Karlsruhe Institute of Technology

High-Accuracy Frequency, Phase and Amplitude Estimation for Robust Speech Recognition

Diplomarbeit
von

Ralf Huber

Institut für Anthropomatik
Fakultät für Informatik

Betreuer:	Prof. Dr. rer. nat. A. Waibel
	Prof. R. Stern, Ph.D.
	Dipl.-Inform. F. Kraft

Bearbeitungszeit: 16. Februar 2012 – 15. August 2012

Ralf Huber

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 15. August 2012

Zusammenfassung

In der vorliegenden Arbeit geht es darum, präzise Mess- bzw. Schätzverfahren für die Bestimmung von Amplituden, Phasen und Frequenzen in Sprachaufnahmen zu untersuchen. Insbesondere soll untersucht werden, ob es mit den Verfahren möglich ist, Raumhall aus Sprachaufnahmen zu entfernen.

In der vorangegangenen Studienarbeit [Huber11] wurde ein Verfahren vorgestellt, dass mittels Phasenregelkreisen (Phase-Locked Loops) den Amplituden- und Phasenverlauf in Aufnahmen von Sinustönen mit konstanter Frequenz bestimmt. Die so berechneten Werte konnten verwendet werden, um Hall von den Aufnahmen zu entfernen.

Im Zuge dieser Arbeit wurde zunächst versucht, das Verfahren auch für Sinustöne mit variabler Frequenz anzuwenden. Da stimmhafte Sprachaufnahmen als Überlagerung mehrerer harmonischer Frequenzen modelliert werden können [McAulay86], wäre die Entfernung des Halls einer einzelnen Frequenz ein erster Schritt zur Hallreduktion von echten Sprachaufnahmen.

Es zeichnete sich ab, dass das Vorhaben nicht den gewünschten Erfolg verspricht und daher wurde anschließend versucht, mit anderen Technologien eine bessere Frequenz-, Amplituden- oder Phasenschätzung zu erzielen. Zur Frequenzschätzung wurde ein Frequenzregelkreis (Frequency-Locked Loop, FLL) ähnlich dem in [Kumaresan12] vorgestellten in Matlab und Simulink umgesetzt. Als Ersatz für die PLL-basierte Amplituden- und Phasenschätzung wurde das APES-Verfahren (Amplitude and Phase Estimation of a Sinusoid [Li96]) in Matlab implementiert und gegenüber dem Originalalgorithmus so modifiziert, dass eine kontinuierliche Schätzung von Sinustönen mit variabler Frequenz möglich ist.

In den abschließenden Experimenten sollten die harmonischen Frequenzen stimmhafter menschlicher Sprache mit den Frequenzregelkreisen bestimmt werden, um im Anschluss die Amplituden- und Phasenverläufe dieser Frequenzen mit APES möglichst exakt zu bestimmen. Mit diesen Informationen wurde dann die stimmhafte Sprache wieder synthetisch erzeugt, um sie dann als Eingabe für die Spracherkennungssoftware JANUS zu benutzen. In einem weiteren Versuch wurde die Frequenzinformation genutzt, um die stimmhaften Sprachanteile innerhalb eines aufgenommenen Satzes mit einem Kammfilter so zu filtern, dass die vorkommenden harmonischen Frequenzen verstärkt und die anderen Frequenzanteile abgeschwächt werden. Die so veränderten Aufnahmen wurden anschließend erneut als Eingabe für das JANUS-Spracherkennersystem verwendet.

Leider ermöglichte keine der untersuchten Methoden die erwünschte Senkung der Wortfehlerrate bei automatischer Spracherkennung unter Umgebungsbedingungen mit Hall.

Contents

1. Introduction

1.1 Motivation

In these days speech recognition systems are built into an increasing amount of devices and they are used by more and more people every day. Some years ago, speech recognition technology could only be used in very controlled environments, such as to select a menu option in phone systems of call-centers or as dictation software for use with headsets. Today, the applications are much broader and less specific, for example in Apple's SIRI command system for the iPhone or in similar systems for other mobile phones such as Android-based phones.

These systems allow the user to speak to the phone in order to initiate a call, create an appointment in the calendar or to search for a term on the internet. Most of these systems also come with a software that tries to guess what the user actually wants to do and as a result, it seems that the phone is actually understanding the user instead of simply reacting to a predefined command. Cars and GPS devices are another field where speech technology is more and more common, for instance to tell the car or GPS device where one wants to go.

Finally, game consoles and TV sets increasingly also come with speech recognition software, such as Microsoft's KINECT, which includes a microphone array and powerful beamforming technology for far-distant speech recognition. Indeed, it seems that microphone arrays are one of the most used approaches to counter adverse effects such as reverb and noise in far-distant speech recognition settings. As everybody can confirm on his or her own, the human hearing system is very good even when the environmental circumstances are bad and the sound that arrives at the ear is distorted.

Besides far-distant speech recognition, there are other fields where the human ear is superior to today's technology, for example when it comes to speaker separation. Speaker separation is also related to the more general problem that is the identification of speech in environments with structural "noise". This includes not only colored noise with the same spectral shape all of the time, but also noise whose spectrum changes over time, such as background music.

This work is supposed to be a contribution to the search for signal processing methods, which might one day enable the use of speech recognition software or improve its performance in situations where current systems fail or have trouble to understand their user.

1.2 Objective

It has been stated that there are several problems which arise in real-world automatic speech recognition (ASR) settings. The focus of this work is mainly on reverberation and therefore, the final evaluation was also done on reverberant data. The goal was to find an algorithm which removes or reduces the effect of reverberation in speech recordings.

Current ASR front-ends mostly utilize a Fourier transform to obtain the power spectrum of the speech input. Further signal processing is used afterwards to improve the signal-to-noise ratio before features are calculated, which are then used for the actual recognition task. The use of the power spectrum implies that the phase information is lost. Interestingly, research indicates that phase information could be used to improve ASR performance, although some of the phase information seems to be somehow included in standard MFCC features [Saratxaga10] [Saratxaga09]. It seemed therefore an interesting question how the phase of the frequency components of speech signals could be tracked and if it could be used to improve an ASR system in reverberant conditions.

Phase-locked loops are a technology which allows to track the phase of a sinusoid. They have already been used for speech recognition tasks, for example for pitch tracking [Pelle03] or as a ASR system front-end [Estienne01]. In the predecessor to this work [Huber11], phase-locked-loops have already been successfully used to remove reverberation from single sinusoids played back in a room. The first goal of this work was to examine if a similar technique could be used to dereverberate more complex signals, such as sinusoids with time-varying frequency or harmonic complexes.

The second main goal of this work was to find methods that allow to track speech signal parameters, such as the fundamental frequency of voiced speech (pitch tracking) and the amplitude of individual frequency components in harmonic speech segments. Frequency-locked loops seemed to be a promising method for pitch tracking [Kumaresan12], which is why a similar system like the one presented in that paper was used in this work, too. The final experiments of this work were an examination of whether the so-obtained pitch information could be used for an improved tracking of the amplitudes of individual frequency components of voiced speech.

1.3 Outline

The following chapter explains the problem of reverb in automatic speech recognition before stating some methods which are currently used to improve reverberant speech recognition.

Chapter 3 contains detailed information about all the technologies that have been used for this work, such as phase-locked loops, frequency-locked loops, gammatone and comb-filters, as well as the APES amplitude and phase estimation algorithm.

Chapter 4 contains descriptions of the evaluation experiments which have been carried out to test the methods from chapter 3.

The final pages of this document contain a summary and rating of the experimental results and information about possible fields of further research.

2. Related Work

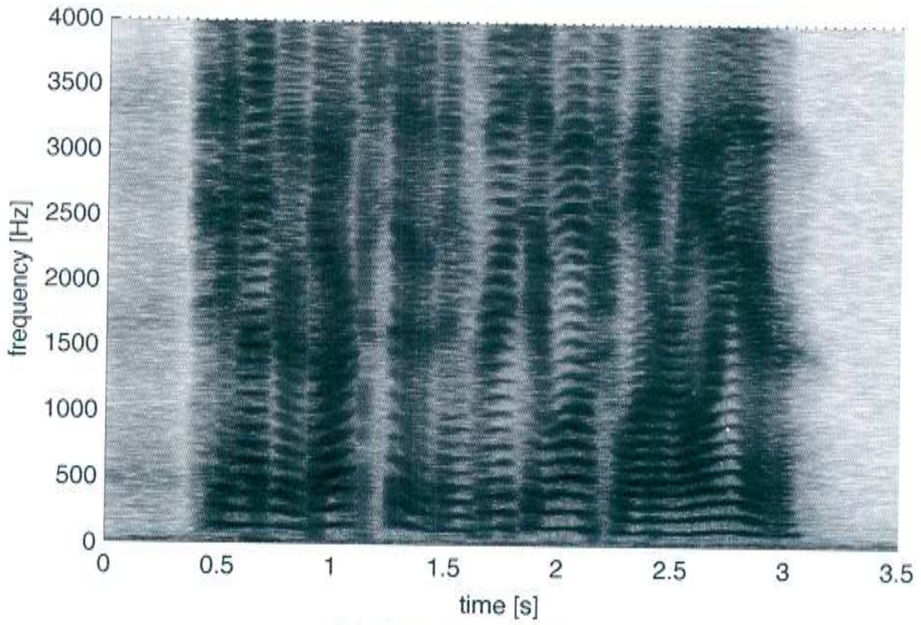
2.1 The Problem of Far-Distant Speech Recognition

Far-distant speech recognition is a difficult problem because of several aspects. The most simple mathematical explanation of reverb alone is $s_R(t) = s_D(t) \otimes h(t)$, where $\otimes$ denotes convolution, s_D is the undistorted direct sound, h is the impulse response that characterizes the filtering effect of the room and s_R is the resulting reverberated speech.

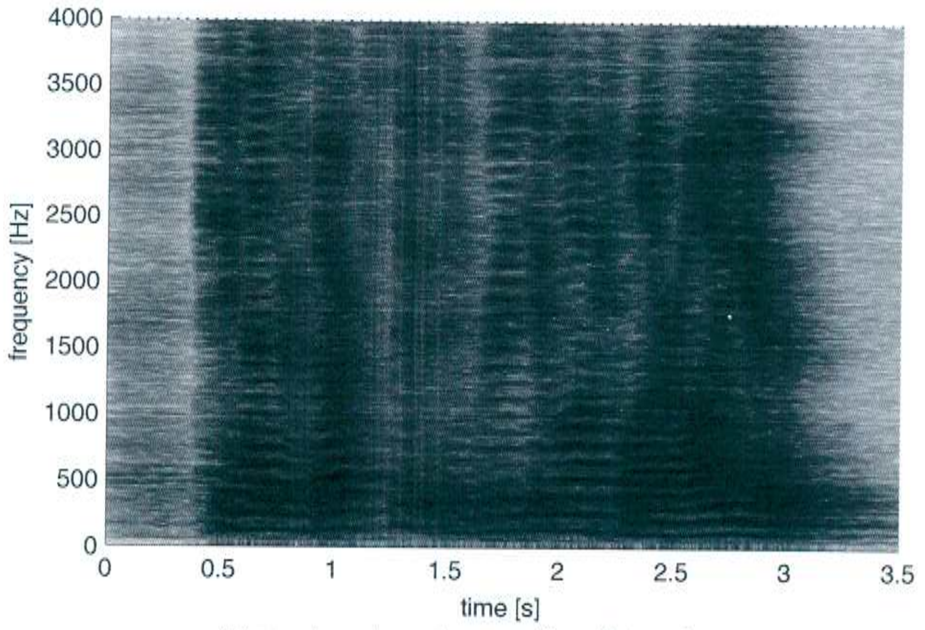
Unfortunately, this description does not include a variety of other effects, such as:

- Noise with a possibly time-varying spectral shape
- Changes of the impulse response due to movement of the speaker or the microphone in the room
- Changes of the impulse response due to changes in the room, such as opening a window

Figure 2.1 depicts a spectrogram of the sentence "She had your dark suit in greasy washwater all year.". The upper image is the clean spectrogram and the lower image the reverberated spectrogram with artificial reverb that corresponds to a recording distance of 2 m in an office room. It can be seen that the frequency components are much less obvious in the reverberant example than in the clean speech example. Basically, the effect of reverberation is a somewhat "smeared" spectrogram. This stems from the fact that a single reflected wavefront, which arrives at the microphone, is simply a delayed and scaled copy of the direct sound. These delayed copies of the sound interfere with the common HMM-based modeling technique used in ASR systems. Modeling speech with HMMs is based on the assumption that there is a transition to another HMM state when a certain part of the speech is over. Since the amount of reflections is generally unknown in a real environment, it is also unknown how often the HMM could possibly go back to an earlier HMM state to model an echo of the current utterance.



(a) clean spectrogram



(b) reverberant spectrogram (2 m distance)

Figure 2.1: Spectrograms of the sentence "She had your dark suit in greasy wash-water all year."

Another problem is that the acoustic model of an ASR system defined by the codebooks and gaussian mixture models does not reflect the reverberant speech: A close-talk ASR setup using a headset is somewhat generic in that there is very little noise and reverb. Therefore, the training conditions can be made to be similar to the later usage environment. The effects of reverb, however, differ vastly from room to room and there cannot be a generic "reverb" training condition which can be used for all possible reverberated environments. This leads to the assumption that it could be beneficial to be able to precisely track the information that is contained in a speech signal in order to better adapt to specific environments.

2.2 Joint Particle Filter Framework

Matthias Wölfel explains in his dissertation [Wölfel09] that additive distortions, like noise, and convolutive distortions, like reverb, influence each other and they should therefore be considered and modeled jointly instead of separately. It should be expected that a system which models noise and reverb in a single model and which also deals with both problems at once yields better far-distant speech recognition results than a system which uses separate noise-reduction and dereverberation algorithms.

A so-called joint particle filter framework is therefore presented in [Wölfel09] in order to deal with convolutive and additive distortions simultaneously. Furthermore, the suggested system was designed to be able to handle non-stationary distortions.

Wölfel also states that approaches which try to deal with distorted speech by manipulating speech features in the feature domain only work well for non-stationary noise, but not for non-stationary reverberation. Therefore, it seems promising to find a means of dereverberation that works in the signal domain rather than the feature domain.

2.3 Multi-Step Linear Prediction

Multi-Step-Linear Prediction (MSLP) is another technique presented in [Wölfel09]. It works by estimating and subtracting reverberation based on the signal that is observed a certain amount of samples earlier. MSLP is introduced to remove late reflections, because it has been found that late reflections, i.e. reflections that occur after 50 ms, contribute the most to the degradation of ASR performance. Another observation was that apparently only the reverberations in a frequency band from 250 Hz to 2500 Hz have a meaningful negative effect on the speech recognition accuracy. Since MSLP primarily targets late reflections, it seems useful to develop a method which deals primarily with early reflections and which can be used in addition to MSLP. Section 4.3 contains an evaluation of an impulse response estimation method which is designed particularly for the estimation of early reflections.

2.4 Phase-Locked Loops

Phase-locked loops have been used in FM receivers for a long time [Best93] in order to demodulate the FM signal from the antenna. They are very useful when it comes to precisely measure the frequency and phase of the incoming radio signal. Because of that, they have also been used in the field of automatic speech recognition, for example for pitch tracking in [Pelle03] or as a whole ASR front-end in [Estienne01].

A single PLL can only track a single frequency. Harmonic speech segments consist - as the name implies - of many frequency components which are all integer multiples of a fundamental frequency. A bandpass filterbank is therefore used in [Pelle03] and [Estienne01] in order to split the whole speech spectrum into subbands that contain only one frequency each. One PLL operates on each of these bands and the result of each PLL is then aggregated in a way that further emphasizes the harmonic structure of the speech.

2.5 Frequency-Locked Loops

When it comes to pitch tracking phase-locked loops actually perform more than what is needed. Instead of simply determining the frequency of the target signal, they attempt to generate a coherent signal, i.e. one that has the same phase. While it was very good if a perfectly-locking PLL was available for speech purposes, it might actually suffice to obtain a signal which has the same frequency as the target signal without being coherent to it.

The task of frequency tracking can be performed by a so-called frequency-locked loop (FLL). FLLs work much in the same way as PLLs and it is explained in section 3.2 that an FLL can be interpreted as a PLL with an additional filter.

FLLs, like PLLs, have already been used for speech processing. A so-called harmonically coupled FLL is presented in [Wang94]. Instead of splitting the speech signal into frequency bands like in [Pelle03], the coupled FLL is built to track multiple harmonics directly. It uses the information that the only possible frequencies are integer multiples of the fundamental frequency in order to reject frequencies which don't fit into this harmonic relationship.

[Kumaresan11] and [Kumaresan12] are more recent publications about FLLs and speech processing. The authors return to the filterbank-based approach of splitting the speech signal and processing each band individually. Their findings (and those from this work) suggest that the choice and setup of the splitting filterbank is crucial to the success of the method.

Besides the splitting approach and the harmonic coupling approach, a third massively parallel approach was examined in this work. It features a very large amount of FLLs which operate on tightly spaced overlapping subbands. The subbands are chosen such that each harmonic frequency is tracked by more than one FLL and a clustering algorithm is applied afterwards to find frequencies which have been "found" by multiple FLLs at the same time. Frequency tracking examples of this method can be found in section 4.4.

2.6 Blind Dereverberation using Harmonic Filtering

It has been explained at the beginning of this chapter that reverb can be modeled as a convolution of the direct sound with a room impulse response (RIR). The RIR is typically unknown in real settings and there are some approaches by Nakatani and Miyoshi which try to estimate the effects of the RIR in order to reverse them [Nakatani03a] [Nakatani03b] [Kinoshita05a] [Kinoshita05b] [Nakatani07].

These methods are called "blind", because they don't use prior knowledge of the room layout or the position of the speaker or microphone.

The processing typically works in three steps. At first the fundamental frequency of the current speech segment is determined. The second step is to estimate the direct sound based on the fundamental frequency estimate. This is done by increasing the signal parts whose frequencies are an integer multiple of the fundamental frequency. Nakatani and Miyoshi call this type of processing "adaptive harmonic filtering". A similar method is tested in this work using comb filters for the harmonic filtering (c.f. section 4.5). The final step is to train an adaptive filter using the reverberant recording as an input and the direct sound estimate as adaptation target. This effectively calculates a dereverberation filter.

The drawback of this method is that it needs training data for the given reverberant situation, i.e. room layout, speaker and microphone position, which is not always available.

3. Methodology

The following sections of this chapter contain mathematical descriptions of various filters and control systems, which will later be used in chapter 4 in order to try to remove reverberation from speech signals.

The first two technologies are phase-locked loops and frequency-locked loops, which are conceptually very similar to each other. Both can be used to track the frequencies in a voiced signal.

In section 3.3 follows a description of gammatone filters, which are parametric band-pass filters modeled after the human auditory system. Section 3.4 will be about feedforward and feedback comb filters, which can be used to enhance the harmonic frequencies of voiced speech.

The chapter continues with a section about the so-called APES amplitude and phase estimation technique, which will later be used to estimate the amplitude of the voiced signal components in human speech.

3.1 Phase-Locked Loops

Phase-locked loops (PLLs) are control circuits which can "tune in" on a frequency which is dominatly present in a signal. They consist of an oscillator, which produces a pure sinusoid at a certain tuneable frequency. The second component is the so-called phase detector or phase discriminator, which is used to compare the phase of the oscillator's output with the phase of the incoming signal. The output of the phase detector is then used to adjust the frequency of the oscillator in order to minimize the phase difference. A loop filter (usually a low-pass filter) is inserted between the phase detector and the oscillator to reduce high-frequency noise at the detector's output.

The predecessor of this work [Huber11] already contains a detailed description of phase-locked loops, two different phase detector circuits and guidelines to design the loop filter. This section focuses on the so-called Carlosena-extension method introduced in [Carlosena07] that can be used to increase the PLL order, which - in turn - allows the tracking of more complex input signals. It should be pointed out that there are many books which cover various aspects of phase-locked loops, such as [Best93], [Stephens02] and [Gardner05]. Figure 3.1 shows a block diagram of a so-

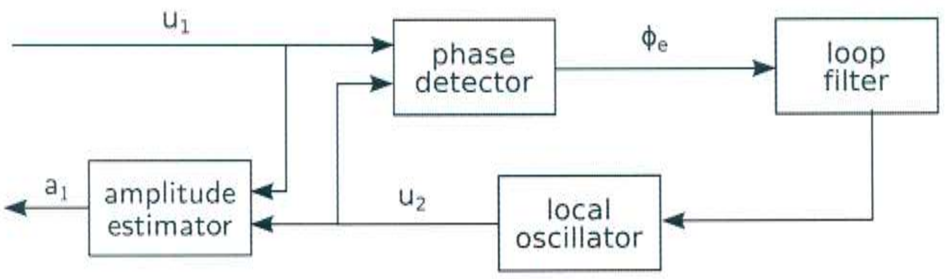


Figure 3.1: Block diagram of an extended phase-locked loop.

called extended PLL, as it was introduced in [Karimi-Ghartemani01]. Standard PLLs only consist of the phase detector, loop filter and local oscillator. The additional amplitude estimator is the reason why this particular setup was named "extended" PLL.

3.1.1 Phase Detector

For now it is enough to know that the local oscillator produces a sine wave with a certain frequency, which is determined by the input of the oscillator. The input u_1 to the PLL and the locally produced signal u_2 can therefore be described by:

$$u_1(t) = a_1 \cdot \sin(\omega_1 t + \phi_1) \tag{3.1}$$
$$u_2(t) = a_2 \cdot \sin(\omega_2 t + \phi_2) \tag{3.2}$$

Obviously, a_1 and a_2 are the amplitudes, ω_1 and ω_2 are the frequencies and ϕ_1 and ϕ_2 are the initial phase offsets of u_1 and u_2, respectively. The phase-locked loop is said to be "in lock" or simply "locked", when ω_2 is approximately equal to ω_1. Some further criteria about the stability of the locking also have to be fulfilled, but $\omega_1 \approx \omega_2$ is a necessary prerequisite for the locked state.

For the Hilbert-Transform phase detector [Best93], both u_1 and u_2 need to be processed by a Hilbert-transformer. A Hilbert-transform operation, denoted by $\mathcal{H}$, shifts all frequency components of a signal equally by $-\frac{\pi}{2}$ radians:

$$u_1(t) = a_1 \cdot \sin(\omega_1 t + \phi_1) \tag{3.3}$$
$$\overline{u_1}(t) := \mathcal{H}\{u_1(t)\} = a_1 \cdot \sin\left(\omega_1 t + \phi_1 - \frac{\pi}{2}\right) = -a_1 \cdot \cos(\omega_1 t + \phi_1) \tag{3.4}$$
$$u_2(t) = a_2 \cdot \sin(\omega_2 t + \phi_2) \tag{3.5}$$
$$\overline{u_2}(t) := \mathcal{H}\{u_2(t)\} = a_2 \cdot \sin\left(\omega_2 t + \phi_2 - \frac{\pi}{2}\right) = -a_2 \cdot \cos(\omega_2 t + \phi_2) \tag{3.6}$$

It is important to note that only $\overline{u_1}$ actually has to be calculated. The other Hilbert-transformed signal, $\overline{u_2}$ can be obtained from the local oscillator directly in addition to u_2 by using both the sin and the cos function.

For simplicity, it is now assumed that $\omega_1 \equiv \omega_2 \equiv \omega$, i.e. the loop is perfectly locked. Note that in this formulation, all parameters of the sine waves (a_i, ω_i, ϕ_i) are constants and independent of time. In reality, however, the frequency and amplitude of the signal are surely varying over time, so that in theory, this whole mathematical description is only valid for steady-state situations. Yet, practical experiments have shown that PLL systems work nevertheless, which is because the sine waves' parameters usually vary slowly compared to the reaction time of the loop. [Best93], for example, also contains a derivation of the loop behavior in the unlocked state. There is, however, no simple solution and further approximations and assumptions have to be made in order to obtain mathematical descriptions of the behavior of an unlocked loop.

After the Hilbert-transform of u_1, the abovementioned four signals are combined to form two intermediate signals, sig_1 and sig_2.

$$\begin{aligned} sig_1(t) &:= u_1(t) \cdot u_2(t) + \overline{u_1}(t) \cdot \overline{u_2}(t) && (3.7)\\ &= a_1 a_2 \cdot (\sin(\omega t + \phi_1) \cdot \sin(\omega t + \phi_2) + \cos(\omega t + \phi_1) \cdot \cos(\omega t + \phi_2)) && (3.8)\\ &= a_1 a_2 \cdot \cos(\omega t + \phi_1 - (\omega t + \phi_2)) && (3.9)\\ &= a_1 a_2 \cdot \cos(\phi_1 - \phi_2) && (3.10) \end{aligned}$$

$$\begin{aligned} sig_2(t) &:= u_1(t) \cdot \overline{u_2}(t) - \overline{u_1}(t) \cdot u_2(t) && (3.11)\\ &= -a_1 a_2 \cdot (\sin(\omega t + \phi_1) \cos(\omega t + \phi_2) - \cos(\omega t + \phi_1) \sin(\omega t + \phi_2)) && (3.12)\\ &= -a_1 a_2 \cdot \sin(\omega t + \phi_1 - (\omega t + \phi_2)) && (3.13)\\ &= -a_1 a_2 \cdot \sin(\phi_1 - \phi_2) && (3.14) \end{aligned}$$

$-sig_2$ is then divided by sig_1, which results in:

$$-\frac{sig_2(t)}{sig_1(t)} = -\frac{-a_1 a_2 \cdot \sin(\phi_1 - \phi_2)}{a_1 a_2 \cdot \cos(\phi_1 - \phi_2)} = \tan(\phi_1 - \phi_2) \qquad (3.15)$$

The phase error, $\phi_e := \phi_1 - \phi_2$ is supposed to be the output of the phase detector and it can easily be calculated using the inverse tangent function. Usual implementations of the inverse tangent function, for example $atan$ from Matlab, return values in the range of $[-\pi/2; \pi/2]$ radians. However, the four-quadrant inverse tangent function, which is denoted with $atan2$ in Matlab, can be used to obtain results in the full interval of $[-\pi; \pi]$ by supplying the numerator and denominator of equation 3.15 to the $atan2$ function individually.

3.1.2 Local Oscillator

Similar to the loop filter, the local oscillator of a digitally implemented phase-locked loop can be implemented in different ways, for instance as a numerically controlled oscillator (NCO) or as a voltage-controlled oscillator (VCO), which was the original design for analog phase-locked loops. Matlab/Simulink, for example, offers a "discrete-time VCO block", whose block diagram is depicted in figure 3.2. This is basically a sampled implementation of a continuous-time analog VCO, which uses an accumulator instead of an integrator.

The VCO output u_2 is supposed to be a sine wave whose instantaneous frequency (in Hz) is $\hat{f}(t) = f_0 + \kappa u_{in}(t)$. u_{in} represents the input of the oscillator, κ is the VCO's gain factor, T_s is the sampling time, f_0 the so-called quiescent frequency (in Hz) and z^{-1} represents a one-sample delay. The quiescent frequency is the frequency which

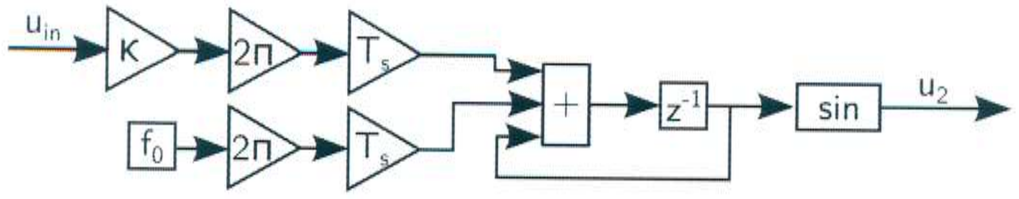


Figure 3.2: Simplified diagram of the Simulink discrete-time VCO block

the VCO output has in case the input u_{in} is 0. In theory, the quiescent frequency could be set to 0, because a well-designed PLL has the ability to lock onto any frequency when given enough time. However, the locking process takes longer the further the target frequency f_1 is away from the current VCO frequency f_2 [Best93]. Therefore, it is usually better to specify a quiescent frequency f_0 that is close to the expected target frequency f_1. This is not only advantageous for the initial lockin process, but also in the case of a re-lockin after the PLL has temporarily lost its lock.

To prove that the diagram in figure 3.2 actually produces a sine wave with the desired instantaneous frequency of $\hat{f}(t) = f_0 + \kappa u_{in}(t)$, it is best to start with the definition of "instantaneous" frequency, which is the derivative of a sinusoid's phase with respect to time:

$$\hat{f}(t) = \frac{1}{2\pi}\frac{\mathrm{d}\theta}{\mathrm{d}t} \tag{3.16}$$

Note that "phase" in this context means the whole argument of the sin or cos function, named θ, whereas "phase offset" refers to ϕ, which only determines the initial phase of a sinusoid for $t = 0$. The phase detector of a PLL actually compares the phases of its inputs, not the phase offsets. If, however, it is assumed that the PLL is locked, i.e. $\omega_1 = \omega_2$, then the calculation reduces to a comparison of ϕ_1 and ϕ_2. In a continuous-time VCO, u_2 would be calculated as follows:

$$u_2(t) = \sin\Bigg(\underbrace{2\pi\int_0^t \underbrace{f_0 + \kappa u_{in}(t)}_{\hat{f}_2(t)}\,\mathrm{d}t + \phi_0}_{=:\theta(t)}\Bigg) \tag{3.17}$$

It can be seen that the derivative of $\theta(t)$ in equation 3.17 with respect to time, divided by 2π is indeed $f_0 + \kappa u_{in}(t)$. The integral in equation 3.17 must be replaced by a sum to obtain a discrete system and the sum must furthermore be weighted by the sampling time T_s to normalize the sum to a given timeframe. Therefore, a discrete version of equation 3.17 is:

$$u_2[n] = \sin\left(2\pi T_s \sum_{m=0}^{n}(f_0 + \kappa u_{in}[m]) + \phi_0\right) \tag{3.18}$$

It can easily be verified that figure 3.2 is a block diagram representation of this equation, except for the additional ϕ_0 summand. ϕ_0 represents the initial phase offset of the VCO's output sinusoid. Similar to the VCO quiescent frequency, the initial phase offset could theoretically be set to any value, since the PLL will minimize the phase difference between the PLL input and the VCO output anyway. If the initial phase offset ϕ_1 of the input was known, ϕ_0 could be set to a nearby value to facilitate locking. Normally, however, ϕ_1 is unknown, so ϕ_0 is simply set to 0.

Next, the Z-domain transfer function of the VCO can be derived by starting with the difference equation of the accumulator in figure 3.2:

$$\text{accu}_{\text{out}}[n] = \text{accu}_{\text{out}}[n-1] + T_s \cdot \text{accu}_{\text{in}}[n-1] \tag{3.19}$$

Note that the sin function can be omitted here, because it is not important for the transfer function of the PLL as a whole. A PLL is actually a control loop which operates based on the phase of its input and the phase of the VCO output. The sin function in figure 3.2 is necessary in the implementation simply because the subsequent phase detector, which has been discussed in the previous section, expects an actual sinusoid as input instead of just the phase value. Ultimately, this leads to the following Z-domain transfer function:

$$\begin{aligned}
 & \Phi_2(z) = z^{-1}\Phi_2(z) + 2\pi\kappa T_s \cdot z^{-1}U_{in}(z) \\
\Leftrightarrow\quad & \Phi_2(z) - z^{-1}\Phi_2(z) = 2\pi\kappa T_s \cdot z^{-1}U_{in}(z) \\
\Leftrightarrow\quad & \Phi_2(z)(1 - z^{-1}) = 2\pi\kappa T_s \cdot z^{-1}U_{in}(z) \\
\Leftrightarrow\quad & F_{VCO}(z) := \frac{\Phi_2(z)}{U_{in}(z)} = 2\pi\kappa T_s \frac{z^{-1}}{1 - z^{-1}} \qquad (3.20)
\end{aligned}$$

3.1.3 PLL Transfer Function and 1st Order Active Lead-Lag Loop Filter

Now that the VCO transfer function is known, it can be used to calculate the transfer function of the PLL as a whole. Figure 3.3 depicts the Z-domain block diagram of a PLL.

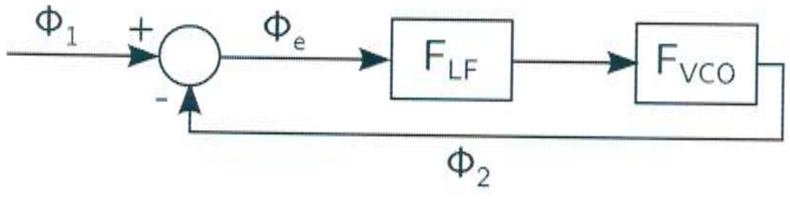


Figure 3.3: Z-domain block diagram of a PLL.

Again, it can be seen that the only values that are meaningful for a PLL are phase values. The Z-domain transfer function can now be obtained easily:

$$\begin{aligned}
\Phi_2(z) &= F_{VCO}(z) \cdot F_{LF}(z) \cdot \Phi_e(z) && (3.21)\\
&= F_{VCO}(z) \cdot F_{LF}(z) \cdot (\Phi_1(z) - \Phi_2(z)) && (3.22)\\
&= F_{VCO}(z) \cdot F_{LF}(z) \cdot \Phi_1(z) - F_{VCO}(z) \cdot F_{LF}(z) \cdot \Phi_2(z) && (3.23)
\end{aligned}$$

$$\Rightarrow \quad \Phi_2(z) + F_{VCO}(z) \cdot F_{LF}(z) \cdot \Phi_2(z) = F_{VCO}(z) \cdot F_{LF}(z) \cdot \Phi_1(z) \tag{3.24}$$

$$\Rightarrow \quad \Phi_2(z) \cdot (1 + F_{VCO}(z) \cdot F_{LF}(z)) = F_{VCO}(z) \cdot F_{LF}(z) \cdot \Phi_1(z) \tag{3.25}$$

In the standard form $F(z) = \frac{Output(z)}{Input(z)}$, this looks like:

$$F_{PLL}(z) := \frac{\Phi_2(z)}{\Phi_1(z)} = \frac{F_{VCO}(z) \cdot F_{LF}(z)}{1 + F_{VCO}(z) \cdot F_{LF}(z)} \tag{3.26}$$

If one were to insert the actual VCO transfer function (equation 3.20) into equation 3.26, there would be a z^{-1} term in both the numerator and the denominator. In terms of digital filter design, it is therefore possible to argue that the order of a phase-locked loop is at least one. The choice of loop filter and its transfer function determine the total order of a PLL. Since the VCO has an order of one, it can be seen from equation 3.26 that the order of a PLL is always equal to the order of the loop filter plus one.

According to [Best93], the order of a PLL is among the most important of its characteristics, because it determines which kind of phase variations of u_1 can be successfully tracked by the PLL without losing its lock. If the order of the loop filter (and therefore the order of the loop) is increased by one, the loop can track increasingly complex signals, for example a phase ramp (= frequency step) instead of a phase step, or a phase parabola (= frequency ramp) instead of a frequency step. Figures 3.4, 3.8 and 3.9 visualize this effect using PLLs of order 2, 3 and 4, respectively. Each of these figures contains 6 subfigures. The 3 subfigures on the left depict different input signals (phase step, frequency step, frequency ramp) and the 3 subfigures on the right show the corresponding PLL response, i.e. the value of the phase error ϕ_e for the given input.

The choice of loop filter is virtually unrestricted, as long as the resulting loop is stable. One of the most basic loop filters is the 1st order active lead-lag filter. It is originally an analog low-pass filter and according to [Stephens02], its S-domain transfer function is described by:

$$F_{LF}(s) = \kappa_{LF} \frac{1 + \tau_2 s}{1 + \tau_1 s} \tag{3.27}$$

The filter gain κ_{LF} corresponds to the DC-gain of the filter. It can be greater than one, which is why the filter is active. τ_1 and τ_2 are the filter's time constants, which determine the corner frequencies of the passband and stopband. The transfer function of the active first-order lead-lag filter looks like the one depicted in figure 3.5.

The Z-domain transfer function of the first-order lead-lag filter can be obtained for example by a bilinear Z-transform, using the substitution $s = \frac{2}{T_s} \frac{1-z^{-1}}{1+z^{-1}}$:

$$F_{LF}(z) = \kappa_{LF} \frac{T_s + 2\tau_2 + (T_s - 2\tau_2) z^{-1}}{T_s + 2\tau_1 + (T_s - 2\tau_1) z^{-1}} \tag{3.28}$$

It can be seen in figure 3.4 that the 2nd-order PLL is able to fully adapt to any phase steps, whereas ϕ_e shows a permanent error in the case of a frequency step or frequency ramp at the input. The error after a frequency step can be made infinitely small by increasing the loop filter gain κ_{LF}. The error after a frequency ramp, however, will always persist. It can only be reduced by higher-order loops which are able to track frequency ramps.

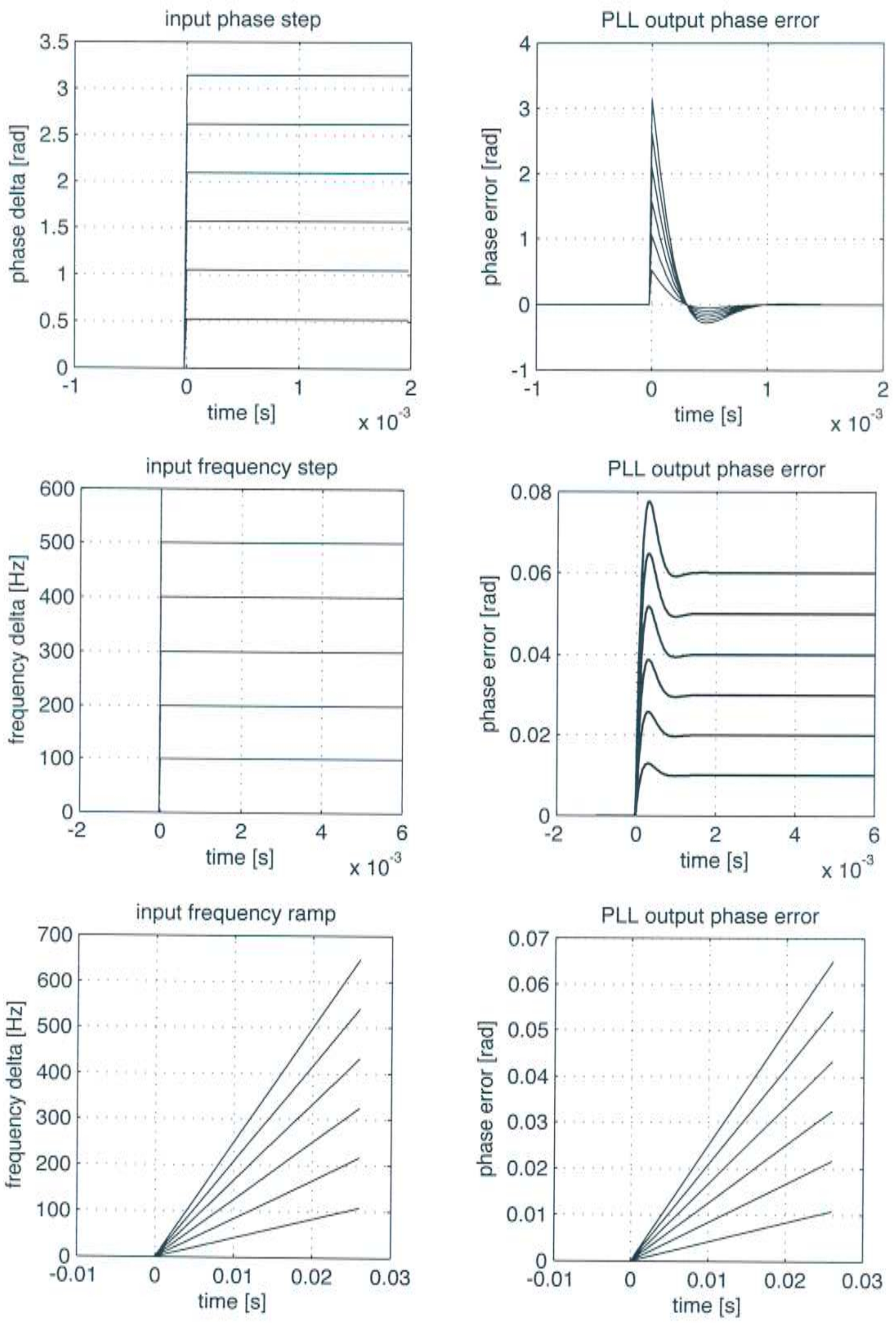


Figure 3.4: Tracking behavior of a 2nd-order PLL using an active lead-lag loop filter.

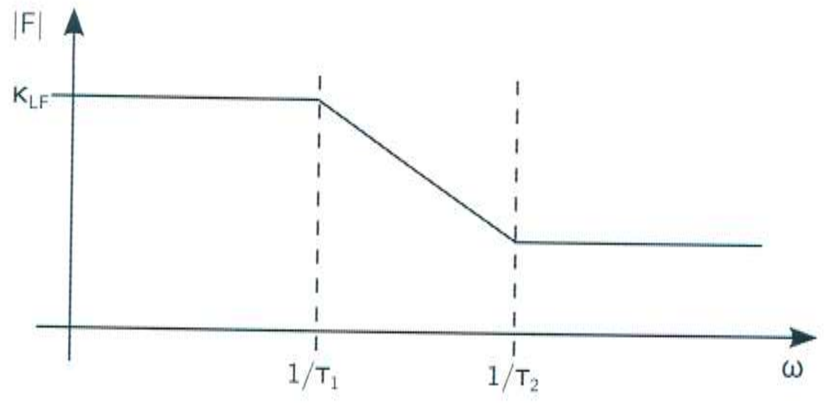


Figure 3.5: Frequency response of a first-order active lead-lag filter.

[Best93] contains various formulas to determine the parameter values of a second-order loop with the aforementioned loop filter. The loop parameters can, for example, be specified via the desired "lock-in time", which is the time the PLL needs to lock onto a newly appearing sinusoid in its input. Other specification methods include a specification of the necessary loop bandwidth. The loop bandwidth determines how much noise can be present at the PLL input without disturbing a locked PLL. An exhaustive coverage and mathematical derivation of all specification methods would be too much to repeat here, which is why it was omitted.

3.1.4 Carlosena-Style Order Extension

It has been said before that higher-order loops have the ability to track more complex input signals than lower-order loops, which is why it seems promising to increase the loop order for better tracking accuracy. One of the drawbacks of this practice is that while the tracking accuracy and stability can be improved with higher-order loops, they also suffer from increasingly slow tracking speed. This means that higher-order loops tend to be able to follow more complex input signals more precisely, but only as long as the changes in the input signal happen slowly enough.

Another important disadvantage of high loop orders is that it becomes increasingly difficult to specify the parameters for the loop filters. At the end of the previous section it was mentioned that formulas exist to precisely (and optimally) specify parameters for a PLL with some desired properties. However, exact formulas exist only for 2nd-order loops. In [Stephens02], the author even suggests to try out different parameter sets for high-order loops to find the one which fits the given application best.

In [Carlosena07] and [Carlosena08], the authors Carlosena and Mànuel-Lázaro suggest a new method with which the order of a PLL can be increased step-by-step. Each step increases the order by one and the additional filter coefficients are selected in such a way that their influence on the "lower-order behavior" of the loop is low, while improving the "high-order behavior" of the loop at the same time.

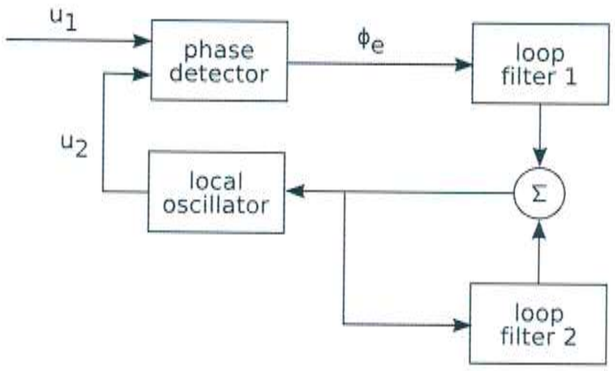
A normal second-order loop, like it has been presented in the previous section, builds the foundation for the Carlosena-style PLL order extension. For each extension step, an additional passive low-pass filter is built into the signal path between the VCO and the loop filters which already exist. The extension filter has the following transfer functions (S-domain and bilinear transformed Z-domain):

$$F_{ext}(s) = \frac{1}{1+\tau s} \tag{3.29}$$

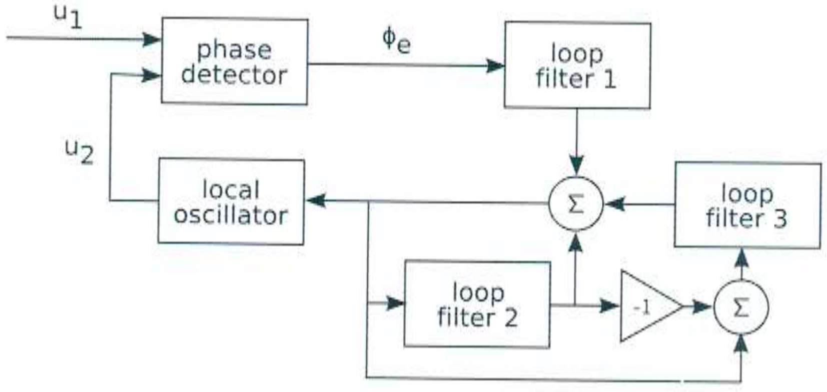
$$F_{ext}(z) = \frac{T_s + T_s z^{-1}}{T_s + 2\tau + (T_s - 2\tau)z^{-1}} \tag{3.30}$$

The additional loop filter time constant τ will be named τ_3 for the 2nd-to-3rd-order extension filter and it will be named τ_4 for the extension from 3rd to 4th order. Remember that τ_1 and τ_2 are already in use for the original 1st-order active lead-lag loop filter.

Figure 3.6 shows the new layout for the 3rd and 4th-order PLLs using one and two Carlosena extensions, respectively. In both cases, loop filter 1 is the original loop's loop filter. Loop filter 2 is the first extension for a total loop order of 3 and loop filter 3 is the second extension filter. Both loop filter 2 and 3 are of the same internal structure (equation 3.30), but with different additional time constants.



(a) 3rd-order loop block diagram



(b) 4th-order loop block diagram

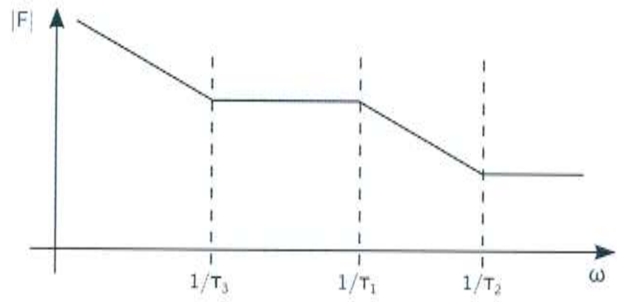
Figure 3.6: Block diagrams of Carlosena-extended PLLs

All loop filters can mathematically be combined to form a single high-order loop filter. According to [Carlosena07], the S-domain transfer function or the 3rd and 4th-order PLLs in figure 3.6 turn out to be:

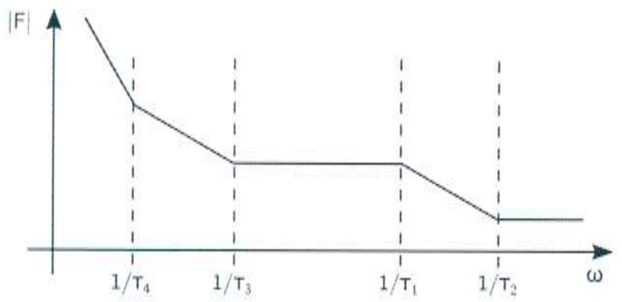
$$F_{3rd}(s) = \kappa_{LF}\frac{1}{\tau_3 s}\frac{(1+\tau_2 s)(1+\tau_3 s)}{1+\tau_1 s} \tag{3.31}$$

$$F_{4th}(s) = \kappa_{LF}\frac{1}{\tau_3 \tau_4 s^2}\frac{(1+\tau_2 s)(1+\tau_3 s)(1+\tau_4 s)}{1+\tau_1 s} \tag{3.32}$$

The Z-domain transfer functions can be obtained by the usual bilinear Z-transform $s = \frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}}$, but the equations turn out to be quite lengthy, which is why they have been omitted here. Additionally, Matlab provides the "c2d()" method, which can be used with 'tustin' as third argument in order to calculate the bilinear Z-transform automatically.



(a) Combined 2nd-order filter frequency response



(b) Combined 3rd-order filter frequency response

Figure 3.7: Frequency responses of Carlosena-extended loop filters

Figure 3.7 shows the frequeny responses of these Carlosena-extended 3rd and 4th-order PLLs. It can be seen that compared to figure 3.5, the transfer function contains additional bends at lower frequencies. Each bend in the transfer function is associated with one of the filter time constants τ_i.

It has been said at the end of the previous section that equations can be found in the PLL-literature which help to specify the timing constants τ_1 and τ_2 of 2nd-order PLLs (1st-order loop filters). The additional timing constants τ_3 and τ_4 for the Carlosena-extension filters must be carefully chosen such that they don't interfere with the behavior of the loop that they are extending. In [Carlosena07], the authors suggest to make any additional filter time constant 5 times higher than the previously highest constant. This ensures that the bends in the filter frequency response are spaced sufficiently apart from each other.

The 3rd and 4th-order PLLs designed with this guideline in mind have been simulated similar to the 2nd-order loop in figure 3.4 and the results are depicted in figure 3.8 and 3.9. It can be seen that, as expected, the 3rd-order loop is able to reduce the phase error ϕ_e to 0 after a frequency step at the input. The 4th-order PLL can even track a frequency ramp. The drawback of the additional loop filters can be seen for instance at the top right subfigures, which contain the PLL response to a phase step. Here, the 3rd and 4th-order PLLs take substantially more time before the phase error reaches 0, compared to the 2nd-order loop.

3.2 Frequency-Locked Loops

It is shown in [Carlosena07], that the 3rd-order PLL depicted in figure 3.6 can be rearranged by not feeding the input of the VCO into the second loop filter, but instead feeding the derivative of its output into the second loop filter. At this point it is important to remember that the output of the VCO that actually matters in a PLL is the phase θ_2 of the VCO output signal. The sine function that is used to calculate the actual VCO output $u_2(t) = \sin(\theta_2(t))$ is only needed because the subsequent phase detector expects a sine wave at its input instead of a simple phase value. Figure 3.10 depicts the resulting rearranged loop.

Taking the derivative of θ_2 at a certain moment in time is equivalent to calculating the instantaneous frequency $\hat{f}_2$ of the VCO output at that very moment (c.f. equation 3.16, page 14). Therefore, the input to the second loop filter is actually the instantaneous frequency. Calculating the derivative of θ_2 simply counteracts the integration that takes place in the VCO.

As a result, it is obvious that the loop does not only act based on the in- and output phase, but also based on the output frequency. In order to track the frequency components of a speech signal, it is not so important to obtain a tracking result in which each output sine wave is exactly synchronous (meaning: in phase) to the corresponding input sine wave. Reflected wavefronts and other environmental influences are among the reasons why the phase offset (and therefore the total phase) of a recorded sinusoid changes very rapidly over time [Huber11]. Furthermore, human listening experiments have shown that the human ear is insensitive to the initial phase offset of harmonic components in human speech [Carlyon97].

Therefore, it seems reasonable to dismiss a perfect in-phase signal tracking for a signal tracking which only focuses on the target signal's frequency. These techniques are known as frequency-locked loops (FLLs). In general, FLLs are similar to PLLs. They both consist of a loop filter, a local oscillator and a discriminator. The difference is that while a PLL phase detector calculates the phase difference between two signals, an FLL frequency detector calculates the frequency difference.

FLLs are widely used, for example in GPS devices, which is why a lot of information about FLLs can be found in GPS literature, such as [Kaplan06] or [Curran12]. [Costas80] is another paper that deals with frequency-locked loops.

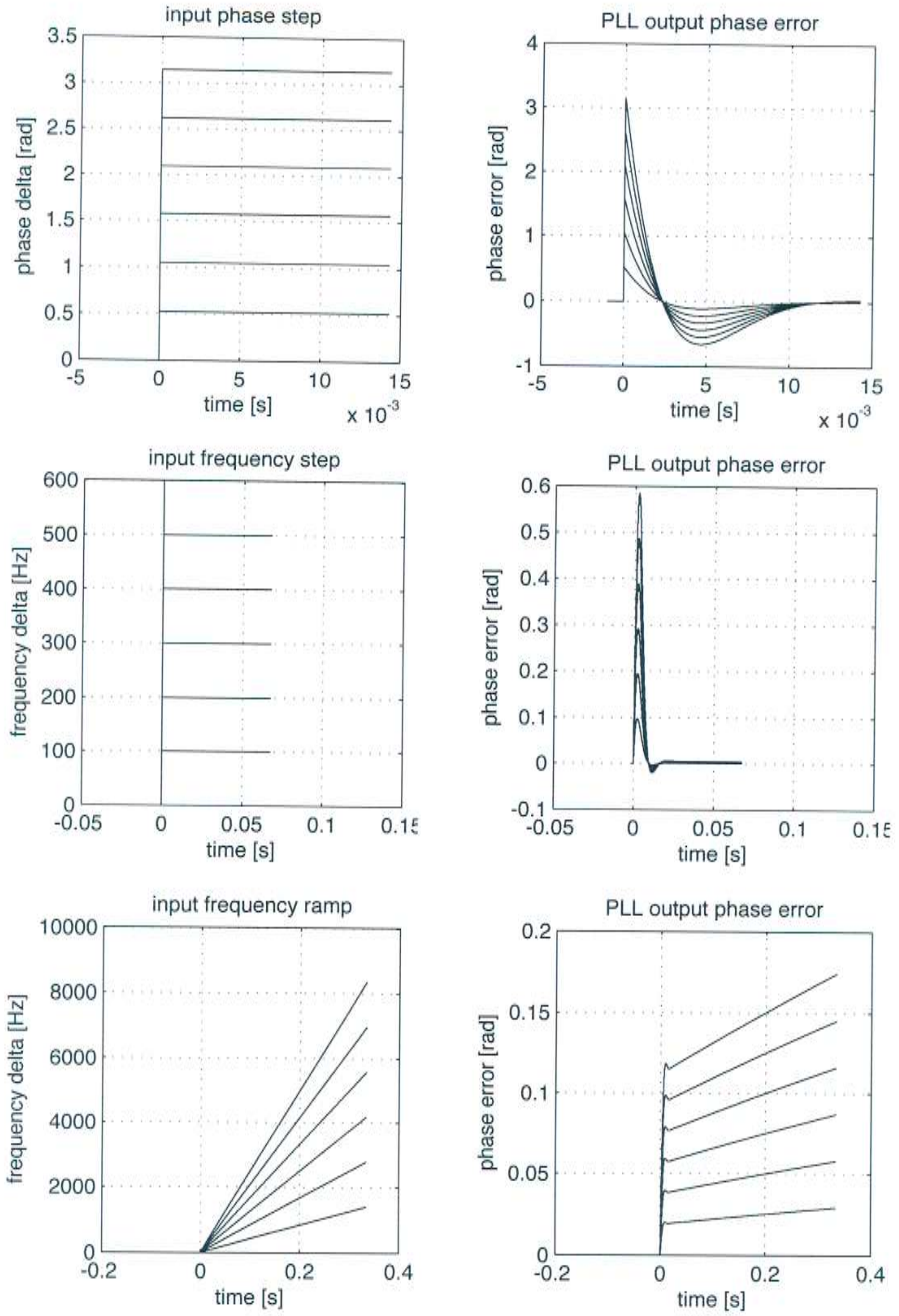


Figure 3.8: Tracking behavior of a 3rd-order PLL using an active lead-lag loop filter and one Carlosena extension.

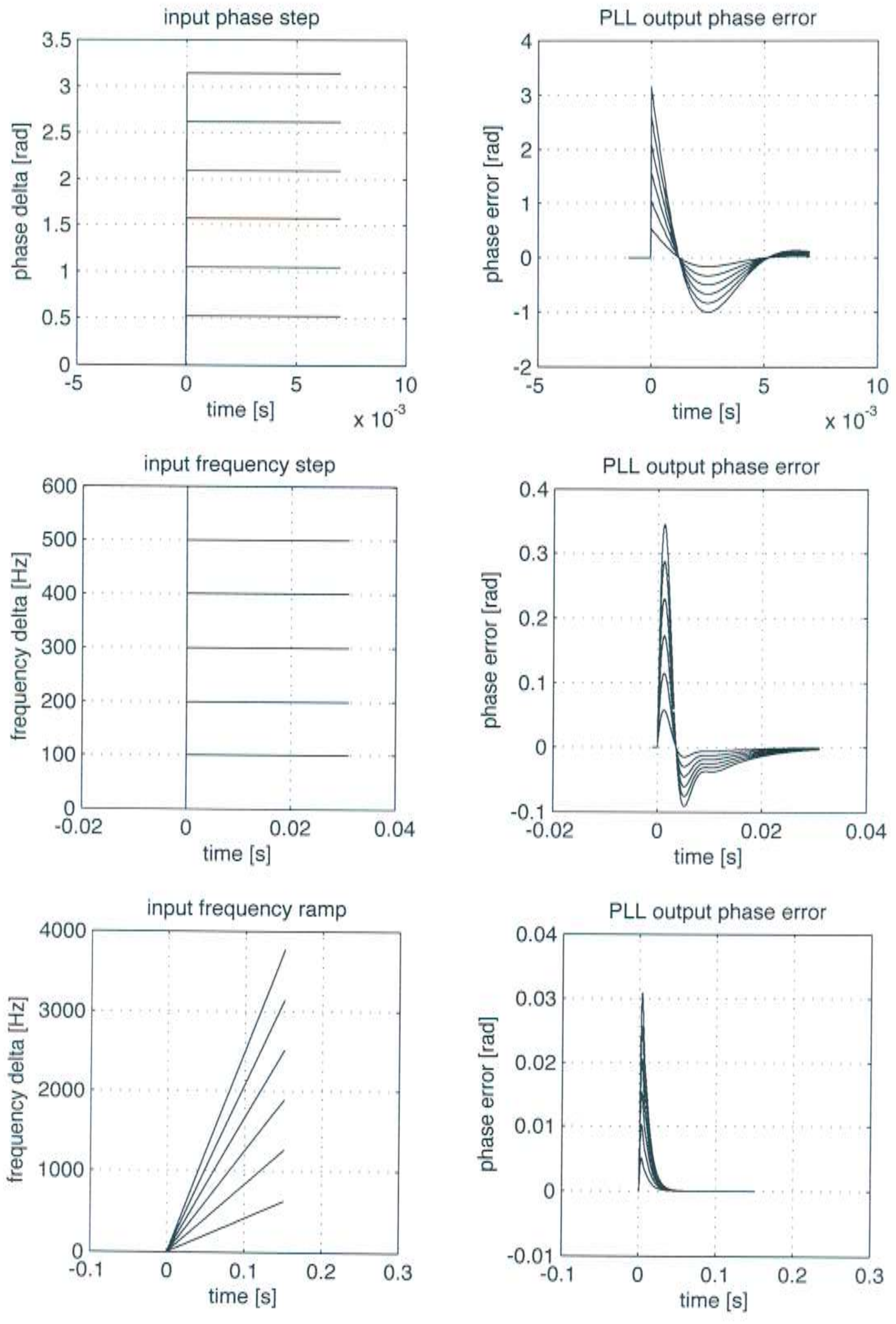


Figure 3.9: Tracking behavior of a 4th-order PLL using an active lead-lag loop filter and two Carlosena extensions.

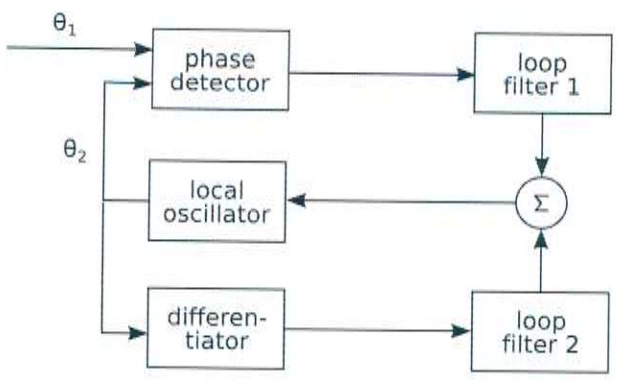


Figure 3.10: Equivalent schematic to the 3rd-order Carlosena extended PLL depicted in figure 3.6

3.2.1 Frequency Detector

In the same way as there are many different phase detector circuits available for PLLs, there are also various frequency detector algorithms for FLLs. [Natali84] is an early paper dealing with FLLs and it describes the so-called differentiator detector, the cross-product detector, the cross-dot-product detector and a comparison with fourier transform. Further publications like [Curran12] and [Tiwari11] also include the so-called four-quadrant arctangent discriminator, which appears to have superior performance.

However, neither of these frequency detectors were designed with speech applications in mind. In [Kumaresan11] and [Kumaresan12], the authors Kumaresan, Peddinti, and Cariani present a frequency-locked loop setup which has already been designed and tested for the task of pitch estimation. Therefore, this seemed like a reasonable approach to examine.

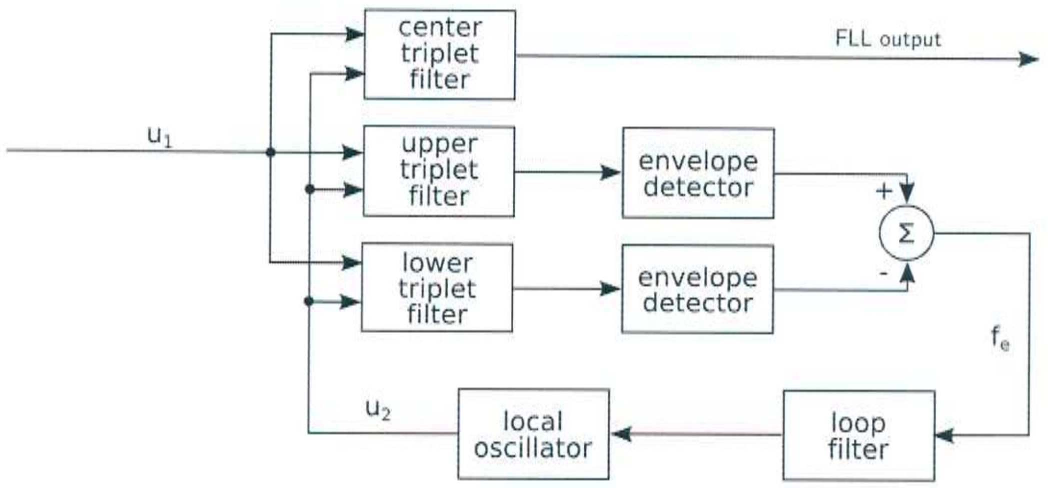


Figure 3.11: Schematic of a triplet-filter frequency-locked loop.

Figure 3.11 shows the basic layout of the system. As expected, it has a loop filter and a local oscillator. The frequency detector, however, is made up of a so-called filter triplet, two envelope detectors and a difference block. In this context, a filter triplet is a combination of three bandpass filters with slightly different passband center frequencies, as depicted in figure 3.12. All three filters of the filter triplet are based on the same basic frequency response, which is the center triplet filter. The lower and upper filter are created by slighly shifting the center filter up and down along the frequency axis.

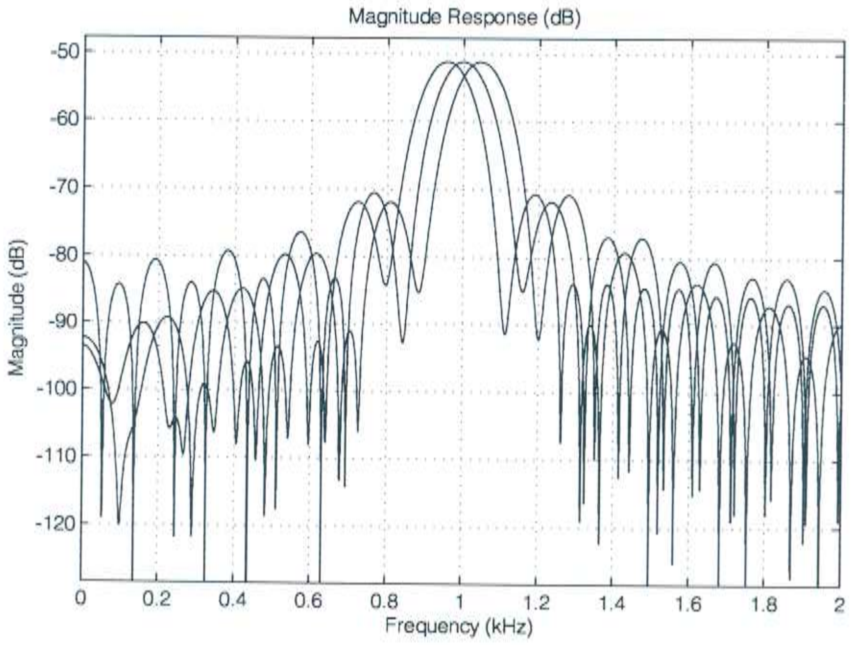


Figure 3.12: Example of a filter triplet centered at 1 kHz

The working principle of the Triplet Filter FLL is that the incoming signal contains a certain frequency and noise, which can have a high energy at other frequencies. If it is assumed that the FLL is locked onto a frequency f, then the center triplet filter is placed such that the center of its passband is also at f. The output of the center triplet filter is then considered the FLL output, since only the spectral components around f can pass the filter.

The output of the lower and upper triplet filters is then processed in an envelope detector, whose output roughly correlates with the energy in the corresponding signal. Afterwards, the difference block is used to subtract the energy of the lower triplet filter output from the upper triplet filter output. If the input signal's instantaneous frequency increases, the energy of the upper triplet filter output will be greater than the energy of the lower triplet filter output. As a result, the difference of both energy values is positive and the local oscillator would therefore increase its frequency.

At first, it seems that this kind of frequency detector is computationally very extensive. After all, the input signal has to be processed in three filters and furthermore, the coefficients of these filters have to be adjusted constantly in order to move the passband center to the correct frequency. It turns out, however, that some non-time-varying filters can be combined to create a frequency response like the one depicted in figure 3.12.

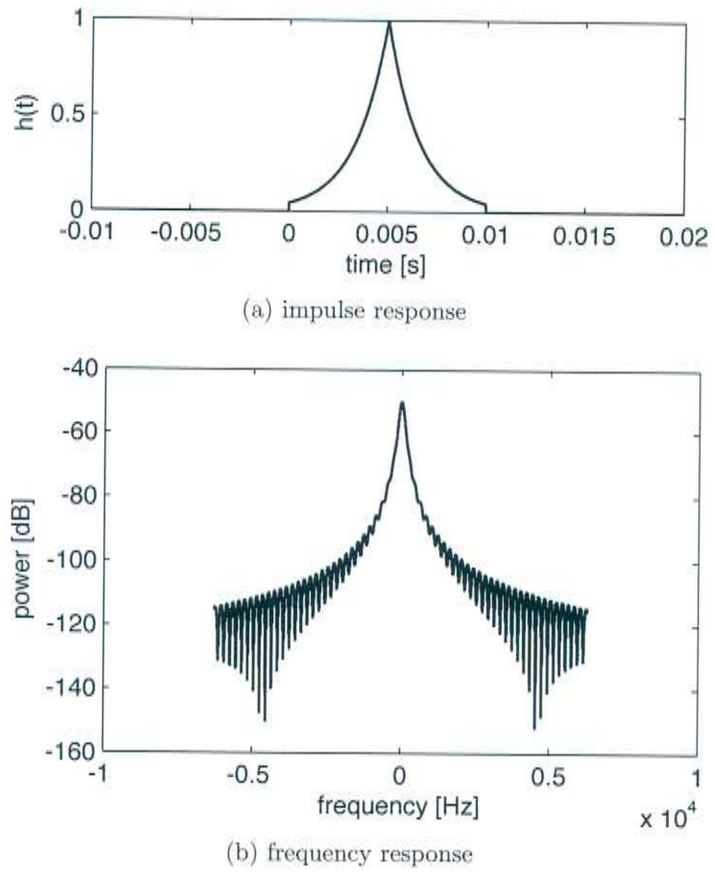
3.2.2 Triplet Filter Calculation

Calculation of the triplet filters starts with a simple filter which has a non-causal impulse response $h(t)$ and a frequency response $H(\omega)$ [Kumaresan12]:

$$h(t) = \exp(-\alpha|t|) \tag{3.33}$$

$$\Rightarrow \quad H(\omega) = \frac{2\alpha}{\omega^2 + \alpha^2} \tag{3.34}$$

For practical use, the non-causal impulse response has to be truncated and shifted in time, such that it begins at $t = 0$. The resulting causal and finite filter impulse response and frequency response is depicted in figure 3.13



(a) impulse response

(b) frequency response

Figure 3.13: Base filter used for triplet filters.

Afterwards, two "intermediate" filters $h_1(t)$ and $h_2(t)$ can be calculated by multiplying $h(t)$ with $\cos(\Delta t)$ and $\sin(\Delta t)$. Δ represents the frequency shift (in radians/second), by which the lower and upper triplet filter are shifted compared to the center filter. Multiplication by a sinusoid in the time-domain leads to a shift of the frequency response in the frequency domain:

$$h_1(t) = h(t) \cdot \cos(\Delta t) \qquad \Rightarrow \qquad H_1(\omega) = \frac{H(\omega - \Delta) + H(\omega + \Delta)}{2} \tag{3.35}$$

$$h_2(t) = h(t) \cdot \sin(\Delta t) \qquad \Rightarrow \qquad H_1(\omega) = -i\frac{H(\omega - \Delta) + H(\omega + \Delta)}{2} \tag{3.36}$$

H_1 and H_2 are then used together with the output of the local oscillator to generate four intermediate signals, named a_1, a_2, b_1 and b_2. There have to be two outputs coming from the local oscillator in order to create these signals, a sine output and a cosine output. The VCO depicted in figure 3.2 (page 14) only generates a sine output. A cosine output can be calculated by feeding the VCO accumulator output into an additional "cos" block. If the current VCO output frequency is ω_c radians/second, the VCO sine output is $\sin(\omega_c t)$ and the cosine output is $\cos(\omega_c t)$. The subscript "c" is used to indicate that this will also be the center frequency of the center triplet filter.

Finally, the intermediate signals a_1 and a_2 can be obtained by multiplying the FLL input u_1 with the VCO cosine output and subsequently filtering the result with H_1 and H_2, respectively. b_1 and b_2 are calculated similarly by using the VCO sine output instead of the cosine output:

$$a_1(t) = (u_1(t) \cdot \cos(\omega_c t)) \otimes h_1(t) \tag{3.37}$$
$$a_2(t) = (u_1(t) \cdot \cos(\omega_c t)) \otimes h_2(t) \tag{3.38}$$
$$b_1(t) = (u_1(t) \cdot \sin(\omega_c t)) \otimes h_1(t) \tag{3.39}$$
$$b_2(t) = (u_1(t) \cdot \sin(\omega_c t)) \otimes h_2(t) \tag{3.40}$$

Here, $\otimes$ denotes convolution. The corresponding frequency domain results are:

$$A_1(\omega) = \frac{U_1(\omega-\omega_c)H_1(\omega) + U_1(\omega+\omega_c)H_1(\omega)}{2} \tag{3.41}$$
$$A_2(\omega) = \frac{U_1(\omega-\omega_c)H_2(\omega) + U_1(\omega+\omega_c)H_2(\omega)}{2} \tag{3.42}$$
$$B_1(\omega) = -i\frac{U_1(\omega-\omega_c)H_1(\omega) - U_1(\omega+\omega_c)H_1(\omega)}{2} \tag{3.43}$$
$$B_2(\omega) = -i\frac{U_1(\omega-\omega_c)H_2(\omega) - U_1(\omega+\omega_c)H_2(\omega)}{2} \tag{3.44}$$

Afterwards, the next step is the production of two more intermediate signals c_1 and c_2, as follows:

$$c_1(t) = a_1(t)\cos(\omega_c t) + b_1(t)\sin(\omega_c t) \tag{3.45}$$
$$c_2(t) = a_2(t)\sin(\omega_c t) - b_2(t)\cos(\omega_c t) \tag{3.46}$$

The corresponding calculations in the frequency domain are a little lengthy, which is why they are not printed here. The results, however, look like this:

$$C_1(\omega) = U_1(\omega) \cdot \underbrace{\frac{H_1(\omega-\omega_c) + H_1(\omega+\omega_c)}{2}}_{=:G_1(\omega)} \tag{3.47}$$
$$C_2(\omega) = U_1(\omega) \cdot \underbrace{-i\frac{H_2(\omega-\omega_c) - H_2(\omega+\omega_c)}{2}}_{=:G_2(\omega)} \tag{3.48}$$

These two equations reveal that the combination of operations, which has been described so far, results in two intermediate signals $c_1(t)$ and $c_2(t)$, which are nothing but the FLL input spectrum U_1 filtered by G_1 and G_2, respectively. The final step which is necessary to obtain the triplet filter output is to add and subtract C_1 and C_2, respectively. Adding C_1 and C_2 produces the lower triplet filter output and subtracting C_2 from C_1 gives the upper triplet filter output:

$$\begin{aligned} UPPER(\omega) &= U_1(\omega) \cdot \left(\frac{H_1(\omega - \omega_c) + H_1(\omega + \omega_c)}{2} + i \frac{H_2(\omega - \omega_c) - H_2(\omega + \omega_c)}{2} \right) \\ &= U_1(\omega) \cdot \underbrace{\frac{H(\omega - \omega_c - \Delta) + H(\omega + \omega_c + \Delta)}{2}}_{=:H_U(\omega)} \end{aligned} \tag{3.49}$$

$$\begin{aligned} LOWER(\omega) &= U_1(\omega) \cdot \left(\frac{H_1(\omega - \omega_c) + H_1(\omega + \omega_c)}{2} - i \frac{H_2(\omega - \omega_c) - H_2(\omega + \omega_c)}{2} \right) \\ &= U_1(\omega) \cdot \underbrace{\frac{H(\omega - \omega_c + \Delta) + H(\omega + \omega_c - \Delta)}{2}}_{=:H_L(\omega)} \end{aligned} \tag{3.50}$$

It can be seen that the frequency response of the upper triplet filter H_U is the frequency response of the prototype filter H, shifted by ω_c and Δ, i.e. shifted to the current VCO frequency and then Δ radians/second further. Correspondingly, H_L is like H, but shifted to the center frequency ω_c and then Δ radians/second back.

All these calculations seem rather complicated at first glance. However, the described procedure has the advantage of using fixed pre-calculated filters H_1 and H_2. Shifting the filter passband to the appropriate VCO frequency at runtime is achieved by a series of time-domain multiplications, which result in the desired frequency shifts in the Fourier domain. Figure 3.14 shows the whole triplet filter circuit

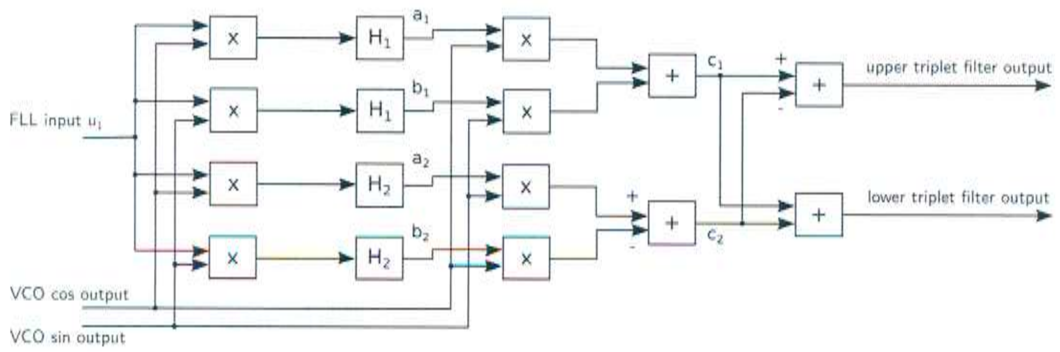


Figure 3.14: Schematic of lower and upper triplet filters.

described above (lower and upper triplet filter outputs). The triplet center filter has been omitted so far. It is not used to steer the VCO frequency and therefore it does not affect the frequency tracking capabilities of the FLL. Instead, the output of the center filter represents the FLL output. The center filter is calculated using a similar procedure like the one used to calculate $c_1(t)$, but the base filter H is used instead of H_1.

By comparing the energy of the lower and upper triplet filter outputs it can be determined if the current VCO frequency is too low or too high. For instance, if the energy of the upper filter output is higher than the energy of the lower filter output, the signal that should be tracked is likely to have a higher frequency than the current VCO frequency. Since the filter outputs cannot be compared directly, it is necessary to calculate and compare their energy, which is achieved by comparing their envelope. Therefore, an envelope detector is necessary as depicted in figure 4 in [Kumaresan12]. For this work, the envelope detector consisted of a Hilbert transform which is used to calculate the complex analytic signal of the upper and lower triplet filter outputs, whose absolute values represent the signal envelopes. According to [Kumaresan12], the logarithm should be taken of each envelope instead of using the envelope directly. It can be seen from the experiments in chapter 4.4 that the base of the logarithm influences the FLL tracking speed and accuracy and higher logarithm bases (50 and higher) worked better than lower bases like 2, 10 or e.

3.2.3 Voltage-Controlled Oscillator and Loop Filter

The VCO for the FLL is essentially the same as the PLL VCO described in section 3.1, except for the additional cosine output, which is needed for the triplet filter. [Kumaresan12] further recommends the use of a standard digital integrator for the loop filter.

3.3 Gammatone Filters

A problem that all of the previously outlined PLL and FLL approaches have is the fact that all of them are only designed to handle a single frequency component in their respective inputs. If a FLL or PLL is used to track one of several frequency components, the PLL or FLL might not be able to track even one of the frequency components, because it constantly switches from one frequency to another.

One way to deal with this problem is to use a large amount of PLLs or FLLs, hoping that at each moment in time, at least one loop will track any of the frequency components. Unfortunately, experiments have shown that the loops tend to switch between components very often. Furthermore, it is difficult to find out when a loop was locked on a frequency and when it was unlocked and moving from one frequency to another. It might also happen that an unlocked loop never becomes locked again because there are too many frequency components at the same time.

In order to track multiple frequencies at once, there is a need for an additional bandpass filter stage prior to the actual PLL/FLL. The filter stage is used to split the whole signal into many frequency bands, which hopefully contain only one frequency component each. [Kumaresan12] suggests the use of a gammatone filterbank for the FLL setup described in the previous section.

Gammatone filters are modeled based on the frequency response of the cells in a human ear and they have originally been described in 1972 by Johannsma. A description of an efficient implementation for gammatone filterbanks can be found in [Holdsworth88]. A single gammatone filter can be described in the time domain by its impulse response:

$$h(t) = at^{n-1} \exp(-2\pi bt) \cos(2\pi f_0 t + \phi) \tag{3.51}$$

In this equation, n is the filter order, b controls the length of the impulse response, i.e. the bandwidth, and f_0 represents the center or peak frequency. ϕ is a phase

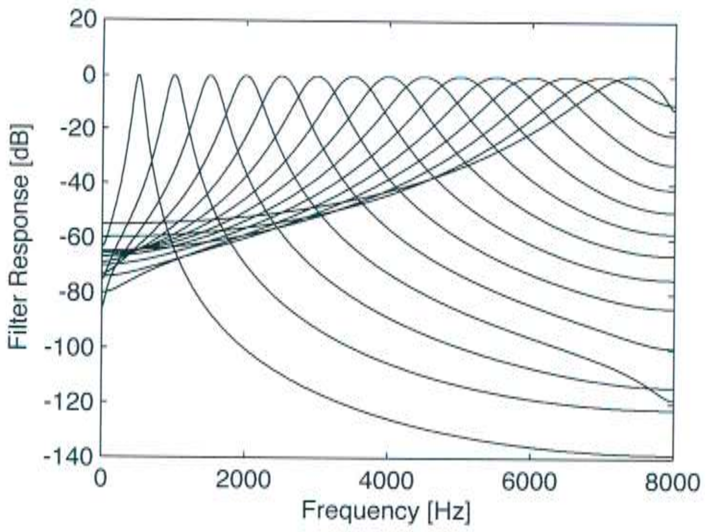


Figure 3.15: Gammatone filter bank with 500 Hz spacing between filters.

term which can normally be set to 0 and a is an additional gain factor. Figure 3.15 depicts an exemplary set of gammatone filters, spaced 500 Hz apart.

It can be seen that the filters are not symmetric and they get broader, the higher their center frequency is. Additionally, the order n determines how steep the slope between pass- and stopband is. [Holdsworth88] shows a special technique, which can be used to apply multiple filters with a low order in order to get the output of a high-order gammatone filter. 4th-order filters have been used for the experiments in this work, made from a cascade of four 1st-order filters.

3.4 Comb Filters

Comb-Filters are a type of filter whose transfer-function has periodic peaks and valleys that resemble the teeth of a comb. A simple computationally inexpensive digital comb filter can be built by adding a scaled and delayed copy of a signal to itself [Smith10]. A schematic of this FIR filter can be found in figure 3.16, where z^{-K} represents a K-sample delay and α is a scaling factor.

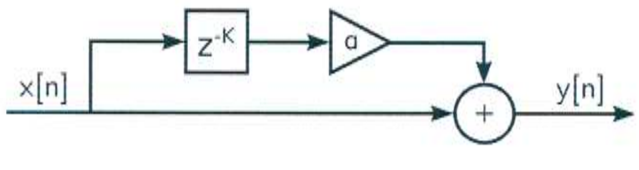


Figure 3.16: Feedforward FIR Comb Filter.

The difference equation for this filter is $y[n] = x[n] + \alpha \cdot x[n-K]$ and a Z-transform of both sides of the equation leads to:

$$Y(z) = X(z) + \alpha z^{-K} \cdot X(z) \tag{3.52}$$
$$= (1 + \alpha z^{-K})X(z) \tag{3.53}$$

Therefore, the Z-domain transfer function is $H(z) := \frac{Y(z)}{X(z)} = 1 + \alpha z^{-K}$. Since the frequency axis can be found on the unit circle of the Z-domain [Smith07], the frequency response can be calculated by substituting $z = \exp(i2\pi f T_s)$, where T_s is the sampling time and f the unnormalized frequency in Hz, which results in a magnitude response of:

$$\begin{aligned}
|H(f)| &= \sqrt{\Re\{1+\alpha\exp(-i2\pi f T_s K)\}^2 + \Im\{1+\alpha\exp(-i2\pi f T_s K)\}^2} && (3.54)\\
&= \sqrt{(1+\alpha\cos(-2\pi f T_s K))^2 + (\alpha\sin(-2\pi f T_s K))^2} && (3.55)\\
&= \sqrt{1+2\alpha\cos(2\pi f T_s K)+\alpha^2} && (3.56)
\end{aligned}$$

Since α is a constant, only the term $2\alpha\cos(2\pi f T_s K)$ is responsible for the peaks and notches in the filter's frequency response. For $\alpha > 0$, peaks can be found at frequencies $f_n, n \in \mathbb{Z}$, where

$$\begin{aligned}
& f_n = \arg\max_f(2\alpha\cos(2\pi f T_s K)) && (3.57)\\
\Leftrightarrow\quad & 2\pi f_n T_s K = 2\pi n && (3.58)\\
\Leftrightarrow\quad & f_n = \frac{n}{T_s K} && (3.59)
\end{aligned}$$

The latter equation states that the peak frequencies are $f = 0, \frac{1}{T_s K}, \frac{2}{T_s K}, \ldots$. This means that there will always be a peak at 0 Hz and further peaks will be located at integer multiples of the fundamental peak frequency $f_1 = \frac{1}{T_s K}$. By solving equation 3.59 for K and setting $n = 1$, the filter can be approximately designed for a certain fundamental peak frequency. For most frequencies, however, it will only be an approximation, since K has to be an integer. It can be seen from equation 3.59 that the approximation will be better if the fundamental peak frequency is low compared to the sampling rate.

The solid line in figure 3.17 is the frequency response of a feedforward comb filter with $\alpha = 1$ and $K = 16$, which means $f_1 = 1000$ Hz at a sampling rate of 16 kHz. The frequency response of this kind of feedforward comb filter has wide peaks and deep notches. For lower (but still positive) α, the notches would be less deep.

The dashed line in figure 3.17 is the frequency response of another type of comb filter, which utilizes feedback instead of a feedforward path and a value of $\alpha = 0.7$. Figure 3.18 is a schematic of this type of feedback IIR comb filter. It can be seen that the feedback comb filter has sharper peaks and wider valleys, which is the opposite of the feedforward comb filter's characteristics.

The difference equation for the feedback comb filter is $y[n] = x[n] + \alpha \cdot y[n-K]$, which leads to the following transfer function:

$$\begin{aligned}
& Y(z) = X(z) + \alpha z^{-k} Y(z) && (3.60)\\
\Leftrightarrow\quad & Y(z)(1-\alpha z^{-k}) = X(z) && (3.61)\\
\Leftrightarrow\quad & H := \frac{Y(z)}{X(z)} = \frac{1}{1-\alpha z^{-K}} && (3.62)
\end{aligned}$$

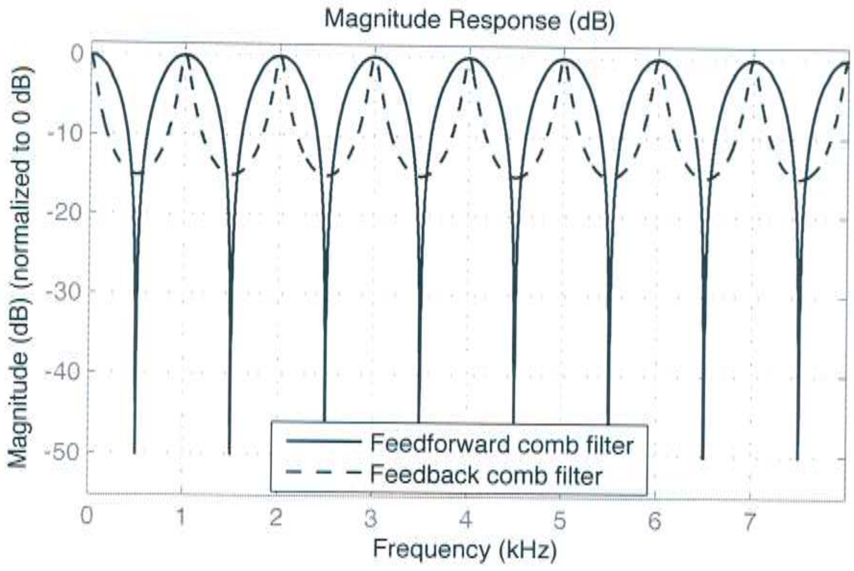


Figure 3.17: Frequency response of comb filters for positive values of α.

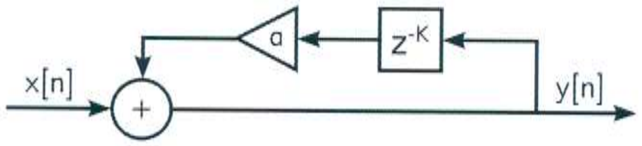


Figure 3.18: Feedback IIR Comb Filter.

For $\alpha > 0$, the location of the peaks of the transfer function can be calculated similar to equation 3.63:

$$|H(f)| = \sqrt{\Re\left\{\frac{1}{1-\alpha\exp(-i2\pi fT_sK)}\right\}^2 + \Im\left\{\frac{1}{1-\alpha\exp(-i2\pi fT_sK)}\right\}^2} \quad (3.63)$$

$$= \sqrt{\frac{(1-\alpha\cos(-2\pi fTK))^2 + (\alpha\sin(-2\pi fTK))^2}{(1-2\alpha\cos(-2\pi fTK)+\alpha^2)^2}} \quad (3.64)$$

$$= \sqrt{\frac{1-2\alpha\cos(-2\pi fTK)+\alpha^2}{(1-2\alpha\cos(-2\pi fTK)+\alpha^2)^2}} \quad (3.65)$$

$$= \frac{1}{\sqrt{1-2\alpha\cos(-2\pi fTK)+\alpha^2}} \quad (3.66)$$

Again, α is a constant and so for $\alpha > 0$, the peaks of this frequency response can be calculated as follows:

$$f_n = \arg\min_f(1-2\alpha\cos(-2\pi fTK)+\alpha^2) \quad (3.67)$$

$$\Leftrightarrow \quad f_n = \arg\max_f(2\alpha\cos(-2\pi fTK)) \quad (3.68)$$

This is the same as equation 3.57, which means that the delay parameter K can be calculated in the same way for the forward and backward comb filter and for a given parameter K, the location of the peaks is the same for both comb filter types. Later in section 4.5 comb filters will be used to enhance the harmonic structure of voiced speech.

3.5 APES Amplitude and Phase Estimator

The abbreviation APES stands for "Amplitude and Phase Estimation of a Sinusoid", which - as the name suggests - is an amplitude estimation algorithm for complex sinusoids. It was first published by Li and Stoica in 1996 [Li96]. Later, it has been found that the APES algorithm can also be interpreted as a matched filter approach, as described in [Stoica97]. [Glentis08] contains a description of an APES implementation that uses certain mathematical properties in order to decrease the computation time.

Furthermore, there is a comparison in [Stoica00], where APES is compared to other spectral estimation methods, such as least-squares estimation or the CAPON estimator [Capon69]. In general, it has been found that APES is a superior means of amplitude estimation compared to these other methods, especially compared to a "standard" windowed Fourier transform. Since voiced speech can be represented by a line spectrum with individual frequency components at f_0 Hz, $2f_0$ Hz, ..., a spectral estimation method used for voiced speech should be good at resolving individual spectral peaks rather than estimating continuous spectra. According to [Li96], FFT-based spectral estimates are better suited in the case of line spectra with many very closely spaced lines, whereas APES seems to be more precise if the spectrum contains fewer lines which are further apart.

Finally, [Li98] contains a comparison of an APES implementation that uses forward filtering with another implementation that uses forward-backward filtering. Since the results of forward-backward APES seem to be better than those of forward APES and the extension of forward APES to forward-backward APES is not very computationally intensive, forward-backward APES has been used for the experiments in this work. The following is a mathematical description of the implemented forward-backward APES.

3.5.1 Derivation of Universal Filter-Based Estimator

Under the assumption that the input signal x contains only a single spectral component with a time-varying instantaneous frequency $\omega(t)$ and a time-varying complex amplitude $\alpha(t) = a(t) \cdot \exp(i \cdot \phi(t))$, x can be modeled in the following way:

$$x(t) = \alpha(t) \cdot \exp\left(i \int_0^t \omega(t)\mathrm{d}t\right) + e(t) \qquad , t \in [0, \infty) \qquad (3.69)$$

$e(t)$ represents the noise term, which is expected to have a mean of zero. Sampling x at a sampling rate of $f_s (= \frac{1}{T_s})$ Hz leads to the following discrete representation:

$$x[n] = \alpha[n] \cdot \exp\left(iT_s \sum_{m=0}^{n-1} \omega[m]\right) + e[n] \qquad , n \in \mathbb{Z} \qquad (3.70)$$

APES is further based on the assumption that the input is stationary and both the amplitude $\alpha[n]$ and the instantaneous frequency $\omega[n]$ are actually constant. To better match these requirements, the input x is split into non-overlapping blocks of N samples each. Within each block $\alpha[n]$ and $\omega[n]$ are approximately constant and they can be represented by α_K and ω_K, respectively, where K is the block index ($k \in \mathbb{Z}$). Therefore, the sum in equation 3.70 can be split into two parts, one for block K, which includes sample n, and another for the blocks prior to K:

$$x[n] \approx \alpha_K \cdot \exp\left(iT_s\left(\sum_{m=0}^{K\cdot N-1}\omega[m] + \sum_{m=K\cdot N}^{n-1}\omega_K\right)\right) + e[n] \tag{3.71}$$

$$= \alpha_K \cdot \exp\left(iT_s\left(\sum_{m=0}^{K\cdot N-1}\omega[m] + (n - K\cdot N)\omega_K\right)\right) + e[n] \tag{3.72}$$

$$= \alpha_K \cdot \exp\left(iT_s\sum_{m=0}^{K\cdot N-1}\omega[m]\right) \cdot \exp\left(iT_s(n - K\cdot N)\omega_K\right) + e[n] \tag{3.73}$$

The first exponential determines the phase offset at the beginning of block K due to the time-varying frequency in the prior blocks. For a continuous analysis of the phase offset over several blocks it is important not to forget that contribution of a time-varying frequency to the phase of the signal. For this work, however, only the real-valued magnitude $|\alpha(t)|$ of the complex amplitude $\alpha(t)$ is of interest. Therefore, the actual phase angle $\angle\alpha(t)$ can be neglected and it is sufficient to examine each block of samples individually by assuming $K \equiv 0$ for all samples.

To avoid confusion with the fully continuous case, a single block of N samples will be denoted by y instead of x from now on. As it has already been stated, the amplitude α and frequency ω are assumed to be constant during a single block, which leads to the following formulation (cf. [Li96], equation 1):

$$y[n] = \alpha \cdot \exp(i\omega T_s n) + e[n] \qquad ,n = [0, 1, \ldots, N-1] \tag{3.74}$$

This will be presented to the APES algorithm in the form of two column vectors (* denotes complex conjugation):

$$\overrightarrow{y} = \begin{bmatrix} y[0] & y[1] & \cdots & y[N-1] \end{bmatrix}^T \tag{3.75}$$

$$\overleftarrow{y} = \begin{bmatrix} y^*[N-1] & y^*[N-2] & \cdots & y^*[1] \end{bmatrix}^T \tag{3.76}$$

$\overrightarrow{y}$ is called the forward vector and $\overleftarrow{y}$ the backward vector, since it contains the samples in reverse order. In the next step, $\overrightarrow{y}$ is split once more into subvectors $\overrightarrow{y_l}$ of length M with an overlap of $M-1$ (e.g. a shift of 1 sample per subvector), which means that there are $L = N - M + 1$ subvectors per input block.

$$\overrightarrow{y_l} = \begin{bmatrix} y[l] & y[l+1] & \cdots & y[l+M-1] \end{bmatrix}^T \qquad ,l = [0, 1, \ldots, L-1] \tag{3.77}$$

$$= \begin{bmatrix} \alpha \cdot \exp(i\omega T_s \cdot l) \\ \alpha \cdot \exp(i\omega T_s \cdot (l+1)) \\ \vdots \\ \alpha \cdot \exp(i\omega T_s \cdot (l+M-1)) \end{bmatrix} + \begin{bmatrix} e[l] \\ e[l+1] \\ \vdots \\ e[l+M-1] \end{bmatrix} \tag{3.78}$$

$$= \alpha \cdot \underbrace{\begin{bmatrix} \exp(i\omega T_s \cdot 0) \\ \exp(i\omega T_s \cdot 1) \\ \vdots \\ \exp(i\omega T_s \cdot (M-1)) \end{bmatrix}}_{=:a(\omega)} \cdot \exp(i\omega T_s l) + \underbrace{\begin{bmatrix} e[l] \\ e[l+1] \\ \vdots \\ e[l+M-1] \end{bmatrix}}_{=:\overrightarrow{e_l}} \tag{3.79}$$

The backward vectors $\overleftarrow{y_l}$ are defined similarly:

$$\begin{aligned}
\overleftarrow{y_l} &= \begin{bmatrix} y^*[N-l-1] & y^*[N-l-2] & \cdots & y^*[N-l-M] \end{bmatrix}^T && (3.80)\\
&= \begin{bmatrix} \alpha^* \cdot \exp(-i\omega T_s \cdot (N-l-1)) \\ \alpha^* \cdot \exp(-i\omega T_s \cdot (N-l-2)) \\ \vdots \\ \alpha^* \cdot \exp(-i\omega T_s \cdot (N-l-M)) \end{bmatrix} + \begin{bmatrix} e^*[N-l-1] \\ e^*[N-l-2] \\ \vdots \\ e^*[N-l-M] \end{bmatrix} && (3.81)\\
&= \alpha^* \cdot a(\omega) \cdot \underbrace{\exp(-i\omega T_s(N-1))}_{=:b(\omega)} \cdot \exp(i\omega T_s l) + \overleftarrow{e_l} && (3.82)
\end{aligned}$$

APES further consists of a finite impulse-response (FIR) filter $h(\omega)$, which has M taps.

$$h(\omega) = \begin{bmatrix} h_0(\omega) & h_1(\omega) & \ldots & h_{M-1}(\omega) \end{bmatrix}^T \tag{3.83}$$

During the runtime of APES, $h(\omega)$ will be calculated in a way such that it passes a certain frequency ω unchanged while it attenuates other frequencies as much as possible, depending on their presence in the input signal. The design of $h(\omega)$ will be presented later in this chapter. For now it is enough to know that there is a filter $h(\omega)$ with the aforementioned properties. Passing $\overrightarrow{y_l}$ and $\overleftarrow{y_l}$ through $h(\omega)$ results in:

$$\begin{aligned}
h^T(\omega)\overrightarrow{y_l} &= \alpha \cdot h^T(\omega) \cdot a(\omega) \cdot \exp(i\omega T_s l) + h^T(\omega)\overrightarrow{e_l} && (3.84)\\
h^T(\omega)\overleftarrow{y_l} &= \alpha^* \cdot h^T(\omega) \cdot a(\omega) \cdot b(\omega) \cdot \exp(i\omega T_s l) + h^T\omega\overleftarrow{e_l} && (3.85)
\end{aligned}$$

These two equations can be simplified further by designing $h(\omega)$ such that $h^T(\omega) \cdot a(\omega) = 1$. Afterwards, all filter outputs can be combined into a single equation.

$$\underbrace{\begin{bmatrix} h^T(\omega)\overrightarrow{y_0} \\ \vdots \\ h^T(\omega)\overrightarrow{y_{L-1}} \\ (h^T(\omega)\overleftarrow{y_0})^* \\ \vdots \\ (h^T(\omega)\overleftarrow{y_{L-1}})^* \end{bmatrix}}_{=:A} = \underbrace{\begin{bmatrix} \exp(i\omega T_s \cdot 0) \\ \vdots \\ \exp(i\omega T_s \cdot (L-1)) \\ b^*(\omega) \cdot \exp(-i\omega T_s \cdot 0) \\ \vdots \\ b^*(\omega) \cdot \exp(-i\omega T_s \cdot (L-1)) \end{bmatrix}}_{=:B} \cdot \alpha + \underbrace{\begin{bmatrix} h^T(\omega)\overrightarrow{e_0} \\ \vdots \\ h^T(\omega)\overrightarrow{e_{L-1}} \\ (h^T(\omega)\overleftarrow{e_0})^* \\ \vdots \\ (h^T(\omega)\overleftarrow{e_{L-1}})^* \end{bmatrix}}_{=:C} \tag{3.86}$$

Using a pseudoinverse of B, α can be isolated ($(\cdot)^H$ denoting the conjugate transpose):

$$(B^H B)^{-1} B^H A = \alpha + (B^H B)^{-1} B^H C \tag{3.87}$$

This can be further simplified using the following transformations.

$$B^H B = \sum_{l=0}^{L-1} \exp(-i\omega T_s l) \cdot \exp(i\omega T_s l) \tag{3.88}$$

$$+ \sum_{l=0}^{L-1} b(\omega) \cdot \exp(i\omega T_s l) \cdot b^*(\omega) \cdot \exp(-i\omega T_s l)$$

$$= L + b(\omega)b^*(\omega) \cdot L \tag{3.89}$$

$$= L + \exp(-i\omega T_s(N-1)) \cdot \exp(i\omega T_s(N-1)) \cdot L \tag{3.90}$$

$$= 2L \tag{3.91}$$

$$B^H A = \sum_{l=0}^{L-1} \exp(-i\omega T_s l) h^T(\omega) \overrightarrow{y_l} + \sum_{l=0}^{L-1} b(\omega) \cdot \exp(i\omega T_s l)(h^T(\omega)\overleftarrow{y_l})^* \tag{3.92}$$

$$= h^T(\omega) \left(\sum_{l=0}^{L-1} \exp(-i\omega T_s l) \overrightarrow{y_l} \right) + h^H(\omega) b(\omega) \left(\sum_{l=0}^{L-1} \exp(-i\omega T_s l) \overleftarrow{y_l} \right)^* \tag{3.93}$$

$B^H C$ can be calculated similarly to $B^H A$. However, after dividing it by $B^H B = 2L$ it will result in the mean of $e[n]$, which is 0 based on the basic assumptions for the APES algorithm. Finally, the simplified version of equation 3.87 is:

$$\alpha = \frac{1}{2L} \left(h^T(\omega) \left(\sum_{l=0}^{L-1} \exp(-i\omega T_s l) \overrightarrow{y_l} \right) + h^H(\omega) b(\omega) \left(\sum_{l=0}^{L-1} \exp(-i\omega T_s l) \overleftarrow{y_l} \right)^* \right) \tag{3.94}$$

It has been found in [Li96] that the first summand in the outer parenthesis is the same as the second summand if $h(\omega)$ is calculated in a certain way. For the derivation of $h(\omega)$ as it will be presented in the following section, the equation eventually reduces to its final form:

$$\alpha = \frac{h^T(\omega)}{L} \cdot \sum_{l=0}^{L-1} \exp(-i\omega T_s l) \overrightarrow{y_l} \tag{3.95}$$

3.5.2 Filter-Design for APES

The method mentioned in the previous section is applicable for various filters $h(\omega)$ and indeed, there are several filter-based amplitude estimation approaches. It is interesting to note that for $h(\omega) \equiv 1$ (no filter), equation 3.95 reduces to the discrete normalized Fourier transform. CAPON [Capon69] is another filter-based approach and APES is an extension of CAPON with less statistical bias [Stoica97]. CAPON and APES are both matched-filter approaches because the filter $h(\omega)$ is calculated to fit the input data optimally such that the signal-to-noise ratio of the filter output is maximized. This means that the design frequency component ω_d is kept constant ($h^T(\omega_d) = 1$), while other frequencies are attenuated. Overall, $h(\omega)$ can be expressed by the following equation [Li98]:

$$h(\omega) = \arg\max_{h(\omega)} \frac{|h^H(\omega)a(\omega)|^2}{h^H(\omega)Qh(\omega)} \tag{3.96}$$

$$= \frac{Q^{-1}a(\omega)}{a^H(\omega)Q^{-1}a(\omega)} \tag{3.97}$$

Q represents the noise covariance matrix, which is defined by $Q := \mathcal{E}\{\overrightarrow{e_l}\overrightarrow{e_l}^H\} = \mathcal{E}\{\overleftarrow{e_l}\overleftarrow{e_l}^H\}$, where $\mathcal{E}$ denotes the expected value. The transition from 3.96 to 3.97 is based on the Cauchy-Schwartz inequality [Li98]. Q can be obtained in the following way starting with the sample covariance matrix R:

$$\begin{aligned}
R &:= \mathcal{E}\{\overrightarrow{y_l}\overrightarrow{y_l}^H\} = \mathcal{E}\{\overleftarrow{y_l}\overleftarrow{y_l}^H\} && (3.98)\\
&= \mathcal{E}\{(\alpha \cdot a(\omega) \cdot \exp(i\omega T_s l) + \overrightarrow{e_l})(\alpha \cdot a(\omega) \cdot \exp(i\omega T_s l) + \overrightarrow{e_l})^H\} && (3.99)\\
&= \mathcal{E}\{(\alpha \cdot a(\omega) \cdot \exp(i\omega T_s l))(\alpha \cdot a(\omega) \cdot \exp(i\omega T_s l))^H\} + \mathcal{E}\{\overrightarrow{e_l}\overrightarrow{e_l}^H\} && (3.100)\\
&\quad + \underbrace{\mathcal{E}\{(\alpha \cdot a(\omega) \cdot \exp(i\omega T_s l)\overrightarrow{e_l}^H)\}}_{=0} + \underbrace{\mathcal{E}\{\overrightarrow{e_l}(\alpha \cdot a(\omega) \cdot \exp(i\omega T_s l)^H)\}}_{=0} &&\\
&= \mathcal{E}\{\alpha \cdot a(\omega) \cdot \alpha^H \cdot a(\omega)^H\} + \mathcal{E}\{\overrightarrow{e_l}\overrightarrow{e_l}^H\} && (3.101)\\
&= |\alpha|^2 \cdot a(\omega)a(\omega)^H + Q && (3.102)
\end{aligned}$$

The third and fourth summand in equation 3.100 are zero because $\overrightarrow{e_l}$ represents additive white Gaussian noise, which has an expected value of zero by definition and which is also statistically independent of $\alpha \cdot a(\omega) \cdot \exp(i\omega T_s l)$. The forward-backward sample covariance matrix R can be calculated as follows from the forward and backward samples:

$$R = \frac{1}{2}\Bigg(\underbrace{\frac{1}{R}\sum_{l=0}^{L-1}\overrightarrow{y_l}\overrightarrow{y_l}^H}_{\overrightarrow{R}} + \underbrace{\frac{1}{R}\sum_{l=0}^{L-1}\overleftarrow{y_l}\overleftarrow{y_l}^H}_{\overleftarrow{R}}\Bigg) \tag{3.103}$$

Therefore, using equation 3.103 with equation 3.102 leads to:

$$\begin{aligned}
Q &= \frac{1}{2}\left(\frac{1}{R}\sum_{l=0}^{L-1}\overrightarrow{y_l}\overrightarrow{y_l}^H + \frac{1}{R}\sum_{l=0}^{L-1}\overleftarrow{y_l}\overleftarrow{y_l}^H\right) - |\alpha|^2 \cdot a(\omega)a(\omega)^H && (3.104)\\
&= R - (\alpha a(\omega))(\alpha a(\omega))^H && (3.105)
\end{aligned}$$

For each vector of N input samples to APES, there are L subvectors of length M that can be used for a least-squares estimation of $\alpha a(\omega)$. Based on equations 3.79 and 3.82 the estimates (denoted by $\widehat{\ }$) obtained using the forward input vectors and the backward input vectors, respectively, are:

$$\widehat{\overrightarrow{\alpha a(\omega)}} = \frac{1}{L}\sum_{l=0}^{L-1}\overrightarrow{y_l} \cdot \exp(-i\omega T_s l) \tag{3.106}$$

$$\widehat{\overleftarrow{\alpha a(\omega)}} = \frac{1}{L}\sum_{l=0}^{L-1}\overleftarrow{y_l} \cdot \exp(-i\omega T_s l) \tag{3.107}$$

Since R in equation 3.105 was calculated using both the forward and backward sample covariance matrix, the second part of that equation also has to be calculated using both $\widehat{\overrightarrow{\alpha a(\omega)}}$ and $\widehat{\overleftarrow{\alpha a(\omega)}}$, which results in the following final equations for the noise covariance matrix Q, which is used in combination with equation 3.97 to form the APES estimator.

$$Q = R - \frac{1}{2}\left(\widehat{\overrightarrow{\alpha a(\omega)}}\,\widehat{\overrightarrow{\alpha a(\omega)}}^H + \widehat{\overleftarrow{\alpha a(\omega)}}\,\widehat{\overleftarrow{\alpha a(\omega)}}^H\right) \tag{3.108}$$

The aforementioned CAPON estimator can be obtained using equation 3.97 as well, but choosing $Q = R$. Analysis of both estimators has shown that CAPON is a biased estimator whereas APES is unbiased within up to a second-order approximation [Stoica97]. It has also been found that (unsurprisingly) a combined forward-backward filtering approach such as the one presented in this section yields better results than a forward- or backward-only approach [Li98]. The same paper also contains further mathematical simplifications, which allow to implement the APES algorithm without actually calculating all the necessary matrices R, Q, etc. Due to the inherent relation between the forward and backward input vectors it is often sufficient to calculate a forward vector and multiply it with a constant to obtain the corresponding backward vector.

For practical considerations it is further important to choose an appropriate input vector lengh N and subvector length M. [Li96] suggests a choice of $M = \frac{N}{2}$ for optimal amplitude estimates. N, on the other hand, depends on the characteristics of the input signal. For signals with slowly changing parameters (frequency, amplitude), a higher N can be chosen for better statistical performance. A lower N should be chosen instead for highly dynamic signals with rapidly changing parameters.

To demonstrate the estimation performance of the APES estimator, a test signal with 6 frequency components has been created. The topmost figure depicts the time-varying instantaneous frequency of each component. The center figure shows the actual time-varying amplitude of these frequency components and the bottom figure depicts the APES output for the given frequencies.

3.6 Impulse Response Late Reflections Estimation

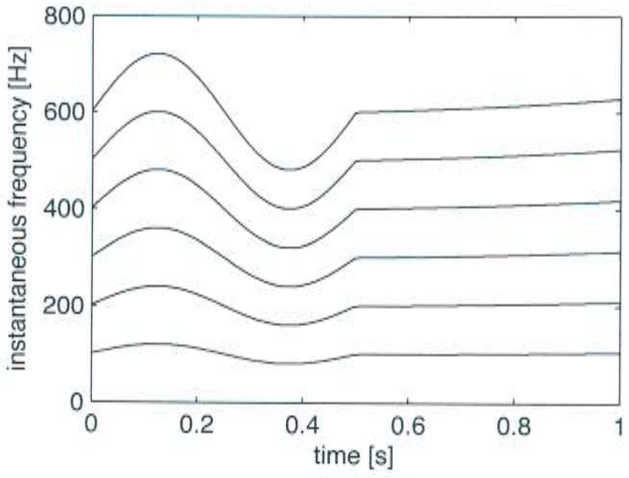
It can be observed from the results in section 4.2 that APES is capable of estimating the first few impulses of a room impulse response. This is done by calculating the amplitude and phase of a sinusoid with constant frequency played back in a room. Rapid changes in phase or amplitude are a sign for a wavefront which arrives at the microphone. The magnitude of a single impulse of the impulse response can then be calculated by comparing the amplitude and phase before the change to the values after the change.

However, this method is only accurate for the first couple of impulses (early reflections), whereas the late reflections cannot be estimated properly. It is therefore necessary to find a way to estimate the late reflection-part of the impulse response. This section describes a way to do just that given the first few impulses and an estimate of the reverberation time T_{60}. The desired impulse response will be denoted with $h(t) := \mathbb{R} \to \mathbb{R}$.

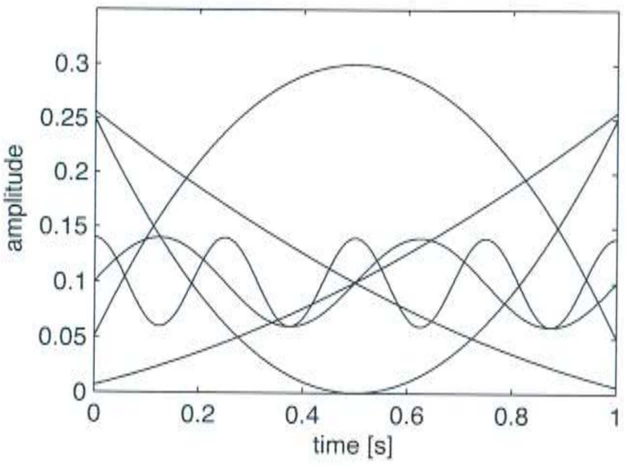
According to Schroeder, the energy $E(t)$ that is left in an impulse response after a certain time t is the integral over the square of the impulse response from t to infinity [Schroeder65]. This is the so-called Schroeder-integral:

$$E(t) = \int_t^\infty h^2(x)\mathrm{d}x \tag{3.109}$$

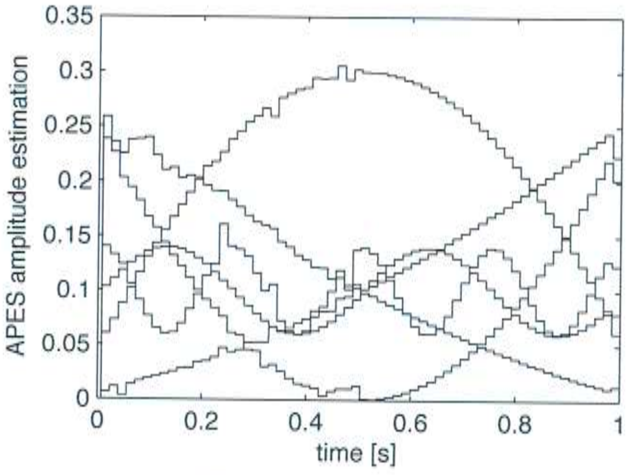
The energy of the reverberations in a room decays approximately exponentially after the direct sound has disappeared. An exponential decay of the energy results in a



(a) 6 frequency components



(b) actual amplitude



(c) APES amplitude estimation

Figure 3.19: Amplitude estimation using APES with $N = 256$ and $M = 128$.

linear decline in the decibel domain. Thus, it is assumed that $E(t)$ decreases by 60 dB every T_{60} s, i. e. the slope of the decay curve in the decibel domain is $-\frac{60}{T_{60}}$:

$$-\frac{60}{T_{60}} \cdot t + 10 \cdot \log(E(0)) = 10 \cdot \log(E(t)) \tag{3.110}$$

$$10^{-\frac{6}{T_{60}} \cdot t} \cdot E(0) = E(t) \tag{3.111}$$

$$10^{-\frac{6}{T_{60}} \cdot t} \cdot E(0) = E(0) - (E(0) - E(t)) \tag{3.112}$$

$$10^{-\frac{6}{T_{60}} \cdot t} = 1 - \frac{E(0) - E(t)}{E(0)} \tag{3.113}$$

$$E(0) = \frac{E(0) - E(t)}{1 - 10^{-\frac{6}{T_{60}} \cdot t}} \tag{3.114}$$

$$E(0) = \frac{\int_0^\infty h^2(x)\mathrm{d}x - \int_t^\infty h^2(x)\mathrm{d}x}{1 - 10^{-\frac{6}{T_{60}} \cdot t}} \tag{3.115}$$

$$E(0) = \frac{\int_0^t h^2(x)\mathrm{d}x}{1 - 10^{-\frac{6}{T_{60}} \cdot t}} \tag{3.116}$$

Assuming that an estimate $\widehat{h}(t)$ for the impulse response is known for $t \in [0; t_k]$, this can be used to estimate $E(0)$—in conjunction with an estimate $\widehat{T_{60}}$ for the reverberation time:

$$\widehat{E(0)} = \frac{\int_0^{t_k} \widehat{h}^2(x)\mathrm{d}x}{1 - 10^{-\frac{6}{\widehat{T_{60}}} \cdot t_k}} \tag{3.117}$$

Equation 3.116 can also be rewritten like this:

$$E(0) - E(0) \cdot 10^{-\frac{6}{T_{60}} \cdot t} = \underbrace{\int_0^t h^2(x)\mathrm{d}x}_{=g(t)-g(0)} \tag{3.118}$$

Here, $g(t)$ denotes an antiderivative of $h^2(t)$. It is now possible to differentiate both sides of the equation with respect to t. The derivative of $g(t)$ with respect to t is $h^2(t)$ by definition and furthermore, because $g(0)$ does not depend on t, it disappears from the equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(E(0) - E(0) \cdot 10^{-\frac{6}{T_{60}} \cdot t}\right) = \frac{\mathrm{d}}{\mathrm{d}t}(g(t) - g(0)) \tag{3.119}$$

$$E(0) \cdot 10^{-\frac{6}{T_{60}} \cdot t} \cdot \ln(10) \cdot \left(-\frac{6}{T_{60}}\right) = h^2(t) \tag{3.120}$$

This equation can then be solved for $h(t)$ by using $\widehat{E(0)}$ from equation 3.117 and $\widehat{T_{60}}$:

$$h(t) = \sqrt{\widehat{E(0)} \cdot 10^{-\frac{6}{\widehat{T_{60}}} \cdot t} \cdot \ln(10) \cdot \left(-\frac{6}{\widehat{T_{60}}}\right)} \tag{3.121}$$

4. Experiments

4.1 Description of Evaluation Data

4.1.1 Database of Recordings

The "CSTR US KED Timit" database (http://festvox.org/dbs/dbs_kdt.html) from the Festvox project of the Language Technology Institute of Carnegie Mellon University has been chosen for evaluation purposes. The database contains the same 452 phonetically balanced utterances as the TIMIT database [Garofolo93], but the utterances have been spoken by a different male speaker at the Centre for Speech Technology Research at the University of Edinburgh. The database was originally chosen because in addition to the usual data it also contains electroglottograph (EGG) data that reveals the movement of the vocal cords. EGG data can be used to calculate the speaker's pitch by a method described in [Bagshaw93], which has already been used for the evaluation of pitch-tracking algorithms, for example in [Seltzer00]. All recordings are sampled at a sampling rate of 16 kHz and therefore, all subsequent processing was also carried out at that sample rate.

4.1.2 Impulse Responses

Artificial reverb was generated for all .wav-files by convolving them with six different impulse responses. Three of these impulse responses were taken from the Aachen Impulse Response Database (AIR) [Jeub09] and the other three were artificial impulse responses created by the Roomsim v3.4 impulse response generator software.

The AIR database of impulse responses contains binaural impulse responses which have been measured in various rooms and with various distances from speaker to microphone. It also contains impulse response measurements of microphones located in dummy heads like the KEMAR. The following code was used to load the impulse responses. load_air is supplied by the AIR software package and its output h contains the impulse response specified by the supplied parameters.

```
airpar.fs       = 16000;      % sampling frequency (Hz)
airpar.rir_type = 1;      % 1 = binaural, 2 = dual-channel mockup phone
airpar.room      = 2;     % 2 = office room
airpar.head      = 0;     % 0 = no dummy head, 1 = with dummy head
airpar.rir_no    = 1;     % 1 = 100 cm, 2 = 200 cm, 3 = 300 cm
airpar.channel   = 0;     % 0 = right channel, 1 = left channel

[h, info] = load_air(airpar);
```

The chosen impulse responses were recorded in an office room and the distance from the speaker to the microphone was 1 m, 2 m and 3 m, respectively.

Roomsim is a software which calculates the impulse response of a room/speaker/microphone setup using the so-called image method, which is described in [Allen79]. It works by repeatedly "mirroring" a shoebox room and especially the position of the sound source in the room along its outer walls. After a certain selectable amount of mirroring operations Roomsim can calculate the "direct" path from each image of the sound source to the microphone. Along the way, it consideres the absoption coefficients of the walls that the corresponding sound wave would hit during its way from the real speaker to the microphone.

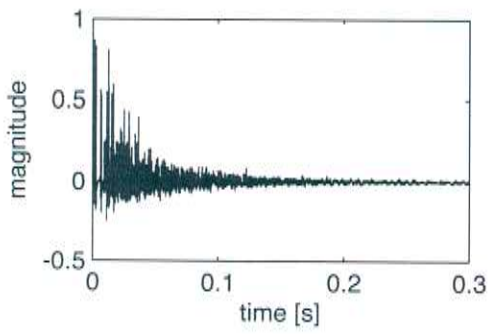
All roomsim impulse responses were generated with a sampling rate of 16 kHz, which is a standard for speech recognition systems. The simulated environment was a cuboid-shaped "shoebox" room with dimensions of 4.2 x 1.9 x 2.2 m (x, y and z axis). The virtual microphone was placed at the coordinate (1.5, 1.55, 1.1), i.e. at half the room height. The artificial speaker was placed at coordinates (2.1, 1.55, 1.1), (2.7, 1.55, 1.1) and (3.3, 1.55, 1.1) for the 60 cm, 120 cm and 180 cm distance recordings, respectively. Roomsim comes with a selection of wall materials, which are used to determine the reverberation coefficients. The following materials have been used:

X-axis wall 1: Glazed wall
X-axis wall 2: Draperies, medium velours
Y-axis wall 1: Ordinary window glass
Y-axis wall 2: Glazed wall
Z-axis wall 1 (floor): Heavy carpet on concrete
Z-axis wall 2 (ceiling): Glazed wall

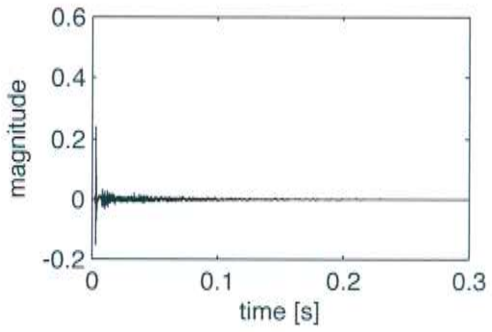
Figure 4.1 depicts all the impulse responses that were used in this work. In the figure, they are truncated to show the most meaningful time span up to 0.3 s. Figure 4.2 shows the relative energy decay of each of these impulse responses as determined by the Schroeder backwards integration method [Schroeder65]. At first, it seems remarkable that the artificially generated impulse responses have a much slower energy decay compared to the AIR impulse responses. The reason for this observation is that Roomsim does not simulate any content in the room such as furniture, wall decoration, people, etc. All these things that are present in a real environment would speed up the energy decay.

4.1.3 Pitch

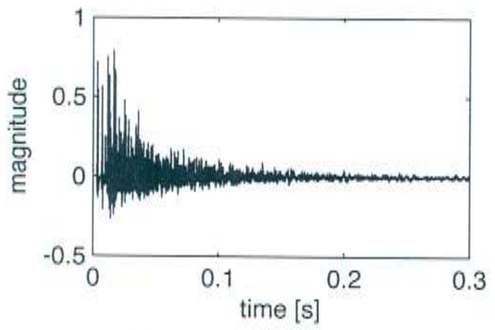
In order to run APES on the artificially reverberated .wav-files it was necessary to first obtain a pitch track for each file. Although the "CSTR US KED Timit" database already contains EGG data that can be used to generate a pitch track, the method was found to be too error-prone. The EGG records the movement of the



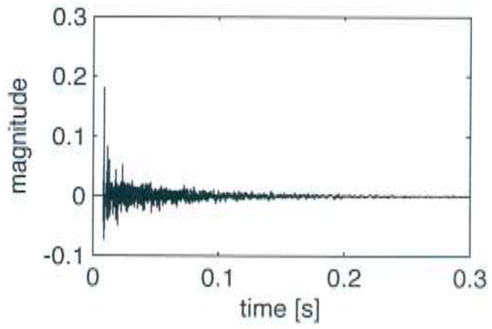
(a) Roomsim 60 cm



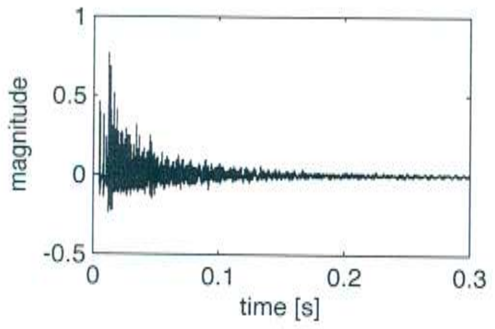
(b) Air 100 cm



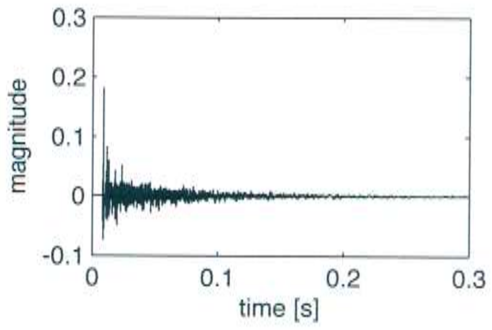
(c) Roomsim 120 cm



(d) Air 200 cm

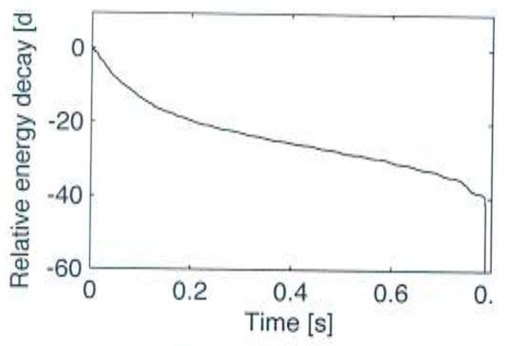


(e) Roomsim 180 cm

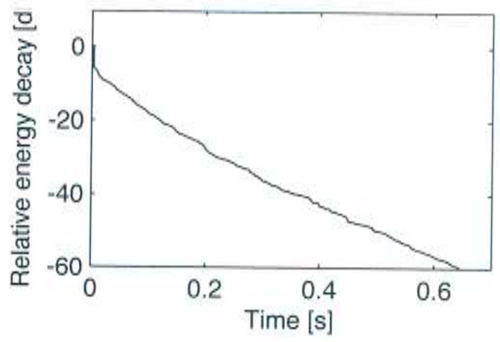


(f) Air 300 cm

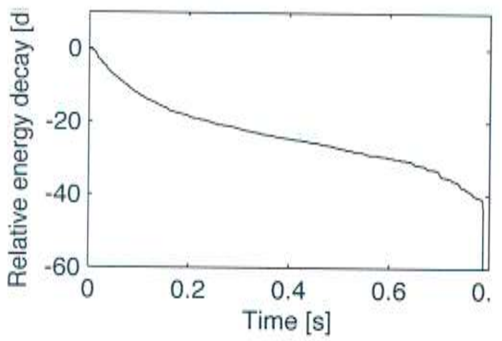
Figure 4.1: Impulse respones used for experiments.



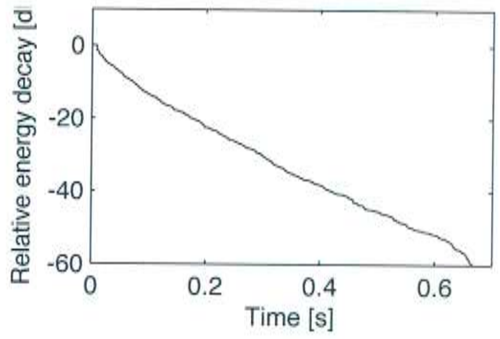
(a) Roomsim 60 cm



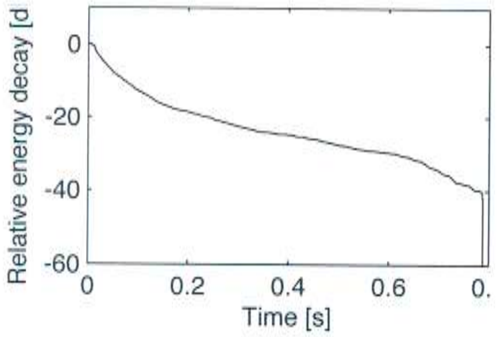
(b) Air 100 cm



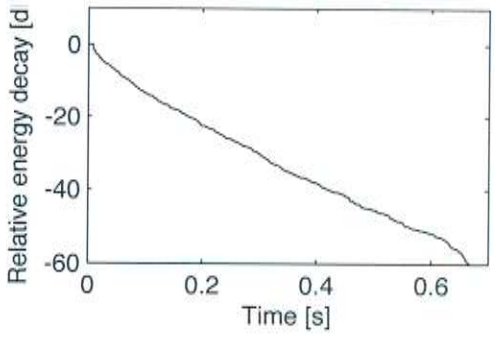
(c) Roomsim 120 cm



(d) Air 200 cm



(e) Roomsim 180 cm



(f) Air 300 cm

Figure 4.2: Relative energy decay of all impulse respones used for experiments.

vocal cords and the frequency of peaks and valleys in the EGG recording reflect the pitch. After inspecting some of the EGG recordings it appears as if the vocal cords always move a little bit, which means the method is not very robust without further post-processing. Additionally, the EGG recording does not indicate which part of the speech is voiced and which part is unvoiced.

Two different methods were used instead of an EGG-derived pitch contour. The first was a baseline "oracle" pitch contour obtained by Mobile Technologies' "Giga" speech synthesis engine in combination with the "Matthew" voice. Its fully parametrized HMM-based speech synthesis produces a reliable and robust pitch contour along with the synthesized speech output. The second method was a more realistic real-world approach using the Histopitch pitch tracker [Seltzer00].

Since the speech synthesis engine produces only close-talk speech and the Histopitch algorithm is not reliable enough in reverberated environments, both methods were not suitable to get pitch tracks for the reverberated data. Instead, the close-talk pitch track was used and shifted in time to account for the time it takes for the sound to travel from the speaker to the microphone.

4.1.4 Speech Recognition System

A speech recognition system was needed for the evaluation of audio data and the Janus Speech Recognition toolkit [Soltau01] (http://isl.ira.uka.de/english/1406.php) was used for this work. The ASR system utilized a vocabulary of 141000 words (including variants) and a language model containing 34 million 4-grams, which were obtained from general-purpose data and web-data. Acoustic modeling was done using 4000 codebooks with 16000 distributions. The training data for the speech recognition system was not processed with any of the methods described in chapter 3, i.e. training and evaluation conditions were unmatched. This was done in order to evaluate if the described methods offer better performance on their own, so that they could be used as a kind of plug-in signal processing in the front-end of a speech recognition system.

4.2 Comparison of Amplitude and Phase Estimators

The predecessor to this work, [Huber11] contains an experiment where phase-locked loops had been used to track the amplitude and phase of sinusoids, which have been played back in a room. The recordings consisted of only one constant frequency component, which meant that the reverb could be modeled as a superposition of sinusoids.

The sum of two sine waves at the same frequency, but with different amplitudes and phase offsets, results in another sine wave with that frequency, but a different amplitude and phase offset. This means that the amplitude and phase of the recorded superposition of sine waves changes whenever a new reflected wavefront arrives at the microphone, as depicted in figure 4.3.

A two-pass PLL system was used to estimate the amplitude and the phase of the recorded sine wave. In the first pass, the frequency and amplitude of the recorded

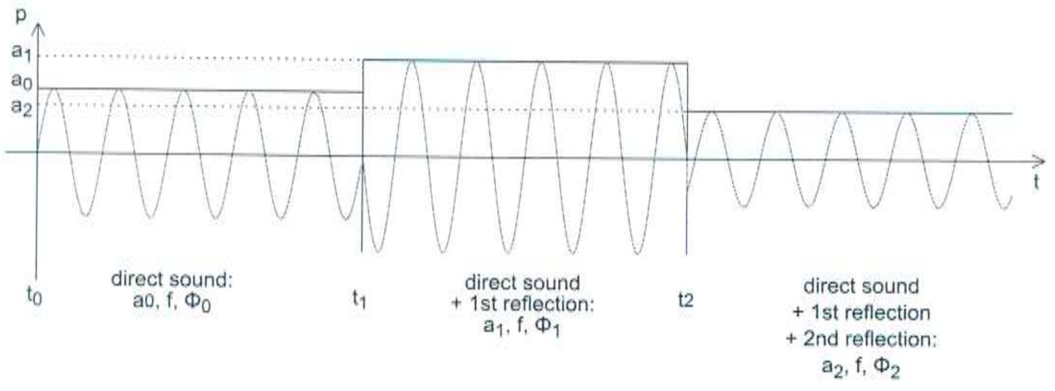


Figure 4.3: Superposition of sine waves

sine wave was estimated using a phase-locked loop. The frequency estimate was then used in the second pass to compare the phase of the recording to a reference sine wave with the same frequency, but constant phase.

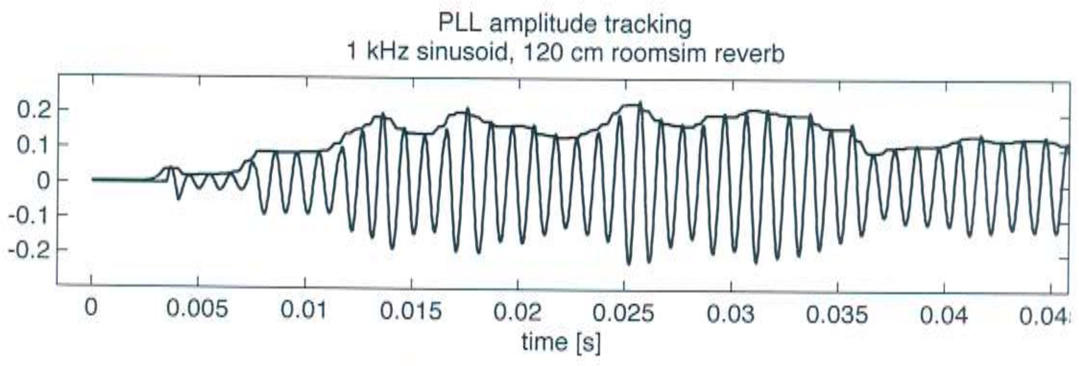
After obtaining the amplitude and phase estimates, their derivative could be calculated to find out when the amplitude or phase-signal exhibits rapid changes. These are the moments when a new reflection arrives at the microphone. Since a recording contains only the combined amplitude and phase that results from the superposition of the direct wavefront and many reflected wavefronts, the amplitude and phase offset of individual components (sine waves) can not be seen directly.

Instead, the difference in phase and amplitude between two subsequent arrivals of a reflected wavefront must be evaluated. This information can then be used to generate the reflected wavefront artificially and to subtract it from the recording, thus removing one particular reflected wavefront from the recording.

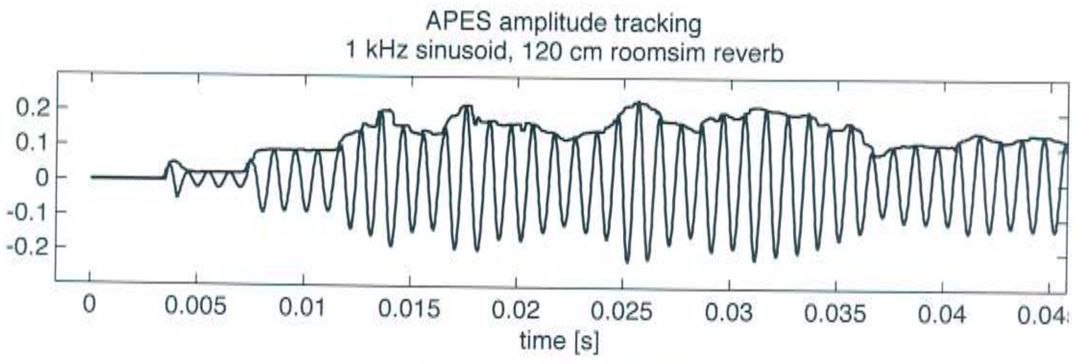
This method needs very precise estimates of the amplitude and phase offset in order to work properly. Since APES is a very precise means of amplitude and phase estimation, a comparison was conducted between the former PLL-based estimates and a set of new APES estimates. Figure 4.4 depicts the PLL-based amplitude estimate along with an estimate obtained from APES for an 1 kHz sine wave that had been convolved with the 120 cm roomsim impulse response.

At first glance, it seems that the APES-based estimate is more precise and it is true that the dereverberation performance (measured by the improvement of the direct-to-reverberant ratio) was better using the APES estimates. The direct-to-reverberant ratio (DRR) is the quotient of the power of the direct sound divided by the reverberation power. The DRR could be improved by up to 12 dB by subtracting the estimated reflected wavefronts from the recording.

However, it was found that the performance improvement observed through the usage of APES-based estimates had more to do with the fact that APES was able to better reveal the exact time for the amplitude and phase changes that occur due to the arrival of a new wavefront. The estimates in the steady-state between two subsequent arriving wavefronts did not differ very much and therefore, the DRR improvement observed with the APES estimates was not much better than PLL-based results.



(a) PLL-based amplitude tracking



(b) APES amplitude tracking

Figure 4.4: Comparison of EPLL-based amplitude estimation and APES

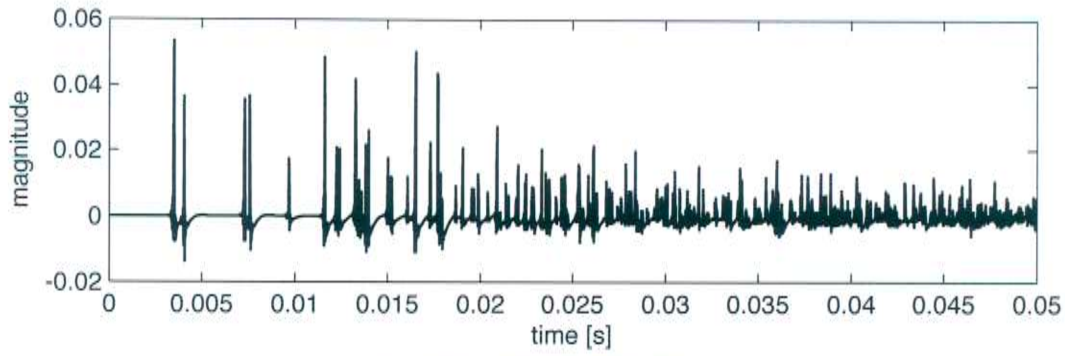
4.3 Impulse Response Estimation Experiments

It has been stated in the previous section that APES did quite well when it came to estimating the moment in time when a particular wavefront arrived at the microphone. It was therefore self-evident to carry out an experiment that would reveal if the estimated wavefront arrival times had anything to do with the actual times.

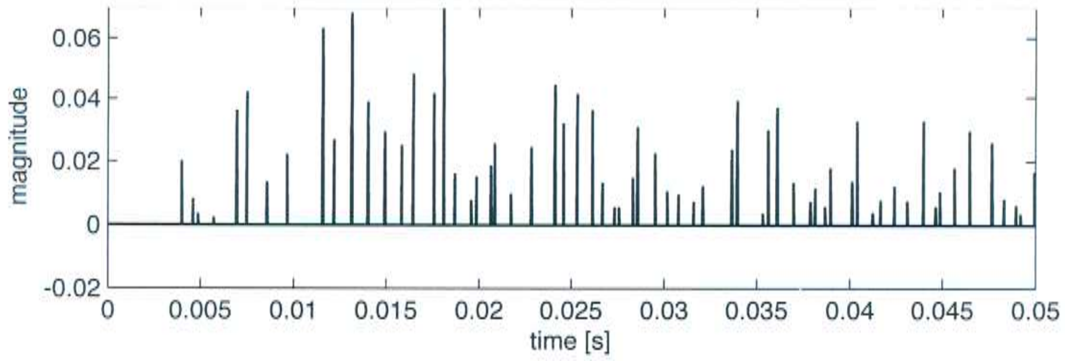
Roomsim was used to generate impulse responses (60 cm, 120 cm and 180 cm) for the experiment, which should then be estimated by the PLL-based method and APES alike. Indeed, the first results were promising and it appeared that APES was superior to the PLL-based approach. Figure 4.5 depicts the first 0.05 seconds of an actual Roomsim-generated impulse response along with the PLL- and APES-based estimates.

While neither of the methods works particularly well for later reflections, APES is better at tracking the first couple of impulses. The PLL method also seems to detect impulses at times where there are none, such as between the fourth and fifth actual impulse. The poor performance of both methods for later impulses can be explained by the fact that only the first couple of impulses are individual, distinctive impulses. After a short amount of time there are so many reflected wavefronts arriving at the microphone roughly at the same time that it is no longer possible to distinguish individual impulses.

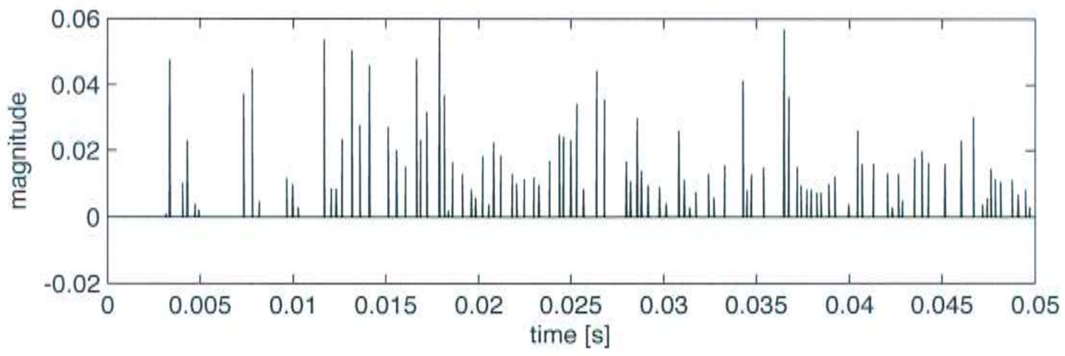
To try and find out if the so-obtained impulse responses could be used to deconvolve actual speech signals, the individual late reflections of the depicted estimates have



(a) Actual impulse response



(b) PLL-based impulse response estimate



(c) APES-based impulse response estimate

Figure 4.5: Example of impulse-response estimation based on the amplitude estimation of an 1 kHz sinusoid (c.f. figure 4.4).

been replaced by an exponentially decaying curve, as explained in section 3.6. Afterwards, the resulting impulse response estimate was used with the standard Wiener deconvolution technique which minimizes the effect of the inverse filter while the signal has a low signal-to-noise ratio. However, the results of these deconvolution experiments did not even seem to be suitable for human hearing experiments, yet alone for automatic speech recognition, which is why the approach was not examined further.

4.4 Pitch Estimation using PLLs and FLLs

A more promising approach seemed to be the tracking of harmonic frequency components during voiced speech using PLLs or FLLs. However, it became obvious very quickly that FLL frequency tracking works better than PLL frequency tracking, due to the fact that PLLs actually try to acquire phase lock, whereas FLLs don't. The phase of a frequency component is constantly changing due to all the reflected wavefronts which arrive at the microphone. This makes it very hard for a PLL to stay in lock. FLLs, on the other hand, have less problems. Figure 4.6 is a so-called capture-

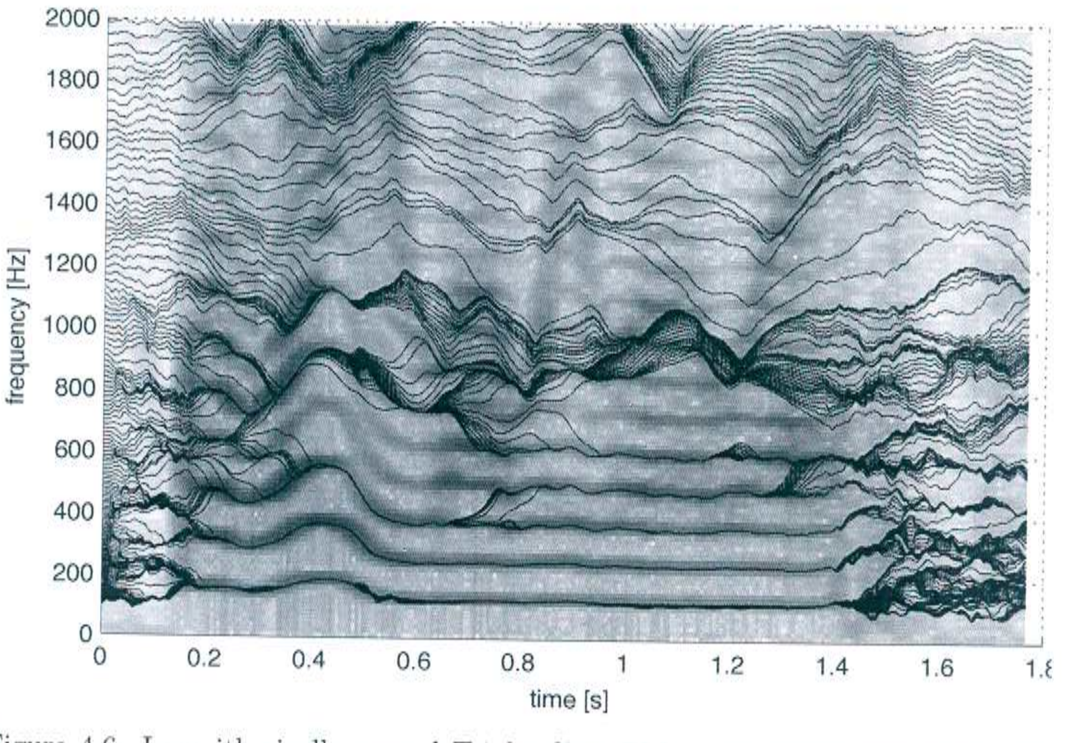


Figure 4.6: Logarithmically spaced Triplet-filter FLLs with wide gammatone prefilter bandwith, raw output

gram. It consists of a spectrogram (TIMIT utterance SX9) in the background and the FLL frequency output in the foreground (black lines). Each black line represents a single FLL and it is obvious that in this case, the frequency locking works only for frequencies below 1 kHz. Figure 4.6 was created using logarithmically spaced FLLs, i.e. the gammatone prefilters had a narrower bandwidth the lower the bandpass center frequency was. The gammatone bandwidth was generally rather high, though. Additionally, more FLLs have been placed at lower frequencies than at higher frequencies. It can be seen that most of the FLLs operating below 600 Hz chose one harmonic frequency. Some harmonics, for example the 4th, is at times not tracked at all, for example from t = 0.3 s to 0.7 s. This is likely due to the increase in frequency observed at that time. The first three harmonics can still be tracked because the frequency change is smaller at lower harmonics.

Figure 4.7a shows the resulting capturegram when 500 very closely spaced FLLs were used. The gammatone prefilters have been spaced linearly this time, i.e. the distance from one gammatone center frequency to the adjacent gammatone center frequency is constant. In addition to using more FLLs the figure was also created by using a narrower gammatone bandwidth. The effect of this choice is clearly visible, since now even the higher order harmonics are still tracked. Apparently, the gammatone bandwidth in figure 4.6 was so large that it spanned multiple harmonic frequencies and as a result, the FLLs could not “decide” which frequency to lock on. The tracking of higher frequencies in figure 4.7a is still not perfect, but certainly an improvement.

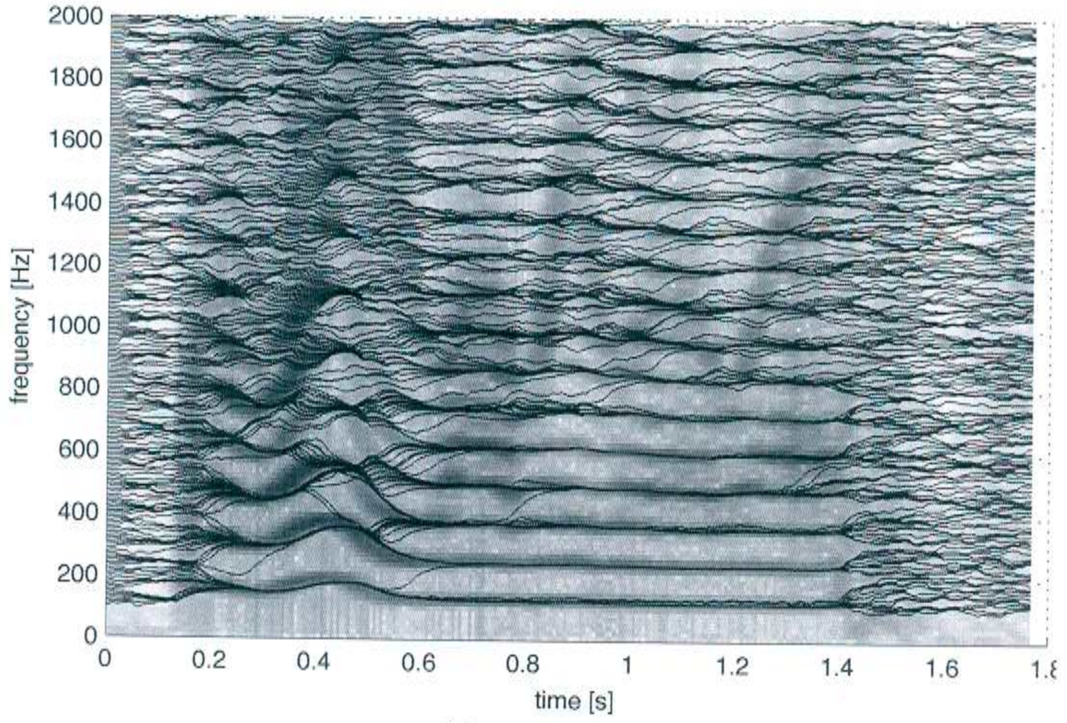
An even larger improvement can be seen on figure 4.7b. It has been achieved by using the so-called OPTICS clustering algorithm presented in [Ankerst99]. OPTICS is a density-based agglomerative clustering algorithm, which joins multiple data points together to a group not only based on their respective distance, but also based on the density around the data points. OPTICS has been applied to all frequencies that were found in each sample step. Since there are so many FLLs the chances are that multiple of them track the same frequency. The FLLs don't track the frequencies perfectly, i.e. their output values are slightly different, even when they essentially track the same frequency. When multiple FLLs put out a similar frequency value at a given moment in time, this is recognized by the OPTICS algorithm as a cluster, which it then merges to the mean of all frequencies in the cluster.

The cleanest tracking output has been obtained when a lower number of FLLs was used (approximately 100). This can be seen in figure 4.8. The upper figure depicts the raw tracking output, while the lower figure depicts the result of the OPTICS clustering. However, obtaining this particular figure needed a lot of parameter tuning (gammatone spacing, gammatone bandwidth, amount of FLLs, loop filter parameters, etc.). Unfortunately, the settings which worked well in this case turned out not to work in other cases and vice versa. Therefore, the FLL-based frequency tracking turned out not to be very robust. Additionally, all the figures in this section have been created using clean speech, while the results for reverberant speech were worse.

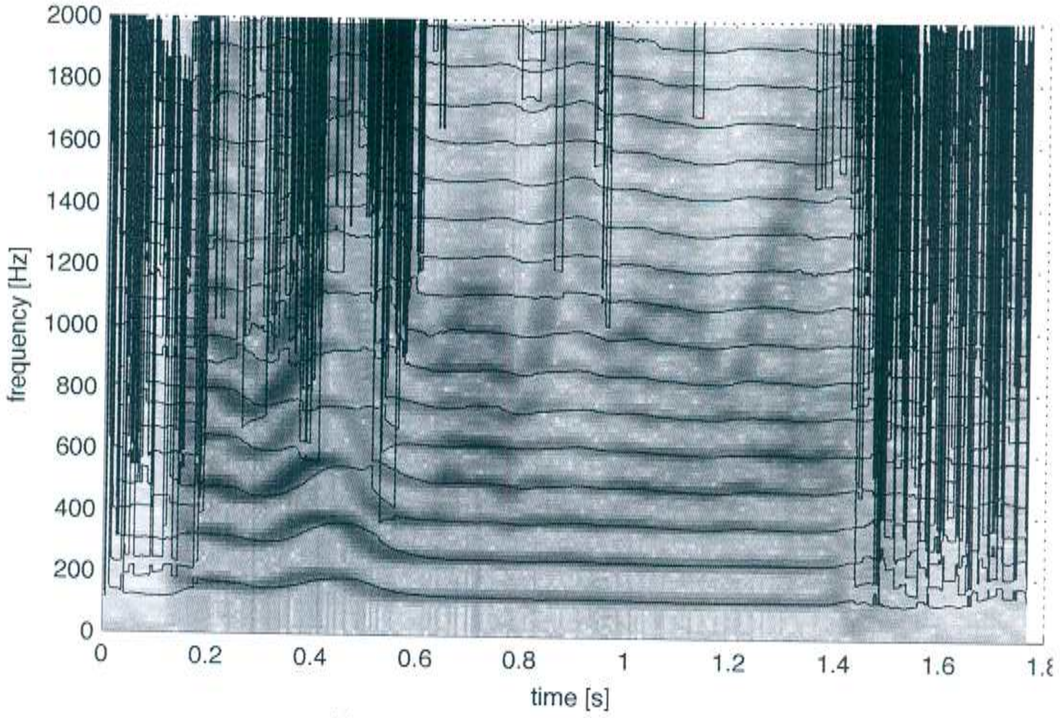
4.5 Dereverberation Experiments

This section contains the results of the final evaluation using the previously described speech recognition system. The recordings from the evaluation database described in section 4.1.1 have been convolved with the impulse responses described in section 4.1.2. Afterwards, the resulting .wav-files have been processed by the speech recognizer in order to obtain baseline recognition performance results.

Afterwards, two kinds of processing have been performed in order to improve the ASR performance by removing reverb from the recordings. The first one was a comb-filtering approach, where the voiced sections of each utterance were filtered using a feedback comb filter as described in section 3.4. The comb filter was supposed

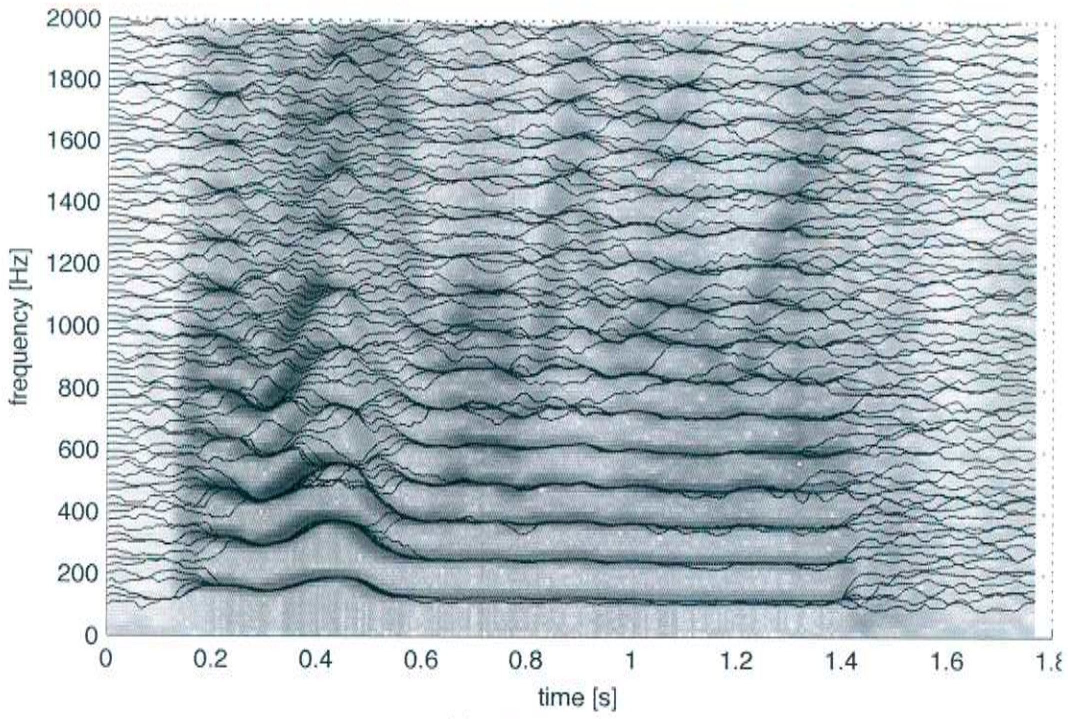


(a) raw output

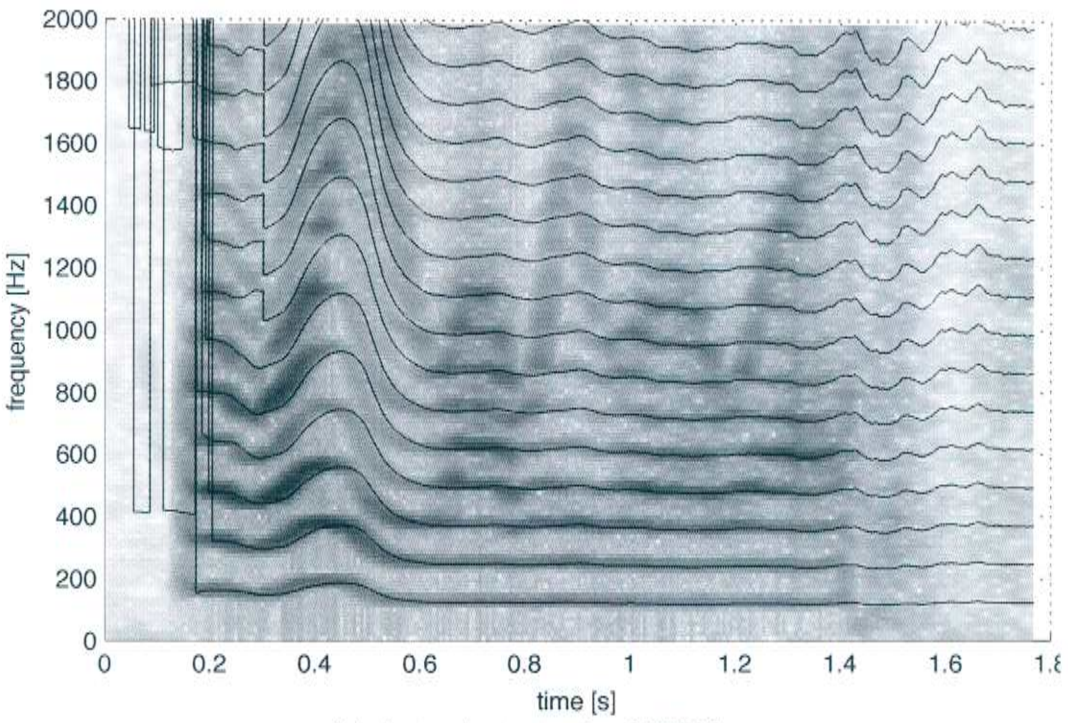


(b) clustered output using OPTICS

Figure 4.7: 500 linearly spaced Triplet-filter FLLs with narrow gammatone prefilter bandwidth.



(a) raw output



(b) clustered output using OPTICS

Figure 4.8: Linearly spaced Triplet-filter FLLs with narrow gammatone prefilter bandwidth.

to attenuate the frequency components between the harmonics of the actual speech and thus increasing the signal-to-noise ratio of the speech.

The second dereverberation method used APES in order to estimate the amplitude of each harmonic frequency component during the voiced parts of each utterance. Afterwards, the amplitude information was used to resynthesize the harmonic components. This means that the pitch track was used to generate a sinusoid for each harmonic, which was then subsequently multiplied by the amplitude estimate from APES.

Both of these processing techniques were first used for each utterance as a whole, but both the comb-filter and the APES approach only work on a given pitch, which has to be known a priori. During unvoiced sections of the recordings there is no pitch and therefore there are no distinct harmonics whose amplitudes could be tracked. Therefore, two sets of data have been generated for each processing technique in order to deal with the unvoiced speech segments. The first set of data was created by crossfading the original reverberated unvoiced sections into the comb-filtered and APES-processed output. As a result, the input for the ASR system was processed during the voiced sections but unprocessed during the unvoiced sections. For the second data set the unvoiced sections have not been replaced by the corresponding reverberated parts, but by the appropriate unreverberant parts which were taken from the original close-talk recording. This was done in order to study whether reverberated voiced or unvoiced speech has the most severe effect on ASR performance.

At the end, all of the aforementioned steps were repeated using synthetically generated speech by the synthesis software mentioned in section 4.1.3. The advantage of these artificially generated speech samples is that their pitch is known perfectly from the synthesis software, which means that pitch estimation errors can be eliminated. The synthesized speech samples will be denoted by the word "synthetic" in the tables of the following sections, whereas the normal recordings will be named "real".

4.5.1 Baseline

This section contains the aforementioned baseline speech recognition experiment results, i.e. the word error rates obtained without any further signal processing.

4.5.1.1 Reverberated Unvoiced Parts / Full Reverb

Table 4.1 lists baseline word error rates for the full reverb case, i.e. the unvoiced reverberated parts have not been replaced by their unvoiced close-talk equivalents. The negative effects of reverb can be seen clearly as the word error rate increases from 36.4 % to more than 80 % for the real recordings and from 20.7 % to more than 70 %, except for the 100 cm AIR impulse response, for the synthetic recordings. It is interesting that the 100 cm AIR recordings have a WER that is approximately 10 % better than the 200 cm recordings, whereas there is not so much difference between the 200 cm and 300 cm recordings.

A reason might be that 100 cm are still within the critical distance for the evaluated room, whereas both for 200 cm and 300 cm the sound pressure level of the reverb is possibly higher than the level of the direct sound. The same effect does not occur for the artificial Roomsim responses, but this might be due to the unrealistic assumptions of Roomsim, which does not consider any objects or people in its simulated rooms. Therefore, the reverberation time is longer than it would normally be for a room of a similar size and the reverb pressure level is higher, too, resulting in a reduced critical distance. If the critical distance actually is the reason for the drop of WER from 200 cm AIR to 100 cm AIR, a comparison of AIR and Roomsim WERs leads to the suggestion that all Roomsim recordings might be outside of the critical distance.

utterance	close	Roomsim			AIR		
type	talk	60 cm	120 cm	180 cm	100 cm	200 cm	300 cm
real	36.4	90.9	94.4	92.7	81.8	91.3	92.4
synthetic	20.7	78.7	83.4	77.1	58.3	72.0	72.5

Table 4.1: Word error rates for fully reverberated real and synthetic utterances at various recording distances using no further processing.

Another interesting detail is the improvement of the 180 cm Roomsim WER compared to the 120 cm Roomsim recordings. It is not clear why or how a greater distance contributes the a better ASR performance in this experiment. It is possible, however, that the position of microphone and speaker in the 120 cm setting are particularly bad, for example because of standing waves in the simulated room. If the relative position of speaker and microphone in the 180 cm setting are for some reason particularly good, it is conceivable that the ASR performance might be better despite the greater distance.

4.5.1.2 Non-Reverberated Unvoiced Parts

The word error rates in table 4.2 were obtained after replacing the reverberated unvoiced parts with unreverberant unvoiced parts from the close-talk signal. This was done to find out whether reverberated voiced and unvoiced speech parts contribute equally to the degradation of ASR performance or if voiced or unvoiced speech plays a greater role. Unsurprisingly, It can be seen that the word error rate using the unreverberant unvoiced parts is significantly lower than in the fully reverberated case from the previous section. This is evidence that all parts of speech have a similar influence on ASR performance

Interestingly, the 180 cm Roomsim recordings now have a better WER than both the 60 cm and 120 cm recordings, although of course, they all contain the same amount of voiced and unvoiced data. As a result, the replacement of reverberated unvoiced segments with corresponding unreverberant segments should basically have similar effects for all distances. The fact that the WER improvement in the voiced-only reverberb case was actually higher for the greater distance could mean that the degradation of unvoiced parts plays a greater role for the decreasing ASR performance than the degradation of the voiced parts.

A further analysis could be done to examine if the difference in word error rate from the fully reverberated case to the voiced-only-reverb case corresponds with the fraction of unvoiced vs. voiced speech.

The most important statement that can be read from the data is that even when the unvoiced speech segments are not reverberated, there is still a large improvement of approximately 30 % which could be obtained by dereverberating just the voiced parts.

utterance	close	Roomsim			AIR		
type	talk	60 cm	120 cm	180 cm	100 cm	200 cm	300 cm
real	36.4	67.0	69.3	65.8	59.0	65.9	65.0
synthetic	20.7	55.1	57.0	53.2	41.6	50.2	51.2

Table 4.2: Word error rates for partially reverberated real and synthetic utterances at various recording distances.

4.5.2 Suppression of Voiced Reverb using Comb-Filtering

The first dereverberation experiment was carried out using the comb-filter approach described in section 4.1.3. Pitch was obtained from the Histopitch pitch tracking algorithm and a feedback comb-filter was then used to attenuate frequency components between the harmonics of voiced speech.

4.5.2.1 Reverberated Unvoiced Parts

The results for the fully reverberant case, i.e. using reverberant unvoiced sections, are listed in table 4.3. They show, however, that this kind of processing had a negative effect on the WER for all examined recording distances.

A reason for this might be the problem of adaptive filter transients, which is described in [Välimäki98] and which results in audible clicks in the filter output. Adaptive filter transients occur when the coefficients of an adaptive feedback filter, like the comb filter in this experiment, are changed while its filter states are not updated accordingly at the same time. The problem does only occur for feedback filters (IIR) and not for feedforward (FIR) filters. The filter states in a feedback filter are constantly calculated based on the previous filter output, which is in turn calculated using the filter coefficients. If the filter coefficients had been equal to the new coefficients all of the time, then the values of the filter states would be different from what they are based on the old filter coefficients.

[Välimäki98] effectively proposes to go a sufficient amount of samples back in time after each coefficient change. This means that the values of the filter states are updated to the values that they would have had if the new coefficients had been in use for all of the time.

4.5.2.2 Non-Reverberated Unvoiced Parts

Table 4.4 contains the results for the recordings with comb-filtered voiced segments and unreverberant unvoiced segments. The results are similar to the case with reverberant unvoiced segments, i.e. the WER is worse after the processing compared to the baseline results.

utterance type	close talk	Roomsim			AIR		
		60 cm	120 cm	180 cm	100 cm	200 cm	300 cm
real (baseline)	36.4	90.9	94.4	92.7	81.8	91.3	92.4
real	44.8	94.8	95.8	96.0	86.6	94.0	94.7
synthetic (b.l.)	20.7	78.7	83.4	77.1	58.3	72.0	72.5
synthetic	26.6	86.4	89.6	86.3	65.1	81.5	81.3

Table 4.3: Word error rates for fully reverberated real and synthetic utterances at various recording distances after comb-filtering of voiced parts.

utterance type	close talk	Roomsim			AIR		
		60 cm	120 cm	180 cm	100 cm	200 cm	300 cm
real (baseline)	36.4	67.0	69.3	65.8	59.0	65.9	65.0
real	44.8	70.6	73.6	69.4	58.8	66.3	68.3
synthetic (b.l.)	20.7	55.1	57.0	53.2	41.6	50.2	51.2
synthetic	26.6	60.3	62.4	59.4	45.0	55.0	56.2

Table 4.4: Word error rates for partially reverberated real and synthetic utterances at various recording distances after comb-filtering of voiced parts.

4.5.3 Suppression of Voiced Reverb using APES-based Resynthesis

The final experiment was the WER calculation based on the APES-resynthesized waveforms.

4.5.3.1 Reverberated Unvoiced Parts

Starting with the waveforms with reverberant unvoiced parts, it is again obvious that the processing degrades the performance compared to the baseline system. However, the results are consistently better than those of the comb-filter processing. This might be due to the fact that the waveforms generated by the APES resynthesis only contain a certain amount (50) discrete harmonic frequencies and absolutely no noise, while the comb-filter output is more similar to a real speech recording, just with less noise.

utterance type	close talk	Roomsim			AIR		
		60 cm	120 cm	180 cm	100 cm	200 cm	300 cm
real (baseline)	36.4	90.9	94.4	92.7	81.8	91.3	92.4
real	44.7	93.5	95.4	93.6	85.5	92.7	93.3
synthetic (b.l.)	20.7	78.7	83.4	77.1	58.3	72.0	72.5
synthetic	45.7	86.3	90.2	86.6	73.1	82.6	83.6

Table 4.5: Word error rates for fully reverberated real and synthetic utterances at various recording distances after APES-based resynthesis of voiced parts.

4.5.3.2 Non-Reverberated Unvoiced Parts

Ultimately, the APES-based resynthesized waveforms have been crossfaded with the unreverberant close-talk unvoiced segments for the final experiment. The result for the 100 cm AIR recordings could be improved by 0.2 percentage points by the APES-processing. Given that all other recording distances showed worse results, it is likely safe to assume that this particular improvement is not statistically significant.

If anything, the fact that the 180 cm result in the unreverberant unvoiced case is once again better than the 120 cm result, whereas this has not been the case for the reverberant unvoiced data, might indicate that reverberation of unvoiced parts plays an increasingly larger role for larger distances. This result is similar to what has been observed from the baseline results in section 4.5.1.2 and also from the comb-filter data in section 4.5.2.2.

utterance	close	Roomsim			AIR		
type	talk	60 cm	120 cm	180 cm	100 cm	200 cm	300 cm
real (baseline)	36.4	67.0	69.3	65.8	59.0	65.9	65.0
real	44.7	72.7	75.3	70.9	62.9	69.4	71.0
synthetic (b.l.)	20.7	55.1	57.0	53.2	41.6	50.2	51.2
synthetic	45.7	73.3	76.4	71.8	65.6	70.7	70.9

Table 4.6: Word error rates for partially reverberated real and synthetic utterances at various recording distances after APES-based resynthesis of voiced parts.

5. Conclusion

The experiments conducted in this work have shown that the precise tracking of amplitude, phase and frequency is a difficult task. Many seemingly successful approaches, like the PLL-based amplitude, phase or impulse response estimation, only work in highly specific environments and under certain circumstances, for instance with sinusoids with a constant frequency.

The word error rate results from the dereverberation experiments also seem disappointing, but it still has to be examined whether the proposed comb-filter or APES-resynthesis approaches are generally not useful as a means of reverberation elimination or if they create a problem for a particular part of the subsequent ASR system.

While this might actually be the case for the APES-processed data it is surprising that the application of a comb-filter in order to enhance the harmonic components has such a negative effect on the ASR performance. The reason for the bad results could be the already mentioned adaptive recursive filter transients which the time-varying comb filter produces. The resulting discontinuities in the audio stream are clearly audible and they might be responsible for the bad ASR performance. This problem could be examined in future work.

After all it might also be reasonable to question the usefulness and practicability of the precise measurement approaches which have been evaluated in this work. It is likely more beneficial to pursue somewhat more statistical approaches which explicitly model the uncertainties involved in the measurement of amplitude, phase or frequency.

Bibliography

[Allen79] Jont B. Allen and David A. Berkley. "Image method for efficiently simulating small-room acoustics". In: *The Journal of the Acoustical Society of America* 65.4 (Apr. 1979), pp. 943–950. DOI: 10.1121/1.382599. URL: http://link.aip.org/link/?JAS/65/943/1.

[Ankerst99] Mihael Ankerst et al. "OPTICS: Ordering Points To Identify the Clustering Structure". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. Philadelpia, Pennsylvania, June 1999, pp. 49–60. DOI: 10.1145/304182.304187.

[Bagshaw93] P. C. Bagshaw, S. M. Hiller, and M. A. Jack. "Enhanced Pitch Tracking and the Processing of F0 Contours for Computer Aided Intonation Teaching". In: *Proceedings of the 3rd European Conference on Speech Communication and Technology (EUROSPEECH)*. Berlin, 1993, pp. 1003–1006.

[Best93] Roland Best. *Phase-locked loops: theory, design and applications*. 2nd ed. New York City, New York: McGraw-Hill, 1993. ISBN: 0-07-911386-9.

[Capon69] J. Capon. "High-Resolution Frequency-Wavenumber Spectrum Analysis". In: *Proceedings of the IEEE* 57.8 (Aug. 1969), pp. 1408–1418. DOI: 10.1109/PROC.1969.7278.

[Carlosena07] Alfonso Carlosena and Antonio Mànuel-Lázaro. "Design of High Order Phase-Lock Loops". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 54.1 (Jan. 2007), pp. 9 –13. DOI: 10.1109/TCSII.2006.883205.

[Carlosena08] Alfonso Carlosena and Antonio Mànuel-Lázaro. "General Method for Phase-Locked Loop Filter Analysis and Design". In: *IET Circuits, Devices and Systems* 2.2 (Apr. 2008), pp. 249 –256. DOI: 10.1049/iet-cds:20070065.

[Carlyon97] Robert P. Carlyon and A. Jaysurya Datta. "Masking period patterns of Schroeder-phase complexes: Effects of level, number of components, and phase of flanking components". In: *The Journal of the Acoustical Society of America* 101.6 (1997), pp. 3648–3657. DOI: 10.1121/1.418325.

[Costas80] John P. Costas. *Residual Signal Analysis - A Search and Destroy Approach to Spectral Analysis.* Tech. rep. Syracuse, New York: General Electric Military Electronic Systems Operation, Oct. 1980. URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a092968.pdf.

[Curran12] James T. Curran, Gérard Lachapelle, and Colin C. Murphy. "Improving the Design of Frequency Lock Loops for GNSS Receivers". In: *IEEE Transactions on Aerospace and Electronic Systems* 48.1 (Jan. 2012), pp. 850 –858. DOI: 10.1109/TAES.2012.6129674.

[Estienne01] Claudio Estienne, Patricia Pelle, and Juan Pablo Piantanida. "A Front-end for Speech Recognition Systems Using Phase-Locked Loops". In: *Proceedings of the Workshop in Information Processing and Control (RPIC).* Santa Fe, 2001, pp. 88–91.

[Gardner05] Floyd M. Gardner. *Phaselock techniques.* 3rd ed. Hoboken, New Jersey: Wiley-Interscience, 2005. ISBN: 978-0-471-43063-6.

[Garofolo93] John S. Garofolo et al. *TIMIT Acoustic-Phonetic Continuous Speech Corpus.* Tech. rep. Philadelphia: Linguistic Data Consortium, 1993.

[Glentis08] George-Othon Glentis. "A Fast Algorithm for APES and Capon Spectral Estimation". In: *IEEE Transactions on Signal Processing* 56.9 (Sept. 2008), pp. 4207 –4220. DOI: 10.1109/TSP.2008.925940.

[Holdsworth88] John Holdsworth et al. *Annex C of the SVOS final report: Implementing a GammaTone Filter Bank.* Tech. rep. Cambridge: University of Cambridge and Cambridge Electronic Design, June 1988.

[Huber11] Ralf Huber. "Combined Phase and Amplitude Analysis in Harmonic Acoustic Signals for Robust Speech Recognition". Karlsruhe: Karlsruhe Institute of Technology, Apr. 2011.

[Jeub09] Marco Jeub, Magnus Schäfer, and Peter Vary. "A binaural room impulse response database for the evaluation of dereverberation algorithms". In: *16th International Conference on Digital Signal Processing (DSP).* Santorini, July 2009, pp. 1–5. DOI: 10.1109/ICDSP.2009.5201259.

[Kaplan06] Elliott Kaplan and Christopher Hegarty. *Understanding GPS: Principles and Applications.* 2nd ed. Norwood, Massachusetts: Artech House, 2006. ISBN: 1-58053-894-0.

[Karimi-Ghartemani01] Masoud Karimi-Ghartemani and M. Reza Iravani. "A new phase-locked loop (PLL) system". In: *Proceedings of the IEEE Midwest Symposium on Circuits and Systems (MWSCAS).* Vol. 1. Dayton, Ohio, 2001, pp. 421–424. DOI: 10.1109/MWSCAS.2001.986202.

[Kinoshita05a] Keisuke Kinoshita, Tomohiro Nakatani, and Masato Miyoshi. "Efficient Blind Dereverberation Framework for Automatic Speech Recognition". In: *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*. 2005, pp. 3145–3148.

[Kinoshita05b] Keisuke Kinoshita, Tomohiro Nakatani, and Masato Miyoshi. "Fast Estimation of a Precise Dereverberation Filter based on Speech Harmonicity". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 1. 2005, pp. 1073–1076. DOI: 10.1109/ICASSP.2005.1415303.

[Kumaresan11] Ramdas Kumaresan, Vijay Kumar Peddinti, and Peter Cariani. "Multiple Pitch Identification Using Cochlear-Like Frequency Capture and Harmonic Grouping". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Prague, 2011, pp. 613–616. DOI: 10.1109/ICASSP.2011.5946478.

[Kumaresan12] Ramdas Kumaresan, Vijay Kumar Peddinti, and Peter Cariani. "Synchrony Capture Filterbank (SCFB): An Auditory Periphery Inspired Method for Tracking Sinusoids". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Mar. 2012, pp. 613–616. DOI: 10.1109/ICASSP.2011.5946478.

[Li96] Jian Li and Petre Stoica. "An Adaptive Filtering Approach to Spectral Estimation and SAR Imaging". In: *IEEE Transactions on Signal Processing* 44.6 (June 1996), pp. 1469–1484. DOI: 10.1109/78.506612.

[Li98] Hongbin Li, Jian Li, and Petre Stoica. "Performance Analysis of Forward-Backward Matched-Filterbank Spectral Estimators". In: *IEEE Transactions on Signal Processing* 46.7 (July 1998), pp. 1954–1966. DOI: 10.1109/78.700967.

[McAulay86] Robert McAulay and Thomas Quatieri. "Speech analysis/Synthesis based on a sinusoidal representation". In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 34.4 (1986), pp. 744–754. DOI: 10.1109/TASSP.1986.1164910.

[Nakatani03a] Tomohiro Nakatani and Masato Miyoshi. "Blind dereverberation of single channel speech signal based on harmonic structure". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 1. 2003, pp. 92–95. DOI: 10.1109/ICASSP.2003.1198724.

[Nakatani03b] Tomohiro Nakatani, Masato Miyoshi, and Keisuke Kinoshita. "Implementation and effects of single channel dereverberation based on the harmonic structure of speech". In: *Proceedings of the International Workshop on Acoustic Echo and Noise Control (IWAENC)*. Hong Kong, 2003, pp. 91–94.

[Nakatani07] Tomohiro Nakatani, Keisuke Kinoshita, and Masato Miyoshi. "Harmonicity Based Blind Dereverberation for Single-Channel Speech Signals". In: *IEEE Transactions on Audio, Speech and Language Processing* 15.1 (2007), pp. 80–95. DOI: 10.1109/TASL.2006.872620.

[Natali84] Francis D. Natali. "AFC Tracking Algorithms". In: *IEEE Transactions on Communications* 32.8 (Aug. 1984), pp. 935–947. DOI: 10.1109/TCOM.1984.1096152.

[Pelle03] Patricia Pelle and Matias Capeletto. "Pitch estimation using phase locked loops". In: *Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH)*. Geneva, 2003, pp. 2873–2876.

[Saratxaga09] Ibon Saratxaga et al. "Simple Representation of Signal Phase for Harmonic Speech Models". In: *Electronic Letters* 45.7 (Mar. 2009), pp. 381–383. DOI: 10.1049/el.2009.3328.

[Saratxaga10] Ibon Saratxaga et al. "Using Harmonic Phase Information to Improve ASR Rate". In: *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*. Makuhari, Japan, Sept. 2010, pp. 1185–1188. DOI: 10.1109/ICASSP.2011.5946478.

[Schroeder65] M. R. Schroeder. "New Method of Measuring Reverberation Time". In: *The Journal of the Acoustical Society of America* 37.6 (1965), pp. 1187–1188. DOI: 10.1121/1.1939454. URL: http://link.aip.org/link/?JAS/37/1187/5.

[Seltzer00] Michael L. Seltzer. "Automatic Detection of Corrupt Spectrographic Features for Robust Speech Recognition". MA thesis. Pittsburgh: Carnegie Mellon University, May 2000.

[Smith07] Julius O. Smith. *Introduction to Digital Filters with Audio Applications*. online book. 2007. URL: https://ccrma.stanford.edu/~jos/fp/Frequency_Response_I.html (visited on 08/11/2012).

[Smith10] Julius O. Smith. *Physical Audio Signal Processing*. online book. 2010. URL: https://ccrma.stanford.edu/~jos/pasp/Comb_Filters.html (visited on 08/11/2012).

[Soltau01] Hagen Soltau et al. "A one-pass decoder based on polymorphic linguistic context assignment". In: *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. Karlsruhe, Dec. 2001, pp. 214–217. DOI: 10.1109/ASRU.2001.1034625.

[Stephens02] Donald R. Stephens. *Phase locked loops for wireless communications: digital, analog and optical implementations.* 2nd ed. Boston, Massachusetts: Kluwer Academic Publishers, 2002. ISBN: 0-7923-7602-1.

[Stoica00] Petre Stoica, Hongbin Li, and Jian Li. "Amplitude Estimation of Sinusoidal Signals: Survey, New Results, and an Application". In: *IEEE Transactions on Signal Processing* 48.2 (Feb. 2000), pp. 338–352. DOI: 10.1109/78.823962.

[Stoica97] Petre Stoica, Andreas Jakobsson, and Jian Li. "Matched-Filter Bank Interpretation of Some Spectral Estimators". In: *Signal Processing* 66.1 (1997), pp. 45–59. DOI: 10.1016/S0165-1684(97)00239-9.

[Tiwari11] Sidhant Kumar Tiwari, Swarna Ravindra Babu, and R. Kumar. "Design of Baseband Processor for High Dynamic GPS Signals Using Higher Order Loops". In: *International Journal of Machine Learning and Computing* 1.5 (Dec. 2011), pp. 516 –521.

[Välimäki98] Vesa Välimäki and Timo I. Laakso. "Suppression of Transients in Time-Varying Recursive Filters for Audio Signals". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Vol. 6. Seattle, Washington, May 1998, pp. 3569–3572. DOI: 10.1109/ICASSP.2011.5946478.

[Wang94] Avery L. Wang. "Instantaneous and Frequency-Warped Signal Processing Techniques for Auditory Source Separation". PhD thesis. Stanford, California: Standford University, 1994.

[Wölfel09] Matthias Wölfel. "Robust Automatic Transcription of Lectures". PhD thesis. Karlsruhe: Karlsruhe Institute of Technology, 2009.