frequency has to be downsapled to 20 Khz filters. This is the major reason for limiting all signals involved to 10 kHz. There is also less precision in the sampling with the 14 bits resolution, but this still allows for 84 dB SNR and is well within the SNR of the signals wich the filters were based upon, and should not affect the correction.

APPENDIX B - MEASUREMENTS

The most common measurements that are done on speakers by designers and independent testers, mostly hi-fi magazines, covers sensitivity, distortion, impedance, magnitude and phase response at various points (on-axis and different degrees off-axis). It is also common to show the impulse and square wave response of the speaker in the time domain.

We want to correct the magnitude and phase response of the speaker and must therefore be able to measure these variables of the speaker in a simple and correct way. There are a several methods to choose from. Sine sweep, white noise, step response, maximum length sequence, and impulse response are some common methods.

Objective measurement alone is insufficient to fully describe the sound quality. Subjective evaluation is definitely essential before making to many sound quality conclusions. But measuring is indeed an very important and essential part in high quality speaker designing, not least cause it gives easily comparable results.

The measuring system

To measure the magnitude and phase response of a loudspeakers, you need a signal generator, an amplifier, the loudspeaker, a microphone and a microphone amplifier and some kind of recorder. Some online realtime processing of the measured signal is also often directly included in the recording environment. Today it is most common with totally digital measuring system which obtain a great flexibility, cause of easily stored and processed data etc

This is a hard part and there is many different approaches about how a speaker should be measured.

The measuring must also be made in some kind of environment, usually some kind of room, but is sometimes carried out outdoors. It is common to do the measurements in an anechoic chamber, which means a room that is totally free from echoes, optimally for all frequencies, but realistically down to about 100 – 200 Hz.

A question is if it right to measure and evaluate a speaker in an anechoic chamber, when the real listening will take place in a normal room with some colouring of the room (resonance and reflections). At least the lower frequencies can in some ways be determined how they will be affected of a normal room and should therefore be included in the design.

This is anyway a never ending discussion and we will not discuss it further here, but instead define that we want a speaker to have a flat magnitude and linear phase response when measured in an anechoic chamber down to about 200 Hz. We also believe that it is more accurate to evaluate the speaker from some different positions than just on-axis. This is because the magnitude response can be very different measured from slight different positions on and off-axis. Sharp dips can arise because pressure waves are out of phase in exactly the measured position. Trying to correct for this dips could results in a very bad sounding system. Some kind of averaging could be made between the different positions. However, since it is much easier to make the measurements in only one point and we have to do a lot of measurements and evaluation that should be repeatable we have made all the measurements in only one position.

Since we didn't have access to an anechoich chamber we had to use other methods to obtain a good measurement in a normal room. There are ways to measure a good estimate of a loudspeakers response in an anechoich chamber in a normal room The most common method to do this is to use an impulse response as the measuring signal, more about this later.

Industry Standard

The industry standard measuring system is definitely *MLSSA* (pronounced "Melissa") by *DRA Laboratories*. *MLSSA*, an acronym of Maximum-Length Sequence System Analyzer, pioneered the MLS (Maximum-Length Sequence) measurement method, which offers a good combination of speed, noise immunity and time-bandwidth product. The *MLSSA* system is a complete signal generating, recording and processing system containing both software and hardware. The hardware and software is to be used in an ordinary PC. The solution is really state of the art and offers incredibly many different features and a huge number of ways to show and evaluate measured data.

Our test environment didn't include the *MLSSA* system that had been preferred. Instead we used software called *SpectraPlus* for signal generating and recording . *Matlab* was used for postprocessing of the measurements and construction of the signals. A normal PC with a simple sound card called *ESS 1868* were used as hardware.

Even though the *Spectra Plus* program did include some of the realtime and post processing methods of the measurements like *MLSSA*, we didn't use them for others than some fast online conclusions. The measurements was saved as wav-files with 16 bit resolution and 44.1 kHz sample frequence i.e. the common digital audio format. Wav-files can be easily converted to matlab-files and vice versa so all processing could be done in Matlab which give full control of what is happening.

Signal Generator

The signal generator should correctly produce the different signals needed for measuring. By using a digital signal generator in form of a PC with soundcard and software *Spectra Plus* some we obtained good flexibility. An important thing to remember is that the signals produced by any kind of digital signal generator will be band limited according to the Nyqvist theorem i.e. half the sample frequency.

All test signals were made in *Matlab* and saved as wav-file which is the format that *Spectra Plus* handle. This seemed, by some reason, to give better results then compared to use the built in signals in SpectraPlus. All signals were made with 16 bit resolution and 44.1 kHz sample frequency.

Amplifier

To drive the loudspeakers with the test signals we have to use an amplifier. The requirements of the amplifier is that it should be as transparent as possible, i.e. have a flat magnitude response and a linear phase. No amplifier is perfect, but compared with other parts like the soundcard and the microphone the amp can often be assumed transparent.

We used a regular hi-fi amplifier, *Sony TA-F170,* which were set into "signal direct", which disconnect the bass and treble controls and should be the most transparent setting.

Microphone

The microphone sense the sound pressure produced by the loudspeaker. This is definitely the most critical component in the measuring chain. The microphone should preferably have linear phase and amplitude response and a trustworthy calibration protocol. With a calibration protocol the distortion from the microphone could be excluded from the measured signal.

Since the microphone delivers very low amplitude signals, they need to be amplified to line level voltage (about 1V amplitude) before fed to an a/d converter that converts the signal so it can be dealt with digitally. The amplifier is called a microphone amplifier and are because of the low level signals that it deals with much more sensitive fore noise and distortion than the amplifier feeding the loudspeaker.

In our test environment we had a *Shure SM-58* microphone, which is a well known rather high

quality singing microphone. However, it is not as good to use as a test microphone because it doesn't have a totally flat magnitude and linear phase response. We didn't have a calibration protocol of this particular microphone either. This means that we have to be aware of the fact that rather than just measuring the loudspeakers the response from the microphone will be included as well.

As long as only the evaluations of the measurements are concerned this is not really a big problem. But doing useable listen tests is harder because the equalizing of the filters we design will correct the whole system, rather than only the loudspeaker as attended.

The microphone amplifier that we used was included in the soundcard *ESS 1868*. This had a very bad signal to noise ratio, which was the source of the most noise in our measurements.

Recorder

A recording system that can obtain the signal from the microphone and store it for later use is required for being able to postprocess the measurements. A digital recorder is preferred because of the possibility to easy store the results digitally and process the measurements with digital signal processing.

We used the PC and the soundcard *ESS 1868* as a recorder. *Spectra Plus* was used as recording software. We always used standard CD sample frequency 44.1 kHz and 16 bit resolution as mentioned before. Which give a high frequency limit of 22 kHz and a digital signal to noise ratio of around 96 dB which (from the rule of thumb 6 dB S/N ratio per bit) is much more than the rest of the system. In figure *A.1* the magnitude function of the background noise is plotted vs the magnitude function of the single driver speaker with our two most used measuring signals impulse and white noise. We can see that the signal to noise ratio, if using an impulse as input signal is between 25 and 40 dB.

The Measuring Room

As already discussed it is common and good to use an anechoich chamber for measuring. But since this wasn't available we had to use an ordinary room. This room was not especially well suited for loudspeaker measurements. It had rather high background noise and a lot of surfaces, because of lab bench etc. which could cause reflections and resonance. We made the best about it by doing the measurements rather high, to decrease floor reflections. By position the microphone close to the speaker and use an impulse as the main test source we did our best to avoid the room reflections and resonance.

There was no other way to come away from the high background noise in the room, but to use as loud test signals as possible. But not as loud as the speaker will be close to close to its dynamic limit.

Different measuring signals

Sine Sweep

Is a rather common method for measuring in anechoich chambers, and can be made with analogue equipment. A sine wave generator is set up to sweep the frequency range of interest (usually increasing logarithmically) and the signal from a microphone is fed into a chart recorder which measures the signal level in time. Of course a digital recorder could be used instead and processing of the measured data after the measuring is done. The time axis on the recording is then given as the frequency axis. There are, however, many shortfalls with this method. It doesn't give you any phase information, there are big problems with room resonance and room reflections since the test signal last during a long time. Making measurements close to the loudspeaker and/or, of course, make the measurements in an anechoic chamber can minimize this problem. Distortion can mean that the magnitude measured at one point in time consists of more

than one frequency thus giving false results. Practice has also showed that there are significant variations in the result based purely on changes in sweep speed and pen response of chart recorder.

With gated sine wave excitation which simultaneously samples and stores the peak amplitude of the received output long enough for a continuos response trace to be recorded will result in elimination of the local wall reflections and the need for an anechoic chamber is removed. Gated sine wave excitation is well described in [1].

We haven't at all used sine sweep as a test signal because of the shortfalls. It does not provide phase information, is not suited to measuring in normal rooms, and requires a special (more complex) processing to give the frequency response if not using an analogue chart recorder.

Impulse

The most straightforward way at least if thinking in signal processing terms is to generate an impulse and measure the response of the system. It's easy to see in the time domain how the speaker's response differs from a perfect impulse. The impulse has some nice advantages than it comes to use it as a test signal for speaker measurements. It is easy to derive the amplitude and phase response by taking the DFT of the sampled speaker response. It makes sense to look at the speaker's response not only in the frequency domain but also in the time domain. In the time domain it's rather easy to see if where is some big room reflections, that can be excluded by windowing the speakers response to an adequate number of samples (length) of the response. If using for example 512 points of the measured response corresponds to a length in time of approximately 12 ms (given a sample frequency of 44.1 kHz). During that time sound travels around 4 metres in air, which means that possible included reflections must come from surfaces not more than 2 metres from the speaker.

One disadvantage of using the impulse as a test source, is that the power that is delivered to a speaker is low. This gives a bad signal to noise ratio. Another is that in typical rooms, such windowed impulse measurements will not give accurate data for lower frequency than about 100-200 Hz [3]. To get accurate information further down in the frequency spectra some kind of other methods have to be used.

The impulse has been used for all of our equalization methods, except the adaptive one. The main reasons why we choose this method is because of the ability to get phase information and the possibility to avoid most of the room interaction. To be able to just look at the response of the speaker directly in the time domain and make some conclusions is also appreciated.

White Noise

Random noise excitation is another common method. White noise has a power density spectrum which is constant over all frequencies. The output sound level can be much larger than the background noise level which in a good way overcomes signal to room noise ratio. With digital recording the amplitude response is obtained from the sampled test signal by taking the DFT of the samples.

From a philosophical point of view, noise more closely resembles music in transient content than for sine waves, and therefor could be considered more relevant as a test source. Because of the method's random nature, there is a need for smoothing of the (with DFT) calculated amplitude response. Even if using extremely long sequences, like 200000 samples the variance of white noise is still quite high at around and this is reflected in the resultant spectrum as random perturbations about the true characteristics. We have used 1/8 octave smoothing and a sample length of 16384 samples. A shortfall is that this method doesn't give any phase information.

We used white noise for measuring in some cases, mainly to check that measuring with impulse and noise gave similar results. The sampled test signal of the noise was windowed with a rectangular window to 16384 samples and taken the discrete fourier transform of to obtain the

amplitude response. Smoothing of the amplitude response is definitely required.

Figure show that the impulse and the white noise measuring gives similar results. The big benefit with using white noise is the improved signal/noise ratio as clearly showed in the figure. Around 20 dB in signal to noise ratio is gained using noise instead of impulse as input signal.

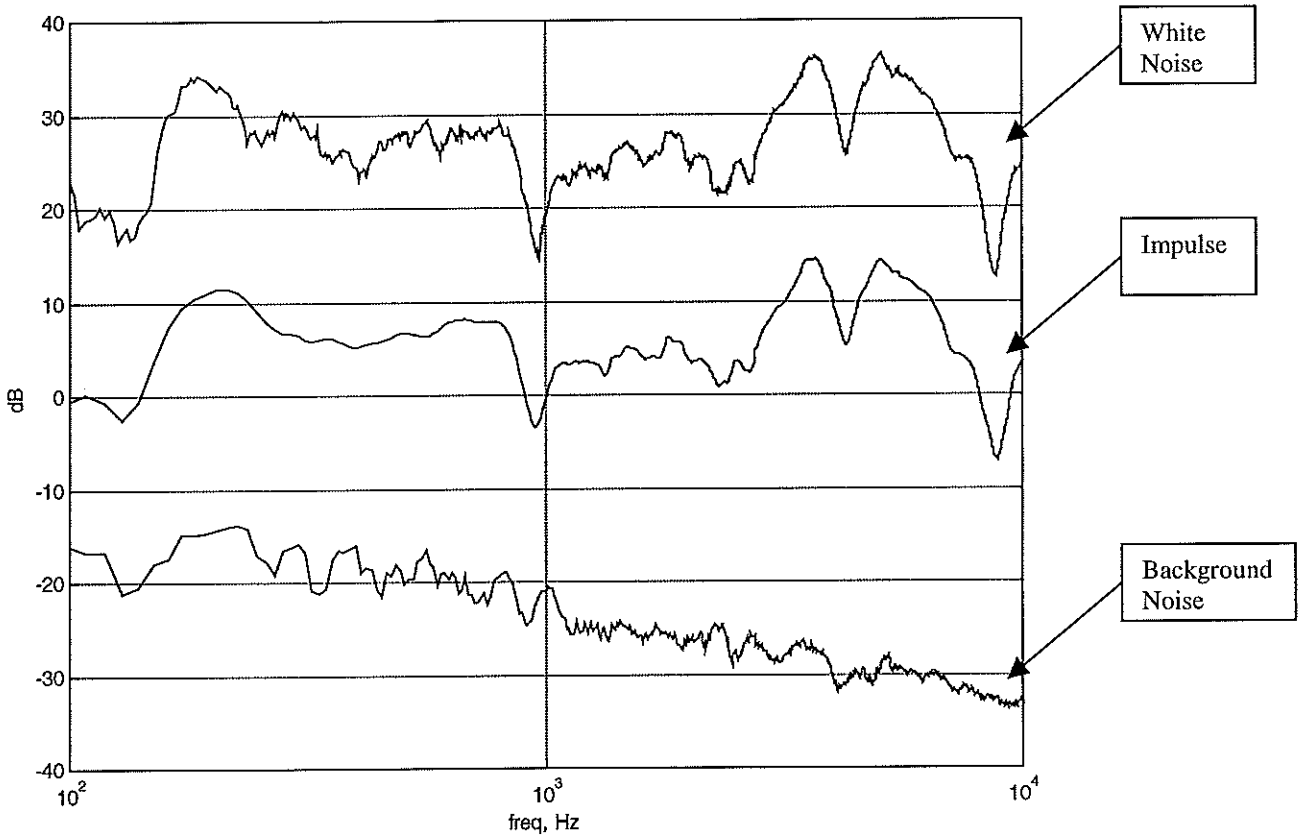


Figure A1: White Noise vs Impulse as input signal. Background Noise is measured with no input signal. All curves is smoothed 1/8 octave.

Maximum-Length Sequence

Probably one of the best test signal to use for obtaining the amplitude and phase response in an anechoic chamber. The maximum-length sequence is a kind of pseudo random noise, that has all the advantages that measuring with white noise has, but also gives phase information. This method is highly used in the MLSSA system.

Since our test environment didn't include this advanced measuring method and since it is best used in an anechoic chamber, we haven't used it.

Smoothing

There is strong evidence that the ear tends to average short-term irregularities and is rather sensitive to broader trends in energy over third-octaves or octave bandwidths.

APPENDIX C - SPEAKERS AND MEASUREMENT SETUP

SMALL SPEAKER	BIG SPEAKER
"Noname"	*Technics*
Type: One way, closed box, no cross-overs.	Type: Two way, closed box.
Dimensions: H,W,D; 37.5,25,17cm	Dimensions: H,W,D; 60,32,19.5cm
Drivers: 6" wideband	Drivers: Bass: 8", Mid: 2", Treble: 1"
	Cross-over freq.: 3500 & 8000Hz

Measurement equipment used:

Microphone: Shure SM-58. A very common dynamic, cardoid pattern, microphone designed for singing, and thus can deal with high pressure and rough handling. On the other hand, this microphone is not made for acoustical measurements, and has neither a flat nor a wide frequency response. With typical values the microphone is within 5dB from 100 to 10kHz.

Power amplifier: *Sony TA-F170,* 60 Watt hi-fi amplifier, measures flat throughout the audio band.

Signal generating, sampling and recording: *ESS 1869* soundcard, very simple full-duplex PC soundcard capable of 44100kHz sampling and recording to harddisk. Specifications were not available, but probably not a very good performer, and obviously got SNR of approx. 50dB.

Software: *SpectraPlus v.4,* shareware program that incorporates a lot of good analysis functions, such as time and frequency domain analysis, distortion measurement, spectral decay plots etc.

Measurement signals: Made in *Matlab,* converted to sound format and played back via *ESS 1869.*

The measurements were made in a "live" environment, in a fairly big electronics laboratory, but not very far from walls producing early reflections. With somewhat 3 meters path for the early reflections, with a near-field (10cm) measurement we would have our first room-interactions in about 9ms, or 385 samples, after the impulse. Because of this we have in some cases been truncating the impulse response to get a "free-field" like measurement, but this will in turn give us less resolution in the lower frequencies.

For the bigger speaker the measurements were carried out at 60cm because of the multiple driver design, and the measurements were even more disturbed by the room response.

Although the measurement conditions have been far from ideal, the aim has been to keep the conditions around the measurements as stable as possible. We have of course in this arrangement not only been measuring the speaker, but the whole measurement chain of components, and hopefully been able to arrange exactly the same conditions at the time for evaluation. After all it's been the *method* that's been of interest, not the performance of this particular speaker.

APPENDIX D - MATLAB SCRIPTS

Inverse Filtering

```
function filt=inversfilter(x,size_of_inverse)

% function filt=inversfilter(x,size_of_inverse)
% Target function is included, can be redesigned by
% editing this function.

%Design of target function:

lowpassorder = 33;
cutfrequencyL = 9000;
fs = 44100;

highpassorder = 1024;
cutfrequencyH = 100;

lowpassfilt = fir1(lowpassorder,cutfrequencyL/(fs/2));
highpassfilt = fir1(highpassorder,cutfrequencyH/(fs/2),'high');

targetfunc = conv(highpassfilt,lowpassfilt);

targetfft = fft(targetfunc',size_of_inverse);
%targetfft(1) = 0.0001;

X = fft(x,size_of_inverse);

H = targetfft'./X;

h = real(ifft(H));

filt = rotatevect(h,delay);

plot(filtsvar)
```

Inverse Minimum Phase Filter

```
function h = minfasfilter(x,size_of_inverse);

% function h = minfasfilter(x,size_of_inverse);

pad = zeros(1,size_of_inverse-length(x));

[y ym] = rceps([x pad]);
%real cepstrum with matlabfunction rceps.

h = inversfiltervhigh(ym,size_of_inverse);
```

Least-Squares

```
function fopt=lsfilter(p,l,t)
% p = impulse response
% l = used length of impulse response
% t = delay in desired response

rp=conv(p(1:l),p(l:-1:1));     % auto-correlation vector, length 2*l-1
Rp=toeplitz(rp(l:2*l-1));      % auto-correlation matrix, size l*l
Rpinv=Rp^(-1);                 % size l*l
g=zeros(1,l);                  % cross-correlation vector, length l
g(1:t)=p(t:-1:1);              % imp. res. shifted backwards into g
fopt=Rpinv*g';                 % optimal filter, length l

figure
subplot(3,1,1); plot(p);
    title('Impulse response p, inverse filter fopt, and their convolution');
    ylabel('p');
subplot(3,1,2); plot(fopt);
    ylabel('fopt');
subplot(3,1,3); plot(conv(p,fopt));
    xlabel('sample nr'); ylabel('conv(p,fopt)');
```

LMS

```
% Adapts towards the inverse of impulse response p for 40 sec.
% We must have p as variable.
% d=wavread('e:\project\lms\music');  % if adapted with music
d=randn(1800000,1);                   % white noise input
fprintf('Noise generated!\n');
x=conv(d,p);                          % input to filter
fprintf('Noise convolved with impulse response!\n');

d=[zeros(100,1); d];                  % delayed reference
lowpass=fir1(31,10000/22050);         % low-pass filter
d=conv(d,lowpass);                    % desired signal, delayed low-pass filtered music

l=1024;                               % filter length
k=0.0001;                             % adaptation constant
f=zeros(l,1);
L=length(d);

for n=1:(L-l*2)                       % LMS updating algorithm
    y=f'*x(n:-1:n-l+1);
    e(n)=d(n)-y;
    f=f+k*x(n:-1:n-l+1)*e(n);
end
    plot(f,'black');
fprintf('Finished!\n');
```

APPENDIX E - ASSEMBLERCODE

```
;*************************************************************************/
; The file programs the Analog IC to sample at 20KHz.                     */
; A serial port Receive-INTerrupt takes the values in                     */
; the serial port receive-registers and applies the FIR                   */
; filter before writing the result to the serial port transmit-register.  */
;*************************************************************************/
;-------------------------------------------------------------------------
         .start  ".text",0x809802    ; Locate code at beginning of on-chip memory
         .start  ".servect",0x809FC5 ; Locate serial port interrupt vectors
         .entry  start
;-------------------------------------------------------------------------
;  Memory map register locations
SGCR0    .set  0x808040              ; Serial port 0 global control register
SPCX0    .set  0x808042              ; Serial port 0 FSX/DX/CLKX control reg.
SPCR0    .set  0x808043              ; Serial port 0 FSR/DR/CLKR control reg.
DTX0     .set  0x808048              ; Serial port 0 data transmit register
DRX0     .set  0x80804c              ; Serial port 0 data receive register
TGCR0    .set  0x808020              ; Timer 0 global control register
TCNT0    .set  0x808024              ; Timer 0 counter register
TPR0     .set  0x808028              ; Timer 0 period register
;-------------------------------------------------------------------------
; AIC (Analog/digital Integrated circuit Chip-set) parameters
TIMERPER .set  1                     ; C31 Timer period register for 50/(8*TIMERPER)MHz
TA       .set  4                     ; TA value   for 20KHz sample rate
TB       .set  39                    ; TB value
RA       .set  4                     ; RA value
RB       .set  39                    ; RB value
PRIMARY  .set  11b                   ; Primary Communications indicating Secondary
                                     ;   Communications follows
A_REG    .set  (TA<<9)+(RA<<2)+0     ; Secondary Communication for Setting TA and RA
B_REG    .set  (TB<<9)+(RB<<2)+2     ; Secondary Communication for Setting TB and RB
C_REG    .set  ((101000b)<<2)+3      ; Control word
;-------------------------------------------------------------------------
         .text
start    ldi   2h,IOF                ; Pull AIC into reset
         ldi   0h,R4                 ; Clear R4
         ldp   SGCR0
         sti   R4,@SGCR0             ; Reset serial port
         ldi   @SINIT1,R7            ; Load initialization value 1 into R7
         sti   R7,@SPCX0             ; Initialize FSX/DX/CLKX control reg.
         sti   R7,@SPCR0             ; Initialize FSR/DR/CLKR control reg.
         ldi   @SINIT0,R7            ; Load initialization value 0 into R7
         sti   R7,@SGCR0             ; Enable RINT & 16-bit transfers
         sti   R4,@DTX0              ; Transmit 0
         sti   R4,@TGCR0             ; Reset timer 0
         ldi   TIMERPER,R7
         sti   R7,@TPR0              ; Store timer 0 period
         sti   R4,@TCNT0             ; Reset timer 0 counter
         ldi   @TIMVAL,R7            ; Load timer control value
         sti   R7,@TGCR0             ; Start timer 0
         ldi   6h,IOF                ; Pull AIC out of reset
         or    2000h,ST              ; Global interrupt enable
         ldi   PRIMARY,R3            ; Primary comm value
         ldi   B_REG,R0              ; Secondary comm value
         call  XMIT                  ; Set TB & RB
         ldi   A_REG,R0              ; Secondary comm value
         call  XMIT                  ; Set TA & RA
         ldi   C_REG,R0              ; Secondary comm value
         call  XMIT                  ; Send control word
;-------------------------------------------------------------------------
         ldi   @SIGNMSK,AR2          ; Put the SIGN MaSK into AR2
         ldi   @BFRPTR,AR1           ; Put 1st location of circ. data (x) buffer in AR1
         or    20h,IE                ; Enable RINT interrupts
;-------------------------------------------------------------------------
LOOP     idle                        ; Wait while sampling period passes
         b     LOOP                  ; Interrupts will trigger execution at RINT
;-------------------------------------------------------------------------
XMIT
         or    10h,IE
         idle
         sti   R3,@DTX0              ; Start control instruction
         idle
         sti   R0,@DTX0              ; Secondary transmit
         idle
         sti   R4,@DTX0              ; Transmit a 0
         xor   10h,IE
         rets
;-------------------------------------------------------------------------
```

```
SINIT0  .word 0xce970300              ; Enable RINT & 16-bit transfers
SINIT1  .word 0x111                   ; Configure as serial port pins
TIMVAL  .word 0x3c1                   ; Timer global control register value
SIGNMSK .word 0x80000000              ; SIGN MaSK for manipulating MSB before outputting
HPTR    .word FIRSEQ                  ; Location of filter sequence
BFRPTR  .word DTABFR                  ; Location of circ. data buffer
;-------------------------------------------------------------------------------
;  Convolution Equation:  y(n)= h(0)*x(n) + h(1)*x(n-1) + ...
;                                                          ... + h(N-1)*x(n-(N-1))
;                                       where h(i)= hBP(i)*w(i)     (Filter and Window)
;  Calling Sequence:
;
;  LOAD AR0       ; ADDRESS OF h(i)
;  LOAD AR1       ; ADDRESS OF x(n-(i))
;  LOAD RC        ; LENGTH OF FILTER MINUS 1, i.e. (N-1)
;  LOAD BK        ; LENGTH OF FILTER, i.e. N
;
;  REGISTERS USED FOR ACCESSING x(): AR0, AR1, RC, BK
;  REGISTERS MODIFIED BY RINT:  R0, R2, AR0, AR1, AR2, RC, BK
;  REGISTER CONTAINING RESULTING y():  R0
;-------------------------------------------------------------------------------
N       .set 32                       ; Order of the FIR filter
RINT    ldi  N,BK                     ; Set BK (Block) for this interrupt routine
        ldi  N-2,RC                   ; Set RC (Repeat Counter)
        ldi  @DRX0,R0                 ; Get received data into 32-bit R0.  14-bit data,
                                      ;      trailed by 2 "zero" bits, is in lower 16 bits
        ash  16,R0                    ; Shift left so sign bit is in MSB
        ash  -16,R0                   ; Shift right again to get 2's-compliment
        float R0,R2                   ; Convert the newest digital sample to Floating
Point
        stf  R2,*AR1++(1)%            ; Store data (x) in circ. buffer of size BK
        ldi  @HPTR,AR0                ; Reset coef. (h) table address register to h(0)
        ldf  0.0,R2                   ; Initialize R2 for accumulation of terms
        mpyf3 *AR0++(1),*AR1++(1)%,R0  ; h(0)*x(n-(0))  ->R0
                                      ; Next, repeat multiplication and accumulate
                                      ;      for each term in convolution equation
        rpts RC                       ; Repeats parallel instruction (RC+1) times
                                      ;      1<=i<=(N-1)
        mpyf3 *AR0++(1),*AR1++(1)%,R0  ; h(i)*x(n-(i))  ->R0
    ||  addf3 R0,R2,R2                ; Parallel Multiply and Add Operation
        addf R0,R2                    ; Add last term + others already accumulated
        fix  R2,R0                    ; Change floating point to integer
        and3 AR2,R0,R2                ; Use a 32-bit mask to recover the sign bit
        lsh  -16,R2                   ; Shift the sign bit down into the lower 16-bits
        and  0x00007ffc,R0            ; Use a mask to recover the relevant 14-bits of data
        or   R2,R0                    ; Overlay the recovered sign and data bits
        sti  R0,@DTX0                 ; Send result to D/A for output
        reti
;-------------------------------------------------------------------------------
        .brstart "fircoef",32         ; Create coef at 32 word boundary (16 if N=16, 64 if
N=64)

                                      ; Coefficients for filter sequence, h(i)
                                      ;      0<=i<=(N-1)
FIRSEQ  .float 0.0, 0.0, 0.0, 0.0
        .float 0.0, 0.0, 0.0, 0.0
        .float 0.0, 0.0, 0.0, 0.0
        .float 0.0, 0.0, 0.0, 0.0
        .float 0.0, 0.0, 0.0, 0.0
        .float 0.0, 0.0, 0.0, 0.0
        .float 0.0, 0.0, 0.0, 0.0
        .float 0.0, 0.0, 0.0, 0.0
;-------------------------------------------------------------------------------
        .brstart "circbuf",32         ; Initiate circ. data buffer at 32 word boundary
DTABFR                                ;      (16 if N=16, 64 if N=64)
        .loop  N                      ; x(n-(i))     0<=i<=(N-1)
        .float 0.0                    ; Have the Assembler place "N" 0's at locations
        .endloop                      ;      following x, as initial data buffer values
;-------------------------------------------------------------------------------
        .sect  ".servect"
        reti                          ; XINT0
        b RINT                        ; RINT0
```