Modeling temporal characteristics of line spectral frequencies with an application to automatic speaker verification

Ville Vestman

Master's Thesis

UNIVERSITY OF
EASTERN FINLAND

Faculty of Science and Forestry

School of Computing

February 2016

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu
School of Computing
Computer Science

Student, Ville Vestman: Modeling temporal characteristics of line spectral frequencies with an application to automatic speaker verification
Master's Thesis, 77 p.
Supervisor of the Master's Thesis: Docent Tomi Kinnunen, Ph.D.
February 2016

Abstract:

Many speech processing systems rely on *short-term spectral features* that capture frequency information from 20–30 ms long speech segments with the help of *discrete Fourier transform* (DFT). These kind of features do not contain information about the dynamic properties of speech. In this thesis, we study long-term *spectro-temporal features* by modeling the temporal characteristics of so-called *line spectral frequencies* (LSFs). Before experimenting on these features, we review the theoretical background of the LSFs, especially that of *linear prediction* (LP). We also study the so-called *frequency domain linear prediction* (FDLP) method, which is utilized to model the temporal characteristics of the LSFs. We experimented on modeling LSF trajectories of individual words (about 400 ms in duration) with different modeling methods. We found that a simple discrete cosine transform (DCT) based method is more suitable than FDLP for LSF trajectory modeling. Then, we continued to evaluate the proposed spectro-temporal features against short-term spectral features. The speaker verification experiment was performed on the noise-free TIMIT corpus by using a Gaussian mixture model – universal background model (GMM-UBM) verification system. The results suggest that the LSF-based spectro-temporal features can bring improvements to verification results when used together with the short-term features. We were able to improve the *equal error rate* (EER) of widely used MFCC features from 0.19% down to 0.11% (relative decrease of 43.6%) by fusing the MFCCs with short- and long-term LSF features.

Keywords: speaker verification, spectro-temporal features, linear prediction, line spectral frequencies

CR Categories (ACM Computing Classification System, 2012 version):

•**Computing methodologies** → *Feature selection; Modeling methodologies;* Supervised learning by classification; •**Applied computing** → Sound and music computing;

Acknowledgments

I acknowledge, with gratitude, my supervisor Docent Tomi Kinnunen, Ph.D., for offering me an interesting research topic and for all the guidance and suggestions I got from him along the way. I would also like to acknowledge Academy Professor Paavo Alku from Aalto University for participating in the research-related brainstorming.

I am thankful to the University of Eastern Finland, especially the School of Computing, for providing me good working facilities. This thesis was financially supported by Academy of Finland, project no. 282118.

List of Abbreviations

ASV	Automatic speaker verification
DCT	Discrete cosine transform
DCT-FDLP	FDLP applied to DCT-signal
DET	Detection error tradeoff
DFT	Discrete Fourier transform
DTFT	Discrete-time Fourier transform
EER	Equal error rate
FDLP	Frequency domain linear prediction
FFT	Fast Fourier transform
GMM	Gaussian mixture model
IDCT	Inverse discrete cosine transform
IDFT	Inverse discrete Fourier transform
IDFT-FDLP	FDLP applied to IDFT-signal
LP	Linear prediction
LPC	Linear prediction coefficient
LSD	Log spectral distance
LSF	Line spectral frequency
LTI	Linear and time-invariant
MAE	Mean absolute error
MFCC	Mel frequency cepstral coefficient
RMSE	Root mean squared error
UBM	Universal background model

Contents

1 Introduction **1**

2 Basics of digital speech signal processing **4**

2.1 Digital speech signal 4

2.2 Discrete Fourier transform 5

2.3 Z-transform 8

2.4 Spectrograms and windowing 10

2.5 Digital filters 13

3 Linear prediction and spectral envelope modeling **15**

3.1 Spectral envelope modeling 16

3.2 Error minimization 18

3.3 Autocorrelation method 20

3.4 Gain computation 21

3.5 Summary: computing spectral estimates 23

3.6 Pole properties 24

3.7 Line spectral frequencies 26

4 Linear prediction and modeling of time domain signals and their envelopes **30**

4.1 Hilbert transforms, analytic signals and Hilbert envelopes 30

4.2 Symmetrized and zero-padded signals 33

4.3 Discrete cosine transform 35

4.4 Subband decomposition using the DCT 37

4.5 Envelope modeling with DCT-FDLP 41

4.6 Envelope modeling with IDFT-FDLP 43

4.7 FDLP-modeling of positive valued signals 45

5 LSF track modeling experiments **48**

5.1 LSF track fitting with DCT 49

5.2 LSF track fitting with DCT-FDLP 50

5.3 LSF track fitting with IDFT-FDLP 53

5.4 Error measures 55

5.5 Experiment design 59

5.6 Results and discussion 60

6 Application to automatic speaker verification **62**
6.1 Feature vector extraction 62
6.2 Experimental set-up 65
6.3 Parameter choices 66
6.4 Baseline short-term features 70
6.5 Comparison of ASV systems 72

7 Conclusions **74**

1 Introduction

The era of electronics has brought up many new research topics related to human speech. One such topic is *automatic speaker recognition*. This problem can be approached in many ways. In *text-dependent* speaker recognition, the speaker is required or assumed to speak exactly the phrases given beforehand, while in *text-independent* speaker recognition, speaker is allowed to speak freely [17]. Text-independent speaker recognition is generally more challenging due to larger variability in the speech.

In many applications, for example in security, it is enough to verify that the speaker is who he or she claims to be. This is known as *automatic speaker verification* (ASV) or *speaker authentication*. On the other hand, in *speaker identification* (SID), we attempt to determine who the speaker is by finding the closest match from a previously collected database. This could be applied, for instance, in searching speakers from Internet resources. In this thesis, we focus on the ASV task (even though the studied methods are expected, to a certain extent, to generalize to SID tasks as well).

Typical ASV systems consist of two main components: *feature extractor* (front-end) and *classifier* (back-end). Feature extraction, the core focus of this thesis, is a process of generating *feature vectors* from a speech signal. There are many possible ways to define and compute feature vectors from speech. In [17], features are categorized in the following five groups: (1) *short-term spectral features*, (2) *voice source features*, (3) *spectro-temporal features*, (4) *prosodic features*, and (5) *high-level features*. The first group, short-term spectral features, are obtained from 20 to 30 milliseconds long segments of speech, and are used to capture frequency information from instantaneous time intervals. To capture information about the dynamic properties of speech, for example intonation and rhythm, we need to use longer segments of speech, and this is where the spectro-temporal features come into play. Spectro-temporal features can span over a few hundreds milliseconds of speech. In this thesis, we study only the short-term and spectro-temporal features.

Before the speaker can be recognized by an ASV system, the speaker must be *enrolled* to the system by providing some sample speech from the speaker. This speech is used to *train* a *speaker model*. In the actual *recognition phase*, a set of feature vectors extracted from the new speech is evaluated against the enrolled speaker models to identify or verify the speaker. The speaker models and evaluations against the speech

features are handled by the back-end classifier. In order to make the classifier perform well, it is necessary to craft such feature vectors that the feature distributions of the different speakers are as diverse as possible.

In this thesis, we propose a method to form spectro-temporal features by modeling the temporal characteristics of so-called *line spectral frequencies* (LSFs) [15, 28, 30]. The idea is similar in spirit to the one presented in [16], in which the temporal characteristics of the *mel frequency cepstral coefficients* (MFCCs) were modeled instead of the LSFs.

The LSFs are an alternative and mathematically equivalent representation for the widely-used *linear prediction coefficients* (LPCs) [20]. We cover the subject of *linear prediction* (LP) in detail. We also detail a related technique known as *frequency domain linear prediction* (FDLP) that has been previously used for feature extraction, for example, in [2] and [9]. These studies apply FDLP to model separate spectral subbands of a speech. In this thesis, instead of modeling the subbands of a speech, we investigate how FDLP can be used to model the temporal characteristics of the LSFs instead.

The following two chapters are quite mathematically oriented. For the reader it is enough to have a basic level understanding in mathematics, specifically in linear algebra, complex variables, and differential calculus. It is also beneficial, but not necessary, to have prior knowledge about the discrete Fourier and cosine transforms. Chapter 2 gives a brief introduction to the basics of the digital speech signal processing, whereas in Chapter 3, we focus on the use of the linear prediction for modeling spectral envelopes. The LSFs will be introduced at the end of Chapter 3.

In Chapter 4, we partly present the theoretical background of FDLP, which includes, for instance, the subjects of *Hilbert transforms* and *Hilbert envelopes*. In [2], [3], and [9], FDLP has been applied to a frequency domain signal obtained using the *discrete cosine transform* (DCT). In addition to this way of using FDLP, we study how FDLP functions when it is applied to a *inverse Fourier transformed* signal instead. The DCT based FDLP is used to model the Hilbert envelope of a signal, while the use of inverse Fourier transform provides a way to model the absolute value of a signal.

Typically, the FDLP method has been used to model temporal envelopes of signals. In this thesis, we also use FDLP to model *positive valued* signals and not just their envelopes. The need to model positive valued signals rises from the fact that the temporal

trajectories of LSFs are positive valued.

In Chapter 5, we experiment on the use of the DCT along with the different FDLP variants in modeling the temporal characteristics of the LSFs. Finally, in Chapter 6, we apply the studied techniques in a text-independent speaker verification application. Section 7 concludes the thesis.

2 Basics of digital speech signal processing

In this chapter, we give a brief introduction to digital speech signal processing. We begin by introducing two important mathematical tools, the Fourier and the Z-transforms. The last part of the chapter focuses on digital filters. Along the whole chapter, we define some of the most commonly used terms in the field.

2.1 Digital speech signal

What we perceive as a sound, is in its physical sense rapidly occurring pressure changes in the surrounding medium, typically air. We use microphones to convert these pressure changes to voltage changes. The resulting analog (continuous) electrical signal has then to be converted to a digital (discrete) form to allow digital processing of the sound. This process, *sampling*, is carried out by an analog-to-digital converter. *Sampling frequency* determines how many discrete values are created from the continuous signal per second.

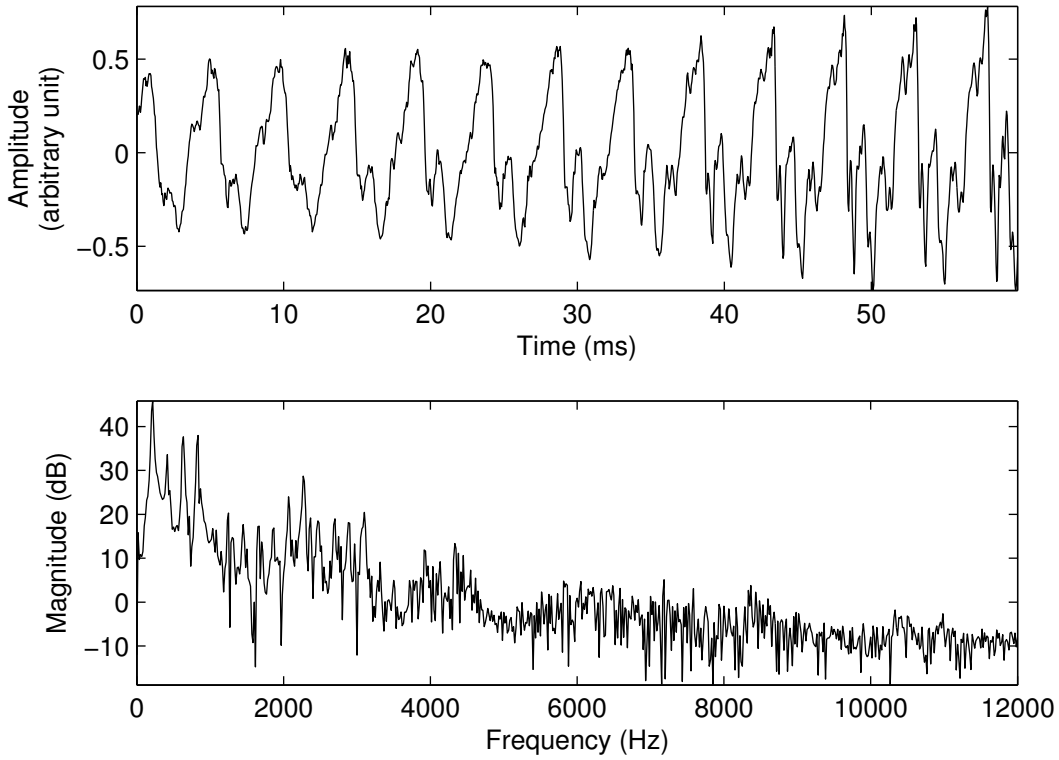

Figure 1: A short segment of a vowel sound /æ/ represented in the time and frequency domains.

After the sampling process we are left with a finite-length time-domain sequence. We use the notation $x[t]$ to represent a value of a signal x at a moment t.

By using the Fourier transform, we are able to obtain information about the frequencies in the signal. Fig. 1 shows an example of a speech signal represented in the time and frequency domains. Both representations contain useful information about the speech or any other form of sound. The time domain representation displays the pressure levels at any given time. The frequency domain view informs us about the distribution of signal energy to different frequency components.

2.2 Discrete Fourier transform

The Fourier transform (originally formulated by Fourier in early 1800s [18, pp. 478–479]) and some other similar transforms are very important mathematical tools in the field of signal processing. These transforms have a capability of revealing the frequency content of the signal.

Definition 2.1 (Discrete Fourier transform). Let $x[n], n = 0, \ldots, N-1$, be a sequence of complex (or real) numbers. Then its *discrete Fourier transform* (DFT) is defined as a complex valued sequence

$$\mathrm{DFT}\{x[n]\} = X[k] = \sum_{n=0}^{N-1} x[n] e^{-\mathrm{i}\frac{2\pi kn}{N}}, \qquad k = 0, \ldots, N-1, \tag{1}$$

where i is the imaginary unit. The sequence $x[n]$ is called the *inverse discrete Fourier transform* (IDFT) of the sequence $X[k]$.

Theorem 2.2 (Inverse DFT). *Let a sequence $x[n]$ be the IDFT of a sequence $X[k]$. Then*

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{\mathrm{i}\frac{2\pi nk}{N}}, \qquad n = 0, \ldots, N-1. \tag{2}$$

Proof. By using the definition of the DFT and by interchanging the order of summation

we get

$$\frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{\mathrm{i}\frac{2\pi nk}{N}} = \frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{m=0}^{N-1}\left(x[m]e^{-\mathrm{i}\frac{2\pi km}{N}}\right)e^{\mathrm{i}\frac{2\pi nk}{N}}\right)$$
$$= \frac{1}{N}\sum_{m=0}^{N-1}\left(x[m]\sum_{k=0}^{N-1} e^{\mathrm{i}\frac{2\pi(n-m)k}{N}}\right).$$

The result follows from the fact that

$$\sum_{k=0}^{N-1} e^{\mathrm{i}\frac{2\pi(n-m)k}{N}} = \begin{cases} N, & \text{when } n = m, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The case $n = m$ is trivial, and the case $n \neq m$ can be shown by using the summation formula

$$\sum_{k=0}^{N-1} a^k = \frac{1-a^N}{1-a} \qquad (a \neq 1)$$

for geometric series. By applying this formula we get

$$\sum_{k=0}^{N-1} e^{\mathrm{i}\frac{2\pi(n-m)k}{N}} = \sum_{k=0}^{N-1}\left(e^{(\mathrm{i}2\pi(n-m)/N)}\right)^k = \frac{1-\left(e^{(\mathrm{i}2\pi(n-m)/N)}\right)^N}{1-e^{(\mathrm{i}2\pi(n-m)/N)}}$$
$$= \frac{1-e^{\mathrm{i}2\pi(n-m)}}{1-e^{(\mathrm{i}2\pi(n-m)/N)}}$$
$$= \frac{1-1}{1-e^{(\mathrm{i}2\pi(n-m)/N)}} = 0. \qquad \square$$

Equation (2) tells us that any sequence $x[n]$ can be represented as a linear combination of (sampled) complex exponential functions. So, when we discuss the frequencies of a signal, we therefore actually refer to the exponential components with the corresponding frequencies.

The constants of the linear combination are complex values and they can be represented in a form

$$X[k] = |X[k]|e^{\mathrm{i}\arg(X[k])}, \tag{4}$$

where $|X[k]|$ and $\arg(X[k])$ are the magnitude and the phase of a complex number $X[k]$, respectively. The sequences $|X[k]|$ and $\arg(X[k])$, $k = 0, \ldots, N-1$, are called the *magnitude spectrum* and the *phase spectrum*, respectively. The squared magnitude spectrum $|X[k]|^2$ is called the *power spectrum*.

Using (4), Equation (2) can be rewritten as

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1}|X[k]|e^{\mathrm{i}\left(\frac{2\pi nk}{N}+\arg(X[k])\right)}, \qquad n = 0, \ldots, N-1. \tag{5}$$

When the sequence $x[n]$ is real-valued, (5) has an alternative form

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1}|X[k]|\cos\left(\frac{2\pi nk}{N}+\arg(X[k])\right), \qquad n = 0, \ldots, N-1.$$

This form can be justified with the Euler's formula $e^{\mathrm{i}\omega} = \cos(\omega) + \mathrm{i}\sin(\omega)$. Since $x[n]$ has no imaginary part, we are only left with the cosine terms. Similarly, from the Euler's formula and from the properties of sine and cosine, it follows that

$$\mathrm{Re}(X[k]) = \mathrm{Re}(X[N-k]) \text{ and} \tag{6}$$
$$\mathrm{Im}(X[k]) = -\mathrm{Im}(X[N-k]) \tag{7}$$

and therefore

$$|X[k]| = |X[N-k]| \text{ and}$$
$$\arg(X[k]) = -\arg(X[N-k])$$

for all $k = 1, \ldots, N-1$. Because of the above symmetries, it is apparent that we get information only about $N/2$ frequencies by using the DFT. Second half of the transform represents the so-called *negative frequencies*.

In Fig. 1, only first half of the magnitude spectrum is drawn. Frequencies in the x-axis are obtained from the DFT-sequence indices by using the formula $f = kf_s/N$, where f_s is the sampling frequency. This formula can be deduced by noting that in (5) kth exponential component has a frequency of k cycles per N samples. When this is multiplied with the sampling frequency, the amount of cycles per second is obtained.

We end this section with a theorem that we will need later:

Theorem 2.3 (Parseval's theorem). *If $X[k]$ is the DFT of $x[n]$, then*

$$\sum_{n=0}^{N-1}|x[n]|^2 = \frac{1}{N}\sum_{k=0}^{N-1}|X[k]|^2.$$

Proof. For any complex number c, we have $|c|^2 = c\bar{c}$, where $\bar{c}$ is the complex conjugate of c. Therefore,

$$\begin{aligned}
\frac{1}{N}\sum_{k=0}^{N-1}|X[k]|^2 &= \frac{1}{N}\sum_{k=0}^{N-1} X[k]\overline{X[k]} \\
&= \frac{1}{N}\sum_{k=0}^{N-1}\sum_{n=0}^{N-1} x[n]e^{-\mathrm{i}\frac{2\pi kn}{N}}\sum_{m=0}^{N-1}\overline{x[m]}e^{\mathrm{i}\frac{2\pi km}{N}} \\
&= \frac{1}{N}\sum_{k=0}^{N-1}\sum_{n=0}^{N-1} x[n]\sum_{m=0}^{N-1}\overline{x[m]}e^{\mathrm{i}\frac{2\pi (m-n)k}{N}} \\
&= \frac{1}{N}\sum_{n=0}^{N-1} x[n]\sum_{m=0}^{N-1}\overline{x[m]}\sum_{k=0}^{N-1}e^{\mathrm{i}\frac{2\pi (m-n)k}{N}} = \sum_{n=0}^{N-1} x[n]\overline{x[n]} = \sum_{n=0}^{N-1}|x[n]|^2,
\end{aligned}$$

where the next-to-last equality follows from (3). □

2.3 Z-transform

The Z-transform maps a sequence of numbers into a function of a complex variable z. The Z-transform can be thought as a generalized version of the *discrete-time Fourier transform* (DTFT), in which the complex variable is limited to the values $z = e^{i\omega}$, i.e. to the values of the complex unit circle. The Z-transform offers some notational convenience over the Fourier transform when manipulating recurrence relations.

Definition 2.4 (Z-transform). Let $x[n]$ be a two-way infinite sequence of complex numbers. Then its *Z-transform* is defined as a function

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n},$$

where z is a complex number.

To make this definition to applicable for finite sequences, the signal $x[n]$ can be assumed to equal zero outside of its original domain. By setting $z = e^{\mathrm{i}\omega}$, the Z-transform reduces to the DTFT

$$X(e^{\mathrm{i}\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-\mathrm{i}\omega n}.$$

Furthermore, if $x[n]$ is zero when n is outside of the range $n = 0, \ldots, N-1$, then the

DTFT reduces to the DFT by letting ω to have only values $\omega = 2\pi k/N$, where k runs from zero to $N - 1$.

The Z-transform is clearly well defined for the sequences that have a finite number of non-zero elements. For the other sequences the convergence of the series has to be ensured.

The rest of this section is dedicated to the properties of the Z-transform that we will use later on. We assume that every Z-transform that appears in the following theorems is convergent without any further notice. In practice, all signals we process are finite and thus their Z-transforms always exist.

Theorem 2.5 (Linearity of the Z-transform). *Let $v[n] = ax[n] + by[n]$, where $x[n]$ and $y[n]$ are sequences and a and b are constants. Then $V(z) = aX(z)+bY(z)$, where $V(z)$, $X(z)$ and $Y(z)$ are the Z-transforms of $v[n]$, $x[n]$ and $y[n]$, respectively.*

Proof.

$$\begin{aligned} V(z) = \sum_{n=-\infty}^{\infty} v[n]z^{-n} &= \sum_{n=-\infty}^{\infty} (ax[n] + by[n])z^{-n} \\ &= a \sum_{n=-\infty}^{\infty} x[n]z^{-n} + b \sum_{n=-\infty}^{\infty} y[n]z^{-n} \\ &= aX(z) + bY(z). \end{aligned}$$

□

Theorem 2.6 (Shifting property). *Let $y[n] = x[n - k]$ for some integer k. Then $Y(z) = z^{-k}X(z)$, where $X(z)$ and $Y(z)$ are the Z-transforms of $x[n]$ and $y[n]$, respectively.*

Proof.

$$Y(z) = \sum_{n=-\infty}^{\infty} x[n-k]z^{-n} = \sum_{m=-\infty}^{\infty} x[m]z^{-m-k} = z^{-k} \sum_{m=-\infty}^{\infty} x[m]z^{-m} = z^{-k}X(z).$$

□

Theorem 2.7 (Convolution theorem). *Let $v[n]$ be the convolution of sequences $x[n]$ and $y[n]$. That is, $v[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k]$. Additionally, let $V(z)$, $X(z)$ and $Y(z)$ be the Z-transforms of $v[n]$, $x[n]$ and $y[n]$, respectively. If $x[n]$ and $y[n]$ contain only a finite number of non-zero elements, then $V(z) = X(z)Y(z)$.*

Proof. According to the definitions of the Z-transform and convolution

$$V(z) = \sum_{n=-\infty}^{\infty} v[n]z^{-n} = \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k]y[n-k]z^{-n}.$$

If $x[n]$ and $y[n]$ contain only a finite number of non-zero elements, we can change the order of the summation, since in that case the above sums become finite. After changing the order of the summation, we modify the obtained expression to arrive to the desired result:

$$\begin{aligned} V(z) &= \sum_{k=-\infty}^{\infty} x[k] \sum_{n=-\infty}^{\infty} y[n-k]z^{-n} \\ &= \sum_{k=-\infty}^{\infty} x[k] \sum_{m=-\infty}^{\infty} y[m]z^{-m-k} \\ &= \sum_{k=-\infty}^{\infty} x[k]z^{-k} \sum_{m=-\infty}^{\infty} y[m]z^{-m} \\ &= \left(\sum_{k=-\infty}^{\infty} x[k]z^{-k} \right) \left(\sum_{m=-\infty}^{\infty} y[m]z^{-m} \right) = X(z)Y(z). \end{aligned}$$

□

2.4 Spectrograms and windowing

The DFT of an N-length signal $x[n]$ divides the signal into N frequency components that are N-periodic. If the signal $x[n]$ contains frequency components that are not N-periodic, the contributions of these frequencies will be spread out to all other frequencies. This phenomenon, known as *spectral leakage* [12], is demonstrated in Fig. 2. This figure shows two sinusoids, both having a frequency of 100 Hz, along with their magnitude spectra. The upper sinusoid has an integer number of cycles, resulting in a periodic signal and in a clean magnitude spectrum spiking only at 100 Hz. The lower sinusoid, in turn, is not periodic in the given interval, so the magnitude spectrum spreads out over the whole frequency range.

To reduce spectral leakage, the signal is often multiplied by a window function. Window functions typically approach zero value at the boundaries while obtaining maximum value of one in the center of the window. Windowing diminishes the discontinuity between the last and the first value of the given signal by tapering the signal towards zero at the boundaries. Fig. 3 shows a windowed version of the sinusoid in Fig. 2 along

with its magnitude spectrum. By comparing the magnitude spectra in Figs. 2 and 3, we find that the frequencies of the windowed version are located more compactly around 100 Hz, but the spectral peak of the windowed singal is slightly wider (extending to 150 Hz).

There exists a wide variety of window functions each of which has their advantages and disadvantages. A detailed review of window functions is provided by Harris [12].

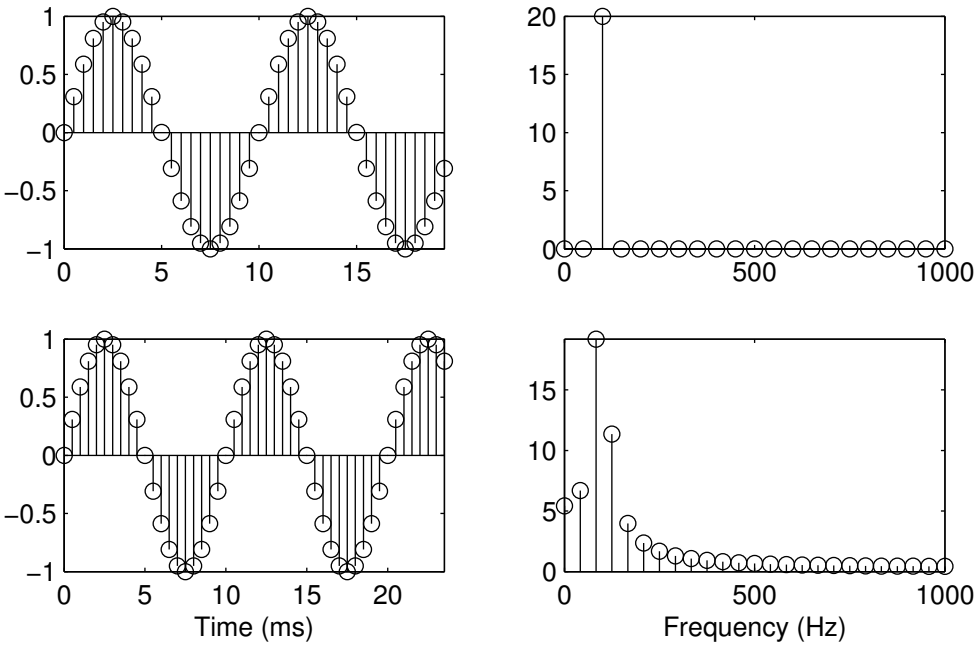

Figure 2: Two discrete 100 Hz sinusoids (on the left) together with their magnitude spectra (on the right). The lower sinusoid has a non-integer number of periods and this leads to a spectrum that is spread out to a wider range of frequencies.

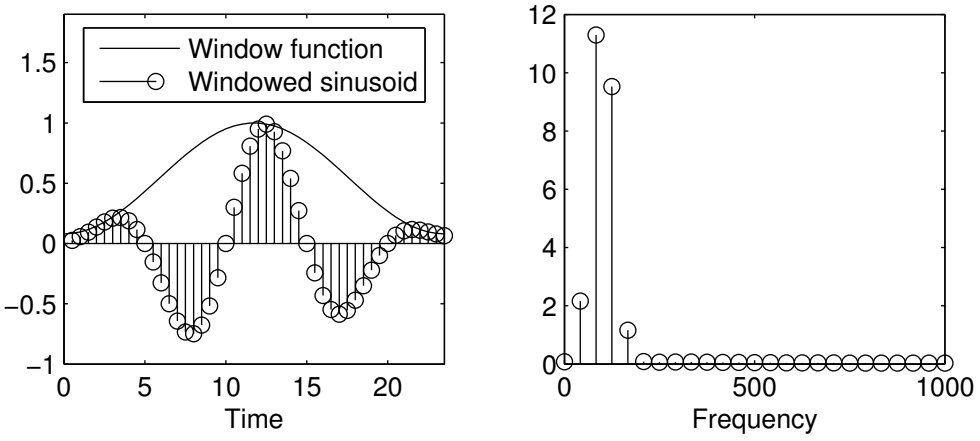

Figure 3: Windowed version of the lower sinusoid in the Fig. 2 and the resulting spectrum.

The magnitude spectrum of a long speech is not informative about the speech since it does not contain any information about the time-varying nature of the signal. For example, an individual phone in a speech utterance might last only for a few milliseconds. The magnitude spectrum of a longer signal than that would then describe a mixture of different phones and therefore, time-localized information about the individual phones would not be obtained. To this end, the signal is typically divided into segments that are 10 - 50 milliseconds long. These segments are called *frames*. When computing magnitude spectra of all of these frames we obtain data that has three dimensions: frequencies, their corresponding magnitudes and the time instants of the frames. A graphical display of this data is known as a *spectrogram*. An example of a spectrogram is shown in Fig. 4. The dark lines in the spectrogram show the locations of the spectral peaks as a function of time.

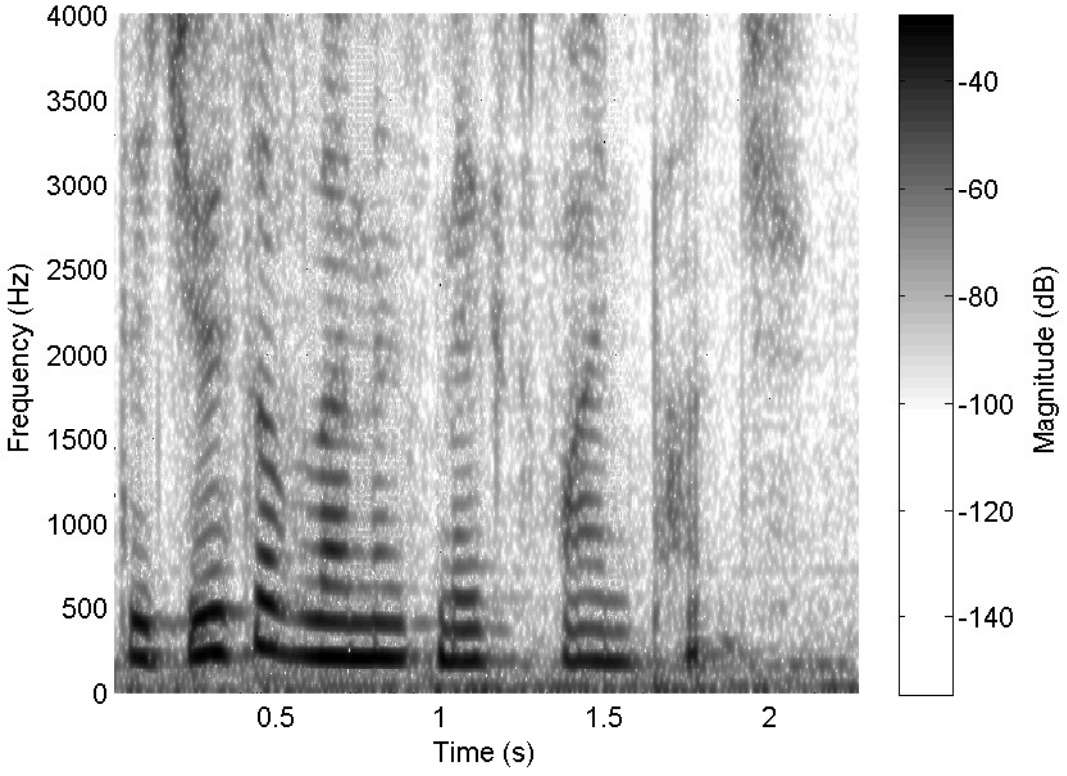

Figure 4: Spectrogram of a speech segment.

When creating a spectrogram, frames are usually windowed before computing magnitude spectra. Since windowing dampens values near the borders of the frame, it is possible to lose some data. This problem can be avoided by dividing the signal into the frames so that the adjacent frames overlap each other.

2.5 Digital filters

The purpose of filtering is to change the characteristics of a signal in a desired way. For example, it is possible to dampen or amplify selected frequency components.

A digital filter is based on a mathematical rule and it can be represented as an operator that processes the input signal to produce an output signal. For example, a filter $\mathcal{O}$ could be defined as

$$\mathcal{O}\{x[n]\} = y[n] = 0.45x[n-1] + 0.50x[n],$$

where $x[n]$ is the input and $\mathcal{O}\{x[n]\}$ is the output. Here $y[n]$ is used to emphasize that the output is also a sequence. This filter forms the output values by calculating the linear combinations of two consecutive input values.

The *impulse response* of a filter $\mathcal{O}$ is defined to be $\mathcal{O}\{\delta[n]\}$, where

$$\delta[n] = \begin{cases} 1, & \text{when } n = 0, \\ 0, & \text{otherwise.} \end{cases}$$

By knowing the impulse response of a linear and time-invariant (LTI) (defined below) filter, we can perform filtering by convolving the input signal with the impulse response of the filter (proof below).

Definition 2.8 (Linearity and time-invariance of filters). A filter $\mathcal{O}$ is *linear* if

$$\mathcal{O}\{ax_1[n] + bx_2[n]\} = a\mathcal{O}\{x_1[n]\} + b\mathcal{O}\{x_2[n]\}$$

for all sequences $x_1[n]$ and $x_2[n]$ and for all constants a and b.

Let $x[n]$ be the input of a filter and let $y[n]$ be the corresponding output. If the output for a shifted input sequence $x[n-k]$ is $y[n-k]$ for any input $x[n]$ and for any integer k, then the filter is *time-invariant*.

Theorem 2.9. *Let $h[n]$ be the impulse response of a LTI filter $\mathcal{O}$. Then*

$$\mathcal{O}\{x[n]\} = \sum_{k=-\infty}^{\infty} x[k]h[n-k]. \tag{8}$$

Proof. The key to proving this is to notice that any discrete signal can be written as a

sum of shifted impulses as follows:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k].$$

Now, assuming that the linearity property of $\mathcal{O}$ holds also in the case of above infinite sum, we get

$$\mathcal{O}\{x[n]\} = \mathcal{O}\left\{\sum_{k=-\infty}^{\infty} x[k]\delta[n-k]\right\} = \sum_{k=-\infty}^{\infty} x[k]\mathcal{O}\{\delta[n-k]\}.$$

The desired result follows from the time-invariance property of the filter. □

The Z-transform of the impulse response is called the *transfer function*. By denoting the output $\mathcal{O}\{x[n]\}$ in (8) by $y[n]$ and by using Theorem 2.7, the transfer function can be represented as

$$Y(z) = X(z)H(z) \Leftrightarrow H(z) = \frac{Y(z)}{X(z)},$$

where $Y(z)$, $X(z)$ and $H(z)$ are the Z-transforms of $y[n]$, $x[n]$ and $h[n]$, respectively.

The DTFT of the impulse response, i.e. $H(e^{\mathrm{i}\omega})$, is called the *frequency response* of the filter.

3 Linear prediction and spectral envelope modeling

According to Markel's and Gray's [21, p. 10] understanding, the first use of the term *linear prediction* (LP) appeared in Wiener's book "Extrapolation, Interpolation, and Smoothing of Stationary Time Series" published in 1949. The LP-technique was first used in speech analysis and synthesis in the late 1960's [21, p. 10]. For further reading, the book [21, pp. 18–20] provides a historical overview about the early days of LP and how different authors contributed to the development of the LP-theory.

The LP-theory given in this chapter is written with the help of [21, pp. 10–15] and Makhoul's tutorial review on LP [20]. While these sources present the LP-theory only for real valued signals, we extend the formulation further for complex-valued signals. According to our understanding, the complex valued LP is not so often seen in the field of speech processing, but it has been used in other applications, for example in temperature forecasting [11] and shape recognition [26]. We provide a derivation of the model coefficients in complex LP that is more detailed than what is given in [26] but simpler than what is presented in [11].

The LP-model can be presented as follows: Let us have a complex-valued signal $x[n]$, $n = 0, \ldots, N-1$, of observed values. Then, predicted samples, $\hat{x}[n]$, are obtained using the formula

$$\hat{x}[n] = \sum_{l=1}^{p} a[l]x[n-l], \tag{9}$$

where the complex values $a[l]$, $l = 1, \ldots, p$, are the *predictor coefficients*. The number p determines the amount of predictor coefficients and is called the *model order*.

In principle, the predictor coefficients can have any values, but naturally one would like to choose these coefficients so that the predicted signal $\hat{x}[n]$ would be as similar as possible to the observed signal $x[n]$. This is typically done by minimizing the error signal or *residual*

$$e[n] = x[n] - \hat{x}[n] \tag{10}$$

in the least squares sense. There exists (at least) two different variations of this minimization process, which are known as the *covariance method* and as the *autocorrelation method*. We will discuss both methods in this chapter.

Throughout the chapter we handle sequences that are defined in a finite interval. How-

ever, on some occasions values will be referred outside the given domain by assuming zero values outside the original domain. In fact, we already used this convention in (9), in which $n - l$ can be negative and therefore not in the range from zero to $N - 1$.

Before proceeding to the least squares minimization, let us first see how the LP-model can be utilized to obtain all-pole spectrum estimates.

3.1 Spectral envelope modeling

With the help of the LP-model, we can formulate a new expression for the magnitude spectrum of the observed signal $x[n]$. By substituting (9) into (10), we get

$$e[n] = x[n] - \sum_{l=1}^{p} a[l]x[n-l]. \tag{11}$$

According to the theorems 2.5 and 2.6, the Z-transform of (11) is

$$E(z) = X(z) - \sum_{l=1}^{p} a[l]z^{-l}X(z),$$

where $E(z)$ and $X(z)$ are the Z-transforms of $e[n]$ and $x[n]$, respectively. This equation can be rewritten as

$$X(z) = \frac{E(z)}{1 - \sum_{l=1}^{p} a[l]z^{-l}}. \tag{12}$$

The variable z in (12) is continuous. By evaluating (12) at the uniformly spaced points $z = e^{\mathrm{i}2\pi k/N}$, $k = 0, \ldots, N-1$ (on the unit circle), we get

$$X[k] = \frac{E[k]}{1 - \sum_{l=1}^{p} a[l]e^{-\mathrm{i}2\pi kl/N}},$$

where $X[k]$ and $E[k]$ are the DFTs of $x[k]$ and $e[n]$, respectively. Thus, the magnitude spectrum of $x[n]$ is given by

$$|X[k]| = \frac{|E[k]|}{\left|1 - \sum_{l=1}^{p} a[l]e^{-\mathrm{i}2\pi kl/N}\right|}, \qquad k = 0, \ldots, N-1. \tag{13}$$

We try to model this spectrum with an *all-pole spectrum* of the form

$$|Y[k]| = \frac{g}{\left|1 - \sum_{l=1}^{p} a[l] e^{-\mathrm{i}2\pi kl/N}\right|}, \qquad k = 0, \ldots, N-1, \tag{14}$$

where g is a real-valued constant, known as the *gain*, and where the coefficients $a[l]$ are the bounded to the ones in (13). Additionally, we require that that the average ratio between the power spectra $|X[k]|^2$ and $|Y[k]|^2$ is one. That is,

$$\frac{1}{N} \sum_{k=0}^{N-1} \frac{|X[k]|^2}{|Y[k]|^2} = 1. \tag{15}$$

In [20], Equation (15) is derived from (28) presented below. In our spectral oriented approach it is natural to instead take (15) as a given and then derive (28).

One possible way to obtain sensible spectrum estimates is to minimize the sum

$$\sum_{k=0}^{N-1} \frac{|X[k]|^2}{|Y[k]|^2} \tag{16}$$

while keeping g fixed. By minimizing (16), we get spectral estimates that follow the shape of $|X[k]|$. Then, by selecting a proper value for g, we can fulfill the requirement (15).

A substitution of (13) and (14) into (16) gives

$$\sum_{k=0}^{N-1} \frac{|X[k]|^2}{|Y[k]|^2} = \sum_{k=0}^{N-1} \frac{|E[k]|^2}{g^2} = \frac{1}{g^2} \sum_{k=0}^{N-1} |E[k]|^2 = \frac{N}{g^2} \sum_{n=0}^{N-1} |e[n]|^2, \tag{17}$$

where the last equality follows from Theorem 2.3. The interesting thing about this continued equality is that it implies that the same coefficients $a[l]$ that minimize the error signal $e[n]$ in the least squares sense, are the same that minimize (16). Next, we focus on this minimization of the error signal, and after that the constant g is determined.

3.2 Error minimization

For any complex number c, we have $|c|^2 = c\overline{c}$, where $\overline{c}$ is the complex conjugate of c. Therefore, we can write

$$\sum_{n=0}^{N-1}|e[n]|^2 = \sum_{n=0}^{N-1} e[n]\overline{e[n]}. \tag{18}$$

By substituting (11) and its complex conjugate into (18) and by expanding the resulting product, we get

$$\begin{aligned}\sum_{n=0}^{N-1}|e[n]|^2 &= \sum_{n=0}^{N-1}\left(x[n]-\sum_{l=1}^{p}a[l]x[n-l]\right)\left(\overline{x[n]}-\sum_{l=1}^{p}\overline{a[l]x[n-l]}\right)\\ &= \sum_{n=0}^{N-1}|x[n]|^2 - \sum_{n=0}^{N-1}x[n]\sum_{l=1}^{p}\overline{a[l]x[n-l]} - \sum_{n=0}^{N-1}\overline{x[n]}\sum_{l=1}^{p}a[l]x[n-l] \\ &\quad + \sum_{n=0}^{N-1}\sum_{l=1}^{p}a[l]x[n-l]\sum_{k=1}^{p}\overline{a[k]x[n-k]}\end{aligned} \tag{19}$$

The sum $\sum_{n=0}^{N-1}|e[n]|^2$ is a real valued function of several complex variables $a[i]$. In [24], it is demonstrated that the stationary points (e.g. minimum, maximum and saddle points) of such a function are located in the points, where for all $j = 1, \ldots, p$

$$\frac{\partial}{\partial\overline{a[j]}}\sum_{n=0}^{N-1}|e[n]|^2 = 0$$

and where the above partial derivative notation means the derivatives with respect to the variables $\overline{a[j]}$ while considering the variables $a[j]$ to be independent of $\overline{a[j]}$. By calculating these derivatives from the expression (19), we get

$$\begin{aligned}\frac{\partial}{\partial\overline{a[j]}}\sum_{n=0}^{N-1}|e[n]|^2 &= -\sum_{n=0}^{N-1}x[n]\overline{x[n-j]} + \sum_{n=0}^{N-1}\sum_{l=1}^{p}a[l]x[n-l]\overline{x[n-j]} \qquad (20)\\ &= -\sum_{n=0}^{N-1}x[n]\overline{x[n-j]} + \sum_{l=1}^{p}a[l]\sum_{n=0}^{N-1}x[n-l]\overline{x[n-j]} \\ &= -C[j,0] + \sum_{l=1}^{p}a[l]C[j,l], \qquad (21)\end{aligned}$$

where the elements $C[j,l]$ are defined as follows:

$$C[j,l] = \sum_{n=0}^{N-1} x[n-l]\overline{x[n-j]}. \tag{22}$$

By setting (21) zero for all $j = 1, \ldots, p$, we arrive at the following p equations:

$$\begin{cases} a[1]C[1,1] + a[2]C[1,2] + a[3]C[1,3] + \ldots + a[p]C[1,p] = C[1,0] \\ a[1]C[2,1] + a[2]C[2,2] + a[3]C[2,3] + \ldots + a[p]C[2,p] = C[2,0] \\ a[1]C[3,1] + a[2]C[3,2] + a[3]C[3,3] + \ldots + a[p]C[3,p] = C[3,0] \\ \quad\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad \vdots \qquad\qquad \vdots \\ a[1]C[p,1] + a[2]C[p,2] + a[3]C[p,3] + \ldots + a[p]C[p,p] = C[p,0] \end{cases}$$

This system of equations can be written in a matrix form:

$$\begin{bmatrix} C[1,1] & C[1,2] & C[1,3] & \ldots & C[1,p] \\ C[2,1] & C[2,2] & C[2,3] & \ldots & C[2,p] \\ C[3,1] & C[3,2] & C[3,3] & \ldots & C[3,p] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C[p,1] & C[p,2] & C[p,3] & \ldots & C[p,p] \end{bmatrix} \begin{bmatrix} a[1] \\ a[2] \\ a[3] \\ \vdots \\ a[p] \end{bmatrix} = \begin{bmatrix} C[1,0] \\ C[2,0] \\ C[3,0] \\ \vdots \\ C[p,0] \end{bmatrix} \tag{23}$$

Let us denote the coefficient matrix by $\boldsymbol{C}$, the predictor coefficient vector by $\boldsymbol{a}$, and the vector in the right-hand side of (23) by $\boldsymbol{c}$. Then, the linear prediction coefficients are obtained as $\boldsymbol{a} = \boldsymbol{C}^{-1}\boldsymbol{c}$ (in Matlab, `a = C\c`).

Now, suppose that we have observed a signal $x[n]$ that is defined in the range $n = -p, \ldots, N-1$ and that we keep the minimization range the same as before (from 0 to $N-1$). Then (23) is still valid. The only thing that changes is the calculation of the elements $C[j,l]$ because in (22) we do not have to assume zeros for $x[n]$ when $-p \leq n < 0$ anymore.

The method that uses the front buffer of p values is known as the *covariance method* [20, 21]. The use of front buffer is not obligatory, but it tends to improve the prediction quality especially in the beginning of the signal, since the predictor (9) has then past values available.

3.3 Autocorrelation method

The autocorrelation method of solving the predictor coefficients is formulated with the help of *autocorrelation sequence*

$$R[l] = \sum_{n=0}^{N-1} x[n]\overline{x[n-l]}, \qquad l = 0, \ldots, N-1.$$

The benefit of the autocorrelation method is that it offers some computational advantage over the covariance method as will be noted in the end of this section. Additionally, the solutions given by the autocorrelation method are guaranteed to be stable (i.e. all poles are inside the unit circle, see Section 3.6). The solutions given by the covariance method are not necessarily stable [20].

To perform an exact minimization of (18) with the autocorrelation method, we have to assume that the last p values of the observed signal $x[n]$ are zero. This assumption is only a minor issue, since we can always add p zeros to the end of the signal (see zero-padding in Section 4.2) or alternatively we can use a window function to bring the last p values close to zero. When the last p values are zero, we can write the elements $C[j,l]$ in terms of the autocorrelation values. The first step in making this conversion is to change the summation limits of $C[j,l]$ as follows:

$$C[j,l] = \sum_{n=0}^{N-1} x[n-l]\overline{x[n-j]} = \sum_{n=-l}^{N-1-l} x[n]\overline{x[n-(j-l)]}.$$

Since $x[n] = 0$ when $n < 0$ and when $n > N-1-l \geq N-1-p$, we can change the summation limits to obtain the correspondence with the autocorrelation values in the case where $j \geq l$:

$$C[j,l] = \sum_{n=0}^{N-1} x[n]\overline{x[n-(j-l)]} = R[j-l]. \tag{24}$$

Similarly, when $j < l$, we get

$$\begin{aligned} C[j,l] = \sum_{n=0}^{N-1} x[n-l]\overline{x[n-j]} &= \sum_{n=-j}^{N-1-j} x[n-(l-j)]\overline{x[n]} \\ &= \sum_{n=0}^{N-1} x[n-(l-j)]\overline{x[n]} \\ &= \overline{R[l-j]}. \end{aligned} \tag{25}$$

And lastly,

$$C[j,0] = \sum_{n=0}^{N-1} x[n]\overline{x[n-j]} = R[j]. \tag{26}$$

By substituting (24), (25) and (26) into (23), we get

$$\begin{bmatrix} R[0] & \overline{R[1]} & \overline{R[2]} & \dots & \overline{R[p-1]} \\ R[1] & R[0] & \overline{R[1]} & \dots & \overline{R[p-2]} \\ R[2] & R[1] & R[0] & \dots & \overline{R[p-3]} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R[p-1] & R[p-2] & R[p-3] & \dots & R[0] \end{bmatrix} \begin{bmatrix} a[1] \\ a[2] \\ a[3] \\ \vdots \\ a[p] \end{bmatrix} = \begin{bmatrix} R[1] \\ R[2] \\ R[3] \\ \vdots \\ R[p] \end{bmatrix}. \tag{27}$$

The coefficient matrix in (27) is a *Toeplitz matrix*. That is, a matrix in which all the left-to-right descending diagonals have a constant value. Numerous algorithms have been developed to solve systems containing a Toeplitz matrix. While the basic *Gaussian elimination* has $O(p^3)$ time complexity, the algorithms for Toeplitz systems can have $O(p^2)$ or even $O(p(\log p)^2)$ time complexities [4].

3.4 Gain computation

Now that we have described how to solve the predictor coefficients $a[l]$, the remaining issue is to determine the gain constant g in (14). From (15) and (17), we obtain

$$g^2 = \sum_{n=0}^{N-1} |e[n]|^2. \tag{28}$$

Then, the substitution of (11) into this gives

$$g^2 = \sum_{n=0}^{N-1} \left| x[n] - \sum_{l=1}^{p} a[l]x[n-l] \right|^2, \tag{29}$$

from where we are able to calculate the value of g. But, there is a simpler way to obtain the same value by taking advantage of the autocorrelation values. We start the derivation by rewriting (29) as follows:

$$\begin{aligned} g^2 &= \sum_{n=0}^{N-1} \left(x[n] - \sum_{l=1}^{p} a[l]x[n-l] \right) \left(\overline{x[n]} - \sum_{l=1}^{p} \overline{a[l]x[n-l]} \right) \\ &= \sum_{n=0}^{N-1} x[n] \left(\overline{x[n]} - \sum_{l=1}^{p} \overline{a[l]x[n-l]} \right) \qquad (30) \\ &\quad - \sum_{n=0}^{N-1} \left(\sum_{l=1}^{p} a[l]x[n-l] \right) \left(\overline{x[n]} - \sum_{l=1}^{p} \overline{a[l]x[n-l]} \right). \qquad (31) \end{aligned}$$

Let us handle the terms (30) and (31) separately starting with the latter one:

$$\begin{aligned} &- \sum_{n=0}^{N-1} \left(\sum_{l=1}^{p} a[l]x[n-l] \right) \left(\overline{x[n]} - \sum_{l=1}^{p} \overline{a[l]x[n-l]} \right) \\ &= - \sum_{n=0}^{N-1} \left(\sum_{j=1}^{p} a[j]\overline{x[n]}x[n-j] - \sum_{j=1}^{p}\sum_{l=1}^{p} \overline{a[l]}a[j]\overline{x[n-l]}x[n-j] \right) \\ &= \sum_{j=1}^{p} a[j] \left(- \sum_{n=0}^{N-1} \overline{x[n]}x[n-j] + \sum_{n=0}^{N-1}\sum_{l=1}^{p} \overline{a[l]x[n-l]}x[n-j] \right) \qquad (32) \\ &= \sum_{j=1}^{p} a[j] \overline{\frac{\partial}{\partial \overline{a[j]}} \sum_{n=0}^{N-1} |e[n]|^2}. \end{aligned}$$

The last equality is obtained by comparing (32) with (20). The predictor coefficients $a[l]$ were obtained by setting

$$\frac{\partial}{\partial \overline{a[j]}} \sum_{n=0}^{N-1} |e[n]|^2 = 0$$

for all $j = 1, \ldots, p$, and hence (31) vanishes. Thus, we are only left with the term (30)

and so

$$\begin{aligned} g^2 &= \sum_{n=0}^{N-1} x[n] \left(\overline{x[n]} - \sum_{l=1}^{p} \overline{a[l]x[n-l]} \right) \\ &= \sum_{n=0}^{N-1} x[n]\overline{x[n]} - \sum_{l=1}^{p} \overline{a[l]} \sum_{n=0}^{N-1} x[n]\overline{x[n-l]} \\ &= R[0] - \sum_{l=1}^{p} \overline{a[l]} R[l], \end{aligned}$$

which is the desired formula for g.

3.5 Summary: computing spectral estimates

A short step-by-step summary of the spectrum modeling using the autocorrelation method and the non-buffered covariance method is given below:

1. Choose the linear prediction model order p.
2. Apply a window function to the signal $x[n]$.
3. Calculate the autocorrelation values $R[0], \ldots, R[p]$. Additionally, in the case of the covariance mathod, calculate $C[j, l]$, defined in (22), for all the combinations of $j, l = 1, \ldots, p$.
4. Solve the predictor coefficients $a[l], l = 1, \ldots, p$, from the system (27) (autocorrelation method) or from the system (23) (covariance method).
5. Calculate the gain $g = \sqrt{R[0] - \sum_{l=1}^{p} \overline{a[l]} R[l]}$.
6. Finally, the magnitude spectrum estimate is given by

$$|Y[k]| = \frac{g}{\left| 1 - \sum_{l=1}^{p} a[l] e^{-\mathrm{i}\frac{2\pi kl}{N}} \right|}, \quad k = 0, \ldots, N-1.$$

The described method is applicable for both real and complex valued signals $x[n]$.

3.6 Pole properties

Markel [21, pp. 164 – 172] describes a way to find the peaks from the all pole spectrum (14). This method requires analysis of the Z-domain form of (14), which is

$$Y(z) = \frac{g}{1 - \sum_{l=1}^{p} a[l] z^{-l}}. \tag{33}$$

The magnitude spectrum (14) can be obtained from (33) by evaluating $Y(z)$ at the points of the unit circle $z = e^{\mathrm{i}2\pi k/N}$, $k = 0, \ldots, N-1$, and by taking the absolute value.

Let us multiply both the numerator, and the denominator of (33) by z^p to obtain a rational function representation

$$Y(z) = \frac{g z^p}{z^p - a[1] z^{p-1} - a[2] z^{p-2} - \ldots - a[p-1] z - a[p]}. \tag{34}$$

The roots of the denominator of a complex rational function are called *poles*. The denominator in (34) is a polynomial of degree p, and so it has p roots and, consequently, $Y(z)$ has p poles. The distance of the pole to the unit circle determines how strongly the magnitude spectrum peaks at the frequency that corresponds to the angle of the pole. Pole frequencies and sharpness values can be defined more exactly as follows: Let r be a pole of $Y(z)$. Then the frequency that corresponds to this pole is given by

$$f = \frac{f_s}{2\pi} \arg(r),$$

where the argument of r is an angle between 0 and 2π and where f_s is the sampling frequency [21, p. 167]. The sharpness of the pole is defined in [2] as follows:

$$s = \frac{1}{1 - |r|}.$$

The value of sharpness goes from one to infinity when the magnitude of the pole goes from zero to one. These sharpness values (together with subband-FDLP, see Chapter 4) are used in [2] to extract temporal features from a speech signal.

Fig. 5 shows how the pole locations are related to the resulting spectrum. Let us consider, for example, the estimate with the model order $p = 15$. From the spectral esti-

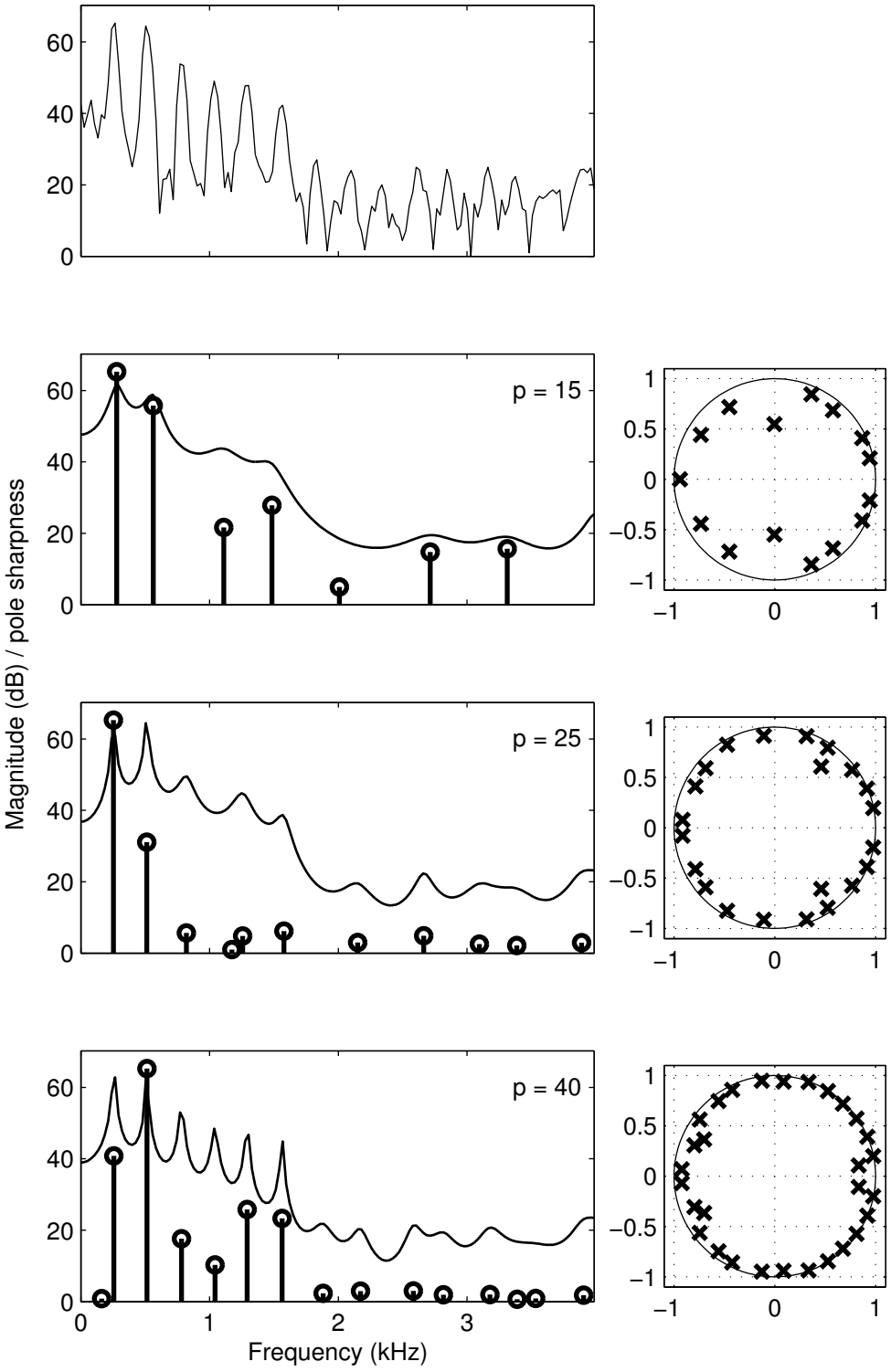

Figure 5: All-pole spectrum estimates of the topmost magnitude spectrum using three different model orders ($p = 15$, $p = 25$, $p = 40$). Peak locations are shown by the vertical lines. Sharpness is indicated by the line height. Sharpness values are normalized to fit the values in the graphs. Additionally, on the right, the pole locations for each estimate are shown in the complex plane.

mate and from the corresponding pole location graph we find that moving left to right in the spectrum corresponds to moving along the unit circle in a counter clockwise direction starting from the point $(1, 0)$. The four first poles along the unit circle are relatively close to the unit circle and therefore the spectrum peaks at the corresponding locations. The fifth pole is much closer to the origin and consequently the fifth peak location in the spectrum is not actually a peak at all.

From the pole location graphs we can observe that poles are located symmetrically with respect to real axis. This follows from the symmetry of the magnitude spectra of the real valued signals (Fig. 5 shows only the first halves of the spectra). To find the peaks from the non-redundant part of the magnitude spectrum, we therefore have to use a model order that is at least twice as large as the desired number of peaks. From Fig. 5, we see that only the estimate obtained by using a model order $p = 40$ shows the six most visible peaks in the spectrum.

3.7 Line spectral frequencies

When the LP-coefficients $a[l]$ are real valued, they can be alternatively represented using the *line spectral frequencies* (LSFs), also known as the *line spectrum pairs*, originally introduced by Itakura in 1975 [13]. The advantage of LSFs is that they are more tolerant to quantization errors than the LP-coefficients [15, 28, 30]. The conversion between the LP-coefficients and the LSFs is lossless, at least in theory. In this section, we explain shortly the mathematical basis of the LSFs, how they are formed from the LP-coefficients, and how the LP-coefficients can be obtained back from the LSFs. Additionally, we show how the LSFs relate to the all pole spectral representation of the signal.

Let us denote the denominator of (33) by $A(z)$. That is

$$A(z) = 1 - \sum_{l=1}^{p} a[l] z^{-l}. \tag{35}$$

Then $A(z)$ can be expressed with the help of two polynomials,

$$P(z) = A(z) + z^{-(p+1)} A(z^{-1}) \quad \text{and} \tag{36}$$

$$Q(z) = A(z) - z^{-(p+1)} A(z^{-1}), \tag{37}$$

as follows:

$$A(z) = \frac{P(z) + Q(z)}{2}. \tag{38}$$

By substituting (35) into the polynomials (36) and (37) we obtain

$$P(z) = 1 + \sum_{l=1}^{p} (a[l] + a[p+1-l]) z^{-l} + z^{-(p+1)}, \tag{39}$$

$$Q(z) = 1 + \sum_{l=1}^{p} (a[l] - a[p+1-l]) z^{-l} - z^{-(p+1)}. \tag{40}$$

From these forms we see that both $P(z)$ and $Q(z)$ have $p+1$ zeros. Soong and Juang [28] have proved two important properties of these zeros; firstly, they lie on the unit circle and secondly, the zeros of $P(z)$ and $Q(z)$ are interleaved with each other. Furthermore, since the polynomial coefficients are real valued, the complex zeros of $P(z)$ and $Q(z)$ occur in complex conjugate pairs. By evaluating (39) and (40) at $z = 1$ and $z = -1$, we find that if p is even, then $P(z)$ has a zero at $z = -1$ and $Q(z)$ has a zero at $z = 1$. If p is odd, then $Q(z)$ has zeros at both $z = -1$ and $z = 1$ and therefore $P(z)$ does not have real valued zeros. Figure 6 demonstrates the explained positioning of the zeros.

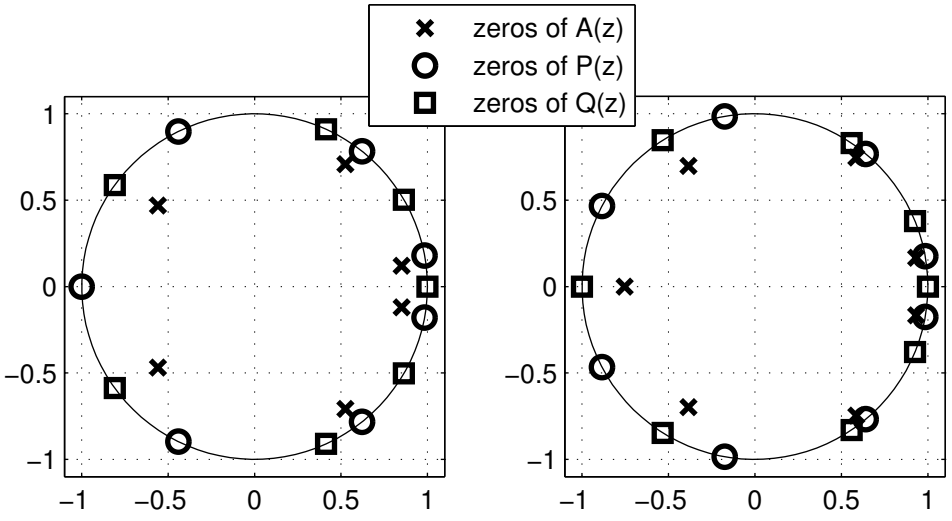

Figure 6: Zeros of $A(z)$, $P(z)$, and $Q(z)$ for even model order $p = 6$ (on the left) and for odd model order $p = 7$ (on the right). In the odd case, both real valued zeros belong to $Q(z)$, while for the even p, the zero at -1 belongs to $P(z)$. All complex zeros occur in complex conjugate pairs (i.e. symmetrically over the real axis). This figure was reproduced from [15].

The above described properties ensure that, in total, there are always p complex zeros

in the upper part of the unit circle. Because we can deduce the real valued zeros from the model order p and because the complex zeros occur in complex conjugate pairs, we can retain all the zeros by just knowing the arguments (phases) of the p complex zeros that lie in the upper unit circle. These arguments are known as the LSFs. The interleaving property makes it possible to know which of the zeros belong to $P(z)$ and which to $Q(z)$.

In summary the calculation of LSFs from the LP-coefficients involves three steps: (i) calculation of the polynomial coefficients in (39) and (40) from the LP-coefficients, (ii) solving the zeros of (39) and (40) (the most difficult part), and (iii) taking the arguments of the complex zeros in the upper unit circle. The reverse operation is much easier, since there is no need to solve the zeros. It is enough to turn the LSFs into the zeros of $P(z)$ and $Q(z)$ and then we can construct the polynomials $P(z)$ and $Q(z)$, which are needed to calculate (38), from where the LP-coefficients are readily obtained.

Fig. 7 shows how the LSFs get located in relation to the corresponding all-pole spectrum. We find that LSFs are more densely located near the spectral peaks. This phenomenon can be explained by substituting (38) into (33). This gives us

$$Y(z) = \frac{2g}{P(z) + Q(z)}.$$

If both $P(z)$ and $Q(z)$ have a zero very close to a point $z = a$, then $|P(a) + Q(a)|$ is small and consequently $|Y(a)|$ is large. Therefore the closeness of the LSFs is related to spectral peaking.

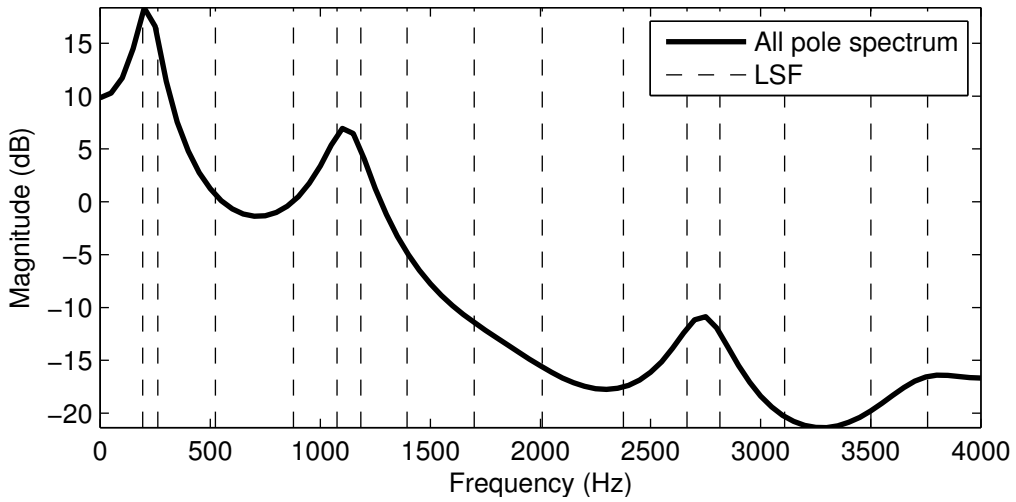

Figure 7: All pole spectrum estimate obtained using a model order $p = 15$ together with the corresponding LSFs. The LSFs are more densely located near the spectral peaks. This figure was reproduced from [30].

4 Linear prediction and modeling of time domain signals and their envelopes

In the previous chapter, we described how the linear prediction (LP) model can be used to obtain an all-pole estimate of the magnitude spectrum of a signal. We achieved this by applying the LP model to a time domain signal. An other, less well-known way is to reverse the roles and apply LP to a frequency domain signal to obtain a positive valued time domain signal [3]. We use the terms *time domain linear prediction* (TDLP) and *frequency domain linear prediction* (FDLP) to discriminate whether the LP model is applied to a time or to a frequency domain signal.

Given a time domain signal, its envelope can be modeled by transforming the signal to the frequency domain and applying FDLP. The key is to use the *discrete cosine transform* (DCT) or the IDFT instead of the DFT to obtain a frequency-domain representation. We use the abbreviations DCT-FDLP and IDFT-FDLP depending whether the FDLP is applied to an DCT-signal or to a IDFT-signal.

We begin this chapter by introducing the concepts needed to understand the theoretical basis of the DCT-FDLP method. First, we introduce the Hilbert transform, analytic signal, and Hilbert envelope. Then, we define symmetrized and zero-padded signals and DCT and reveal following interesting DCT-related matters: the connection between the Hilbert envelope and the DCT, subband decompositions using the DCT, and the ability of the DCT to mirror the spectrogram. When the above is done, we are ready to introduce the DCT-FDLP method, after which an alternative FDLP technique, IDFT-FDLP, is proposed. We conclude this chapter by discussing the modeling of positive valued signals (instead of signal envelopes) using FDLP.

4.1 Hilbert transforms, analytic signals and Hilbert envelopes

In the following sections, we will show that DCT-FDLP produces estimates of the Hilbert envelope of the modeled signal. Before introducing the Hilbert envelope, we need to define the Hilbert transform and the analytic signal. The Hilbert transform operates on a signal to produce another signal in the same domain. The transformed signal has (almost) the original magnitude spectrum, but the phase spectrum is modified. The phase modification is done by shifting the phase of the positive frequency

components by $-\pi/2$ and the phase of the negative frequency components by $\pi/2$. The formal definition of this is given as follows:

Definition 4.1 (Hilbert transform). Let $x[n], n = 0, \ldots, N-1$, be a sequence and let $X[k]$ be the DFT of $x[n]$. Then, the *Hilbert transform* of $x[n]$ is defined as

$$\mathcal{H}\{x[n]\} = \mathrm{IDFT}\{X_h[k]\},$$

where

$$X_h[k] = \begin{cases} 0, & k = 0, \\ -\mathrm{i}X[k], & 1 \leq k < N/2, \\ 0, & k = N/2, \\ \mathrm{i}X[k], & N/2 < k \leq N-1. \end{cases}$$

The multiplications with $-\mathrm{i}$ and i correspond to phase shifts of $-\pi/2$ and $\pi/2$, respectively. In the frequency domain representation of the Hilbert transform, the first and the $(N/2)$th values are set to zero. The first DFT-coefficient equals the sum of the values of $x[n]$ and the $(N/2)$th DFT-coefficient is the sum of even-indexed values of $x[n]$ subtracted by the odd-indexed values of $x[n]$. This can be seen from (1) by setting $k = 0$ and $k = N/2$. When N is odd, the $(N/2)$th DFT-coefficient does not exist, and we can simply ignore the third line of the piecewise given definition of $X_h[k]$.

When $x[n]$ is real valued, its DFT $X[k]$ fulfills the symmetry properties (6) and (7). The way $X_h[k]$ is formed from $X[k]$ guarantees that $X_h[k]$ has the same symmetry properties as $X[k]$ has. Therefore, the Hilbert transform of a real valued signal is also a real valued signal.

The analytic signal is constructed from a real valued signal $x[n]$ by setting the real part of the analytic signal to be $x[n]$ itself and by setting the complex part to be the Hilbert transform of $x[n]$:

Definition 4.2 (Analytic signal). Let $x[n]$ be a real valued sequence. Then the *analytic representation* of $x[n]$ is defined as

$$\mathrm{An}\{x[n]\} = x[n] + \mathrm{i}\mathcal{H}\{x[n]\}. \tag{41}$$

An alternative formulation for the analytic signal is revealed by the following theorem:

Theorem 4.3. *Let $x[n], n = 0, \ldots, N-1$, be a real-valued sequence and let $X[k]$ be the DFT of $x[n]$. Then the* analytic representation *of $x[n]$ is given by*

$$\mathrm{An}\{x[n]\} = \mathrm{IDFT}\{X_a[k]\}, \tag{42}$$

where

$$X_a[k] = \begin{cases} X[0], & k = 0, \\ 2X[k], & 1 \leq k < N/2, \\ X[N/2], & k = N/2, \\ 0, & N/2 < k \leq N-1. \end{cases}$$

Proof. By substituting (41) into (42), we get

$$\mathrm{IDFT}\{X_a[k]\} = x[n] + \mathrm{i}\mathcal{H}\{x[n]\}$$

and by applying the DFT to both sides of this equation, we may write

$$X_a[k] = X[k] + \mathrm{i}X_h[k].$$

Piecewise evaluation of this equation proves the theorem:

$$\begin{aligned} &X_a[0] = X[0] + \mathrm{i}X_h[0] = X[0], \\ &X_a[k] = X[k] + \mathrm{i}X_h[k] = X[k] + \mathrm{i}(-\mathrm{i}X[k]) = 2X[k], \quad 1 \leq k < N/2, \\ &X_a[N/2] = X[N/2] + \mathrm{i}X_h[N/2] = X[N/2], \\ &X_a[k] = X[k] + \mathrm{i}X_h[k] = X[k] + \mathrm{i}(\mathrm{i}X[k]) = 0, \qquad N/2 < k \leq N-1. \ \square \end{aligned}$$

The *Hilbert envelope* of a signal $x[n]$ is defined as the modulus of the analytic signal $|\mathrm{An}\{x[n]\}|$. Fig. 8 shows examples of the Hilbert envelopes of two different kinds of signals. The Hilbert envelope of an amplitude modulated sinusoid (on the top) forms a smooth curve above the signal. This modulated sinusoid is created by multiplying a high frequency sinusoid with a sinusoid that has ten times lower frequency. On the other hand, a speech signal depicted in the lower panel of Fig. 8 consists of various frequency components and has a Hilbert envelope that looks much more irregular. This irregularity will not be a major concern for us since our objective is to create smoothed estimates of Hilbert envelopes using the DCT-FDLP method, which will described in

Section 4.5.

From (41) we deduce that the Hilbert envelope is obtained by calculating

$$|\mathrm{An}\{x[n]\}| = \sqrt{x[n]^2 + h[n]^2}, \tag{43}$$

where $h[n] = \mathcal{H}\{x[n]\}$. This implies that the Hilbert envelope lies above the modulus of the signal and also above the modulus of the signal's Hilbert transform. As seen from Fig. 8, the Hilbert transform tends to peak when the signal itself is close to zero. This property of the Hilbert transform together with (43) explains why the Hilbert envelope to tends run relatively smoothly over the signal while avoiding fast oscillations.

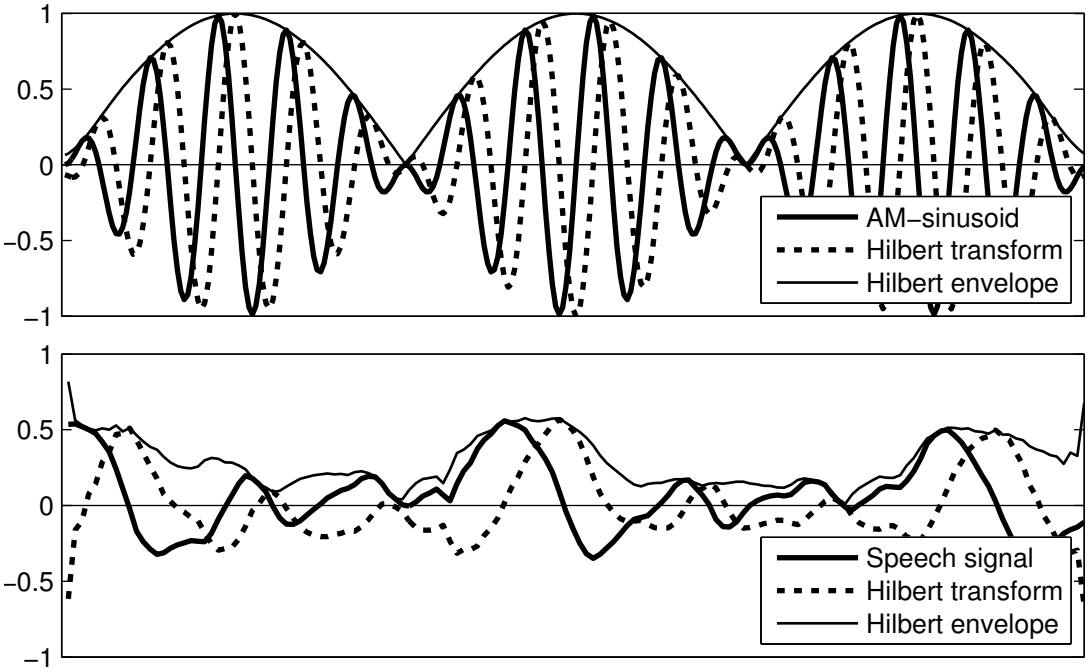

Figure 8: Amplitude modulated sinusoid (on the top) and a short clip of a speech signal (on the bottom) together with their Hilbert transforms and envelopes. In both cases, the Hilbert envelope lies above the modulus of the signal and also above the modulus of the Hilbert transform as indicated by (43).

4.2 Symmetrized and zero-padded signals

We are going to use a result from [3] that connects the DCT with the analytic signal. This result includes symmetrized and zero-padded signals. The definitions for these signals are given below.

Definition 4.4 (Symmetrized signal). Let $x[n], n = 0, \ldots, N-1$, be a sequence. Then the *symmetrized* version of this signal is defined as

$$\mathrm{Sym}\{x[n]\} = \begin{cases} x[n], & n = 0, \ldots, N-1, \\ x[2N-1-n], & n = N, \ldots, 2N-2. \end{cases}$$

In this way of symmetrization, the first value of the original signal is the only one that does not occur twice in the symmetrized signal. Therefore symmetrized N-length signals have a length of $2N-1$. An illustration of a symmetrized speech signal is given in Fig. 9.

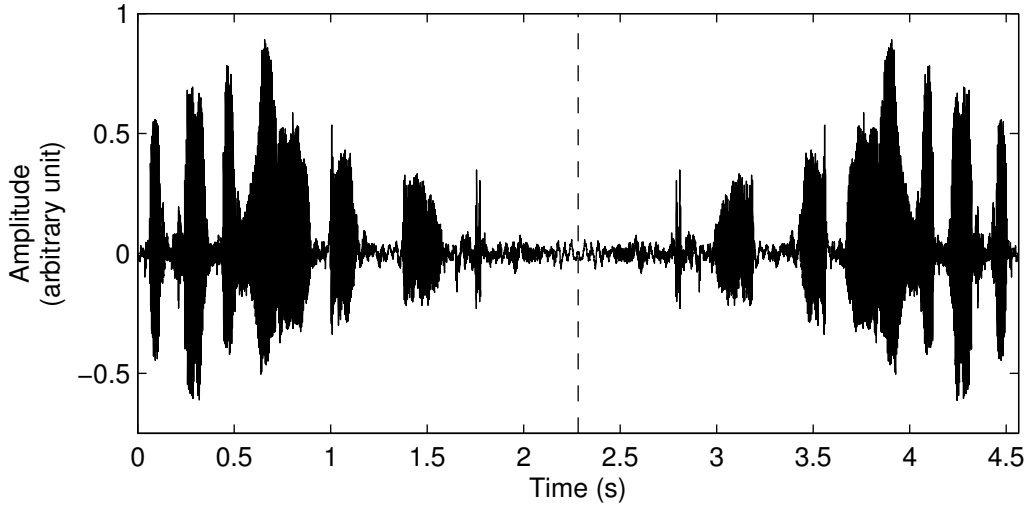

Figure 9: The symmetrized signal is created by appending the flipped version of the original signal to the end of the signal. After this, the last value is removed to make the length of the output $2N-1$.

Definition 4.5 (Zero-padded signal). Let $x[n], n = 0, \ldots, N-1$, be a sequence. Then the *zero-padded* version of this signal is defined as

$$\mathrm{ZP}\{x[n]\} = \begin{cases} x[n], & n = 0, \ldots, N-1, \\ 0, & n = N, \ldots, 2N-2. \end{cases}$$

The above definition defines a zero-padding operator ZP that appends $N-1$ zeros to the end of the input signal. Obviously, any other amount of zeros could be appended as well, but this is the amount that we will need when presenting the connection between the DCT and the analytic signal.

Zero-padding can be used to create an interpolated version of the DFT of the signal. Fig. 10 shows the magnitude spectra of the original signal and the signal that is zero-padded so that it is four times longer than the original. The magnitude spectrum of the zero-padded signal (lower curve) has more data points and hence it is smoother.

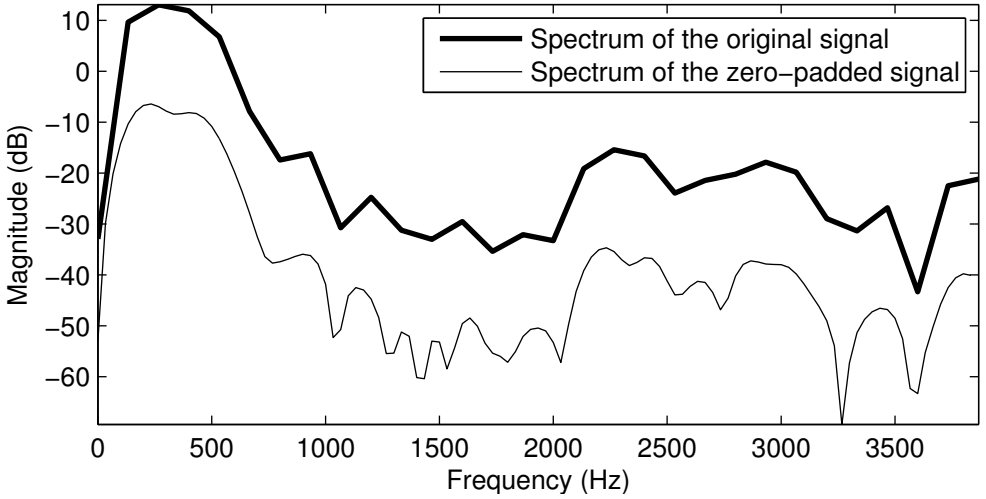

Figure 10: The upper curve is a magnitude spectrum of a 7.5 ms long speech segment. The lower curve is the magnitude spectrum of a 30 ms long zero-padded version of this segment (shifted by -20dB to improve visualization).

4.3 Discrete cosine transform

In this section, we define the DCT and review an interesting property, elaborated in [3], which connects the DCT with analytic signals.

There are many variations of the DCT [5, pp. 29 – 35]. Multiple types of DCTs have been formulated because there are different ways to define how a finite length signal is *continued periodically* (discrete trigonometric transforms treat discrete signals as infinite periodic ones). At both endpoints it is possible to make the continuation even or odd, and the continuation can be done over a "full sample" or over a "half sample". To explain the difference, consider a sequence $\{1, 2, 3, 4\}$. A signal continued over the *full sample* is $\{1, 2, 3, 4, 3, 2, 1\}$, while the *half sample* continuation results in $\{1, 2, 3, 4, 4, 3, 2, 1\}$.

The DCT-type that is used in [3] is known as the *type I odd DCT*. Some other sources, for example [5], refer to this DCT as the *type V DCT*. The latter naming might be more clear, since in this case, the "oddness" does not refer to the odd continuations, but to

the fact that one border is continued over the half sample, while the other is continued over the full sample [5, p. 34]. The definition of this DCT-type is given below:

Definition 4.6 (Discrete cosine transform (type I odd)). Let $x[n], n = 0, \dots, N-1$, be a real-valued sequence. The *discrete cosine transform* of $x[n]$ is defined as the sequence

$$\mathrm{DCT}\{x[n]\} = V[k] = \frac{4}{2N-1} \sum_{n=0}^{N-1} a_k a_n x[n] \cos \frac{2\pi k n}{2N-1}, \qquad k = 0, \dots, N-1,$$

where

$$a_j = \begin{cases} 1/2, & j = 0, \\ 1, & j > 0. \end{cases}$$

It can be shown that the inverse transform is given by

$$\mathrm{IDCT}\{V[k]\} = x[n] = \sum_{k=0}^{N-1} V[k] \cos \frac{2\pi n k}{2N-1}, \qquad n = 0, \dots, N-1.$$

Now we have defined all the necessary objects to present the result that gives the relation between the DCT and the analytic signal:

Theorem 4.7. *Let $x[n], n = 0, \dots, N-1$, be a real-valued sequence. Then*

$$(2N-1)\,\mathrm{IDFT}\{\mathrm{ZP}\{\mathrm{DCT}\{x[n]\}\}\} = \mathrm{An}\{\mathrm{Sym}\{x[n]\}\}. \tag{44}$$

Proof. See the laborious derivation of (18) in [3]. Coefficient $2N-1$ results from the differing coefficients in the definitions of the DFT and DCT between this thesis and [3]. □

From the equations (1) and (2) we obtain

$$|\mathrm{DFT}\{x[n]\}| = N\,|\mathrm{IDFT}\{x[n]\}|.$$

Therefore, by taking the modulus of both sides of (44), we get

$$|\mathrm{DFT}\{\mathrm{ZP}\{\mathrm{DCT}\{x[n]\}\}\}| = |\mathrm{An}\{\mathrm{Sym}\{x[n]\}\}|, \tag{45}$$

since the input for the IDFT has a length of $2N-1$. Equation (45) tells us that we can obtain the Hilbert envelope of a symmetrized signal by calculating the expression on the left hand side of the equation. That is, by calculating the magnitude spectrum of a zero-padded DCT of the signal. This relation allows us to create all-pole estimates of the Hilbert envelope by applying the LP-technique presented in Section 3.5 to the zero-padded DCT-signal.

4.4 Subband decomposition using the DCT

Let us next examine how a signal can be decomposed to multiple subband signals using cosine transforms. The subband-constructs presented in this section are formulated by the author of this thesis.

Suppose that $V[k]$ is the DCT of a signal $x[n], n = 0, \ldots, N-1$. By defining a sequence

$$V_{a,b}[k] = \begin{cases} V[k], & \text{if } a \leq k < b, \\ 0, & \text{otherwise} \end{cases} \qquad (k = 0, \ldots, N-1)$$

we can write

$$V[k] = V_{0,a_1}[k] + V_{a_1,a_2}[k] + V_{a_2,a_3}[k] + \ldots + V_{a_{m-1},a_m}[k] + V_{a_m,N}[k], \tag{46}$$

where $0 < a_1 < a_2 < \ldots < a_m < N$. Now, by taking the IDCT of equation (46) we get

$$x[n] = \mathrm{IDCT}\{V_{0,a_1}[k]\} + \mathrm{IDCT}\{V_{a_1,a_2}[k]\} + \ldots + \mathrm{IDCT}\{V_{a_m,N}[k]\}, \tag{47}$$

since the IDCT is a linear transform. Here, the signal $x[n]$ is decomposed to multiple signals of a form $\mathrm{IDCT}\{V_{a,b}[k]\}$, each containing only a subset of the frequency components. We refer to these decomposed signals as the *subband signals*.

Fig. 11 displays a speech signal decomposed into three subband signals. As an example, consider the highest frequency subband. This subband can be thought to give information about whether there are high frequencies present at any given time moment. Thus, the envelope of this subband can be thought to tell how much there are

high frequencies at any given time moment.

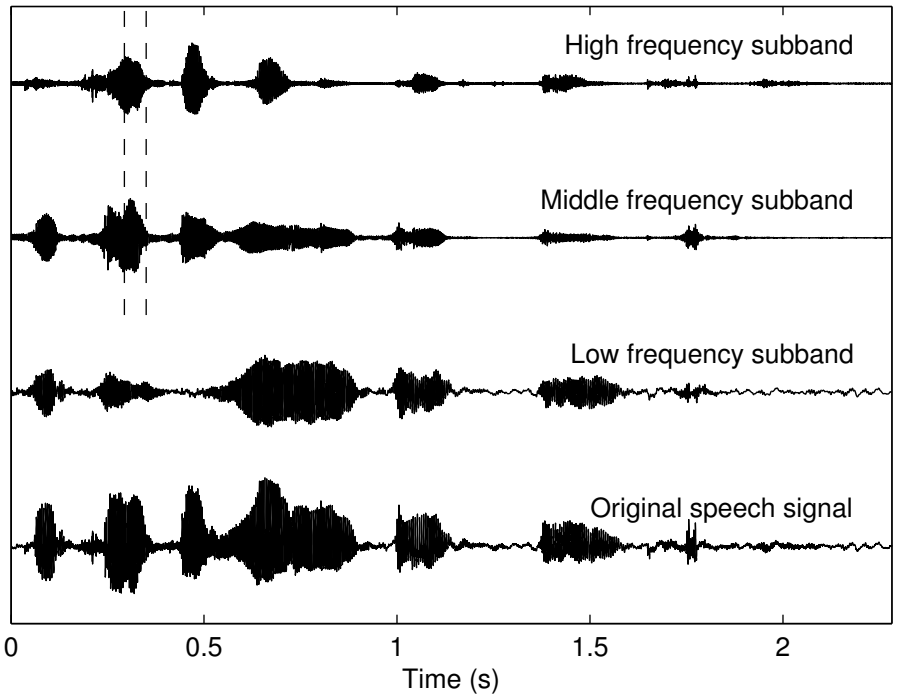

Figure 11: A speech signal (8 kHz) decomposed into three subbands. The low frequency subband ranges from 0 to 444 Hz, the middle frequency subband ranges from 444 to 888 Hz, and the high frequency subband ranges from 888 to 8000 Hz. Adding these subbands together will result in the original signal. A close view of the area marked by the dashed vertical lines is shown in Fig. 13.

One interesting property related to the Hilbert envelopes of the subband signals was found by replacing the subband signals of the form $\mathrm{IDCT}\{V_{a,b}[k]\}$ with $\mathrm{IDCT}\{W_{a,b}[k]\}$, where $W_{a,b}[k]$ is defined as

$$W_{a,b}[k] = \begin{cases} V[k+a], & \text{if } 0 \leq k < b-a, \\ 0, & \text{otherwise.} \end{cases} \qquad (k = 0, \ldots, N-1)$$

While $V_{a,b}[k]$ contains the selected DCT-coefficients at their original places, the sequence $W_{a,b}[k]$ puts the selected DCT-coefficients in the beginning of the sequence. Therefore equations (46) and (47) no longer hold when the sequences V are replaced with the sequences W.

We refer to the signals of the form $\mathrm{IDCT}\{W_{a,b}[k]\}$ as *pseudo*-subband signals. The word "pseudo" is used because even if the pseudo-subband signals do not equal the actual subband signals, they share similar coarse structure. This can be seen by com-

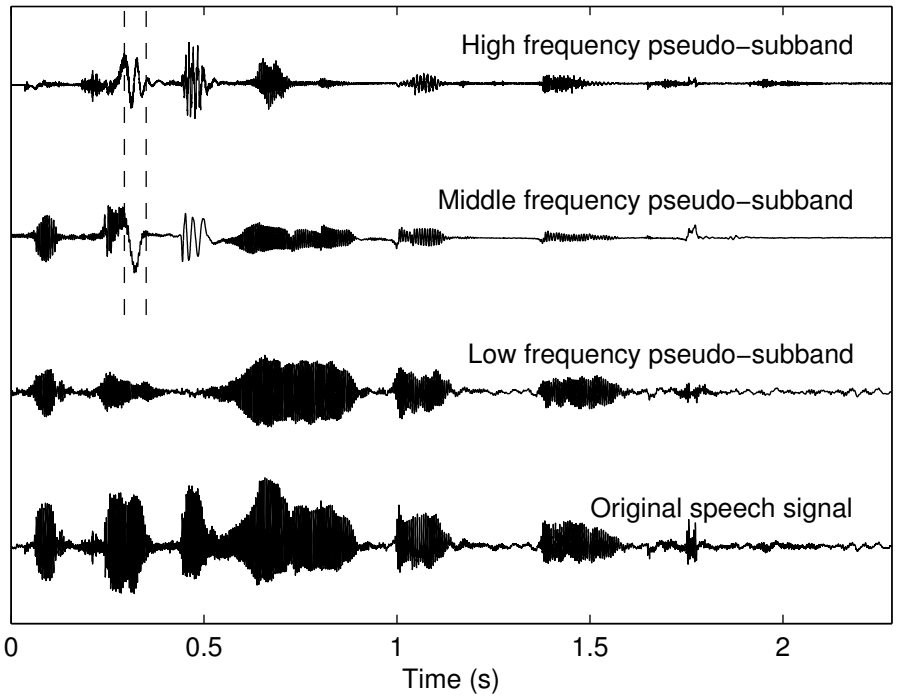

Figure 12: Pseudo-subband signals that correspond to the subband signals in Fig. 11. Although these signals are similar to the subband signals in Fig. 11, adding these signals together will not result in the original signal. A close view of the area marked by the dashed vertical lines is shown in Fig. 13.

paring Figs. 11 and 12. The similarity of the coarse structure of a subband signal and its corresponding pseudo-subband signal leads to the above referred interesting property: Pseudo-subband signals have almost identical Hilbert envelopes with the corresponding subband signals. Fig. 13 shows a close view of the high- and mid-frequency subbands and pseudo-subbands that appear in Figs. 11 and 12. We find that on a close level the pseudo-subband signals differ considerably from the corresponding subband signals. That said, their Hilbert envelopes are still overlapping. The figure does not show the lowest frequency subband and the corresponding pseudo-subband because they are equal ($V_{0,b}[k] = W_{0,b}[k]$).

It was observed in [2] that the spectrogram of the DCT-signal looks almost like the mirrored version of the spectrogram of the original time domain signal (Fig. 14). The similarity between the Hilbert envelopes of the subband and the pseudo-subband signals provides an insight to this mirroring phenomenon: Let us consider a vertical slice in a DCT-spectrogram that has been formed by taking a magnitude spectrum of a frame from the DCT-signal. Let a and b be, respectively, the start and the end indices of this frame in the original DCT-signal. Then this frame is the same as $W_{a,b}[k]$ without the

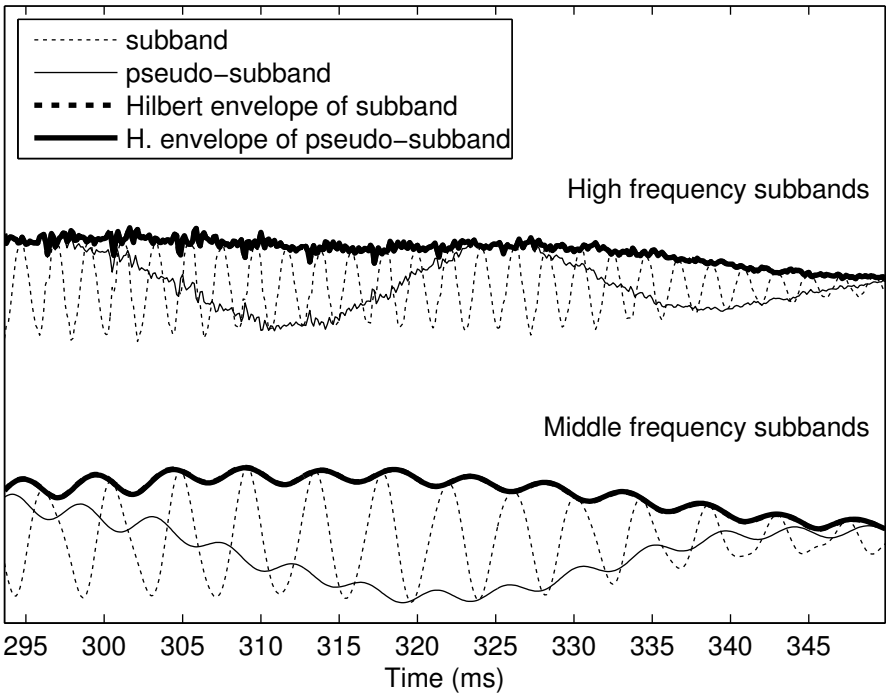

Figure 13: A close view of the area marked by the vertical dashed lines in the figures 11 and 12. Both high and middle frequency subbands are drawn together with their corresponding pseudo-subbands. Additionally, their Hilbert envelopes are drawn. These envelopes overlap and therefore only one Hilbert envelope is visible for each pair of subband and pseudo-subband signals. The low frequency subband and its corresponding pseudo-subband are not drawn since they are equal.

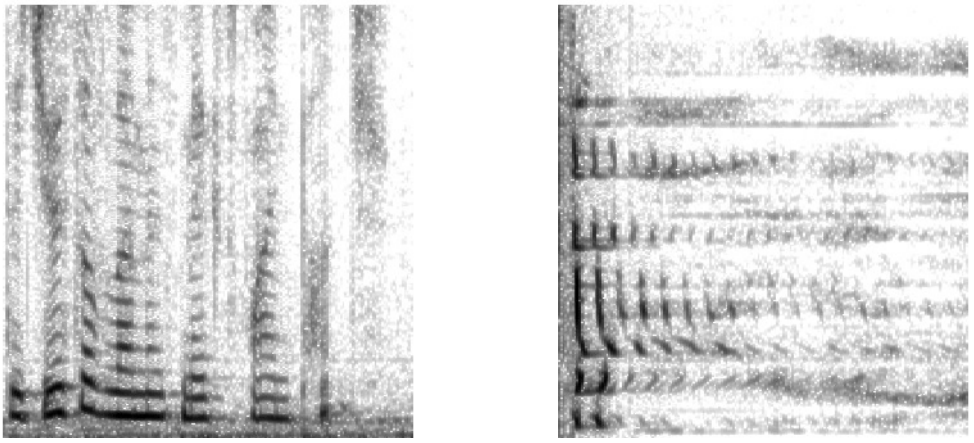

Figure 14: The spectrogram of a speech signal $x[n]$ (on the left) and the spectrogram of the cosine transformed signal $\mathrm{DCT}\{x[n]\}$ (on the right). These spectrograms are almost the mirror images of each other when reflected over the diagonal line from bottom left to top right.

trailing zeros. According to (45), the magnitude spectrum of $W_{a,b}[k]$ provides a decimated version (because of lack of the trailing zeros) of the Hilbert envelope of the pseudo-subband signal specified by the indices a and b. Since the Hilbert envelope of the subband signal can be approximated by the Hilbert envelope of the corresponding pseudo-subband signal, the vertical slice in the DCT-spectrogram represents a decimated and approximated version of the Hilbert envelope of the subband signal. Thus, this slice tells us how much there are subband-frequencies at any given time moment. On the other hand in the conventional spectrogram it is the horizontal slices that give the same information, hence the mirroring effect.

4.5 Envelope modeling with DCT-FDLP

Equation (45) allows us to use the FDLP on the DCT-signal to obtain an estimate of the Hilbert envelope of a symmetrized time-domain signal. Similarly, envelopes of subband signals can be modeled by applying FDLP to the signals $V_{a,b}[k]$ or, alternatively, to the signals $W_{a,b}[k]$.

Fig. 15 shows examples of DCT-FDLP estimates for a speech signal and its subband signals. It should be noted that DCT-FDLP produces estimates for the symmetrized signal, and in this figure only the first half of these estimates are shown. The DCT used to obtain these estimates was Matlab's `dct`-routine which is a type II DCT.

We found experimentally that the use of type II DCT instead of type I odd DCT causes only minor differences in the resulting estimates. According to [5, p. 35] the DCT-matrices of different types are asymptotically equivalent as the signal length becomes very large. If the signal length is large and the use of other DCT-types than type I odd is preferred, one should ensure that the DCT-constants in the alternative DCT-type are defined similarly as in Definition 4.6. For example, when using Matlab's `dct`-routine, one should multiply the resulting DCT-signal with a factor $\sqrt{2/N}$ and after that divide the first element of DCT-signal by $\sqrt{2}$.

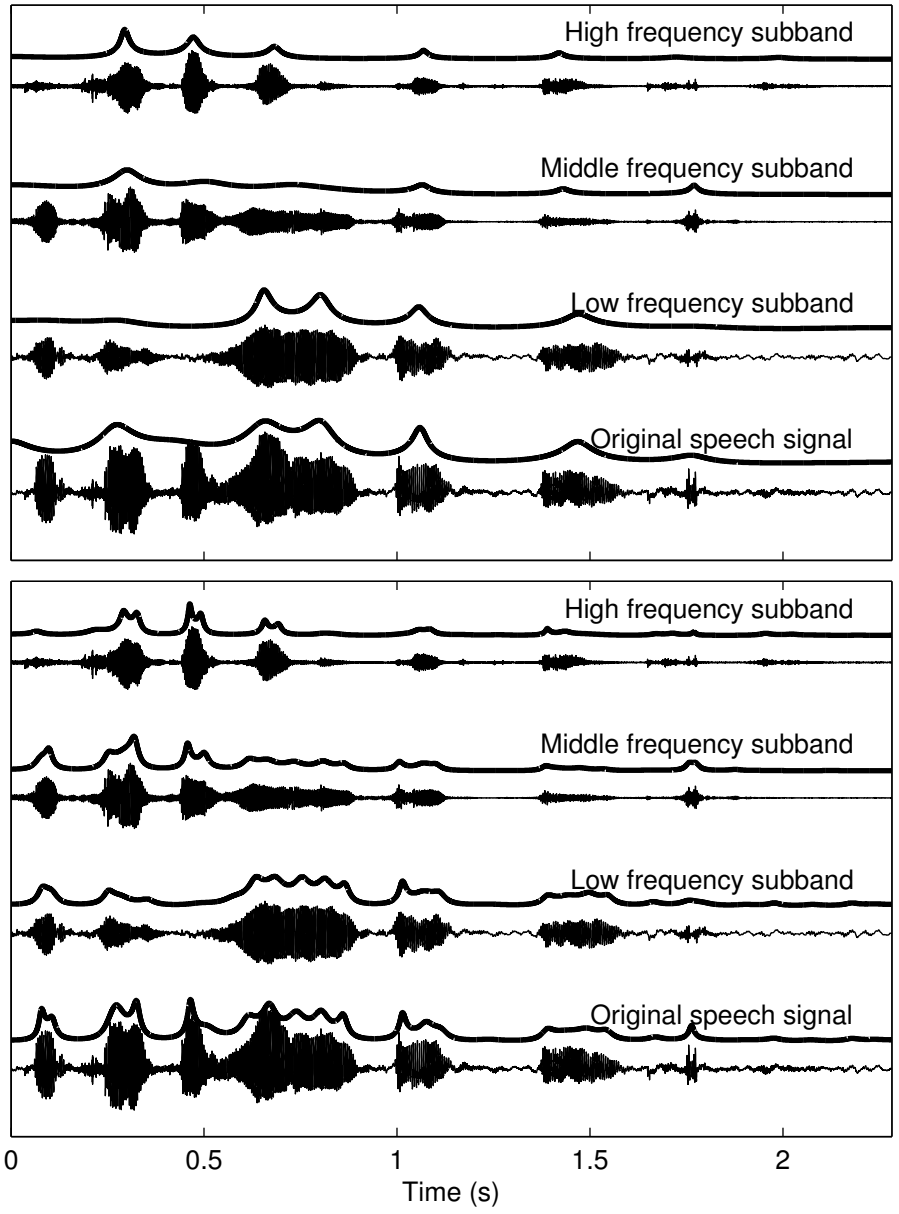

Figure 15: Estimates of the Hilbert envelopes of the speech signal and its subbands obtained by using DCT-FDLP with the model orders $p = 20$ (top) and $p = 60$ (bottom). These estimates are shifted upwards for better visibility.

4.6 Envelope modeling with IDFT-FDLP

In Theorem 2.2, we have shown that $\mathrm{IDFT}\{\mathrm{DFT}\{x[n]\}\} = x[n]$. Similarly, it can be shown that $\mathrm{DFT}\{\mathrm{IDFT}\{x[n]\}\} = x[n]$. Hence,

$$|\mathrm{DFT}\{\mathrm{IDFT}\{x[n]\}\}| = |x[n]| \tag{48}$$

and this allows us to use the FDLP on the IDFT-signal to obtain an estimate of the modulus of a time-domain signal.

We can also use the DFT and its inverse to decompose a time domain signal to multiple subbands: Let $X[k]$ be the IDFT of $x[n], n = 0, \ldots, N-1$. By defining

$$X_{a,b}[k] = \begin{cases} X[k], & \text{if } a \leq k < b \quad \text{or} \quad N-b < k \leq N-a, \\ 0, & \text{otherwise} \end{cases} \qquad (k = 0, \ldots, N-1)$$

we can write

$$X[k] = X_{0,a_1}[k] + X_{a_1,a_2}[k] + X_{a_2,a_3}[k] + \ldots + X_{a_{m-1},a_m}[k], \tag{49}$$

where $0 < a_1 < a_2 < \ldots < a_m$ and

$$a_m = \begin{cases} \frac{N}{2} + 1, & \text{if } N \text{ is even}, \\ \frac{N+1}{2}, & \text{if } N \text{ is odd}. \end{cases}$$

By taking the DFT of equation (49) we get

$$x[n] = \mathrm{DFT}\{X_{0,a_1}[k]\} + \mathrm{DFT}\{X_{a_1,a_2}[k]\} + \ldots + \mathrm{DFT}\{X_{a_{m-1},a_m}[k]\}.$$

Here the signal $x[n]$ is decomposed to multiple subbands of the form $\mathrm{DFT}\{X_{a,b}[k]\}$. These subbands are real-valued because of the symmetry of sequences $X_{a,b}[k]$; these sequences contain a selected range of the positive and the negative frequency components that correspond to each other.

Fig. 16 shows the subband signals obtained by using the IDFT and the DFT. It can be seen that these subbands match very closely to the ones obtained with the DCT shown in Fig. 11. Fig. 16 also shows the IDFT-FDLP estimates of the absolute value signals. These estimates are remarkably similar to the estimates of the Hilbert envelopes in Fig.

15. The same accuracy level of the estimates is obtained at half the model orders when compared to DCT-FDLP, since IDFT-FDLP does not produce symmetrized estimates.

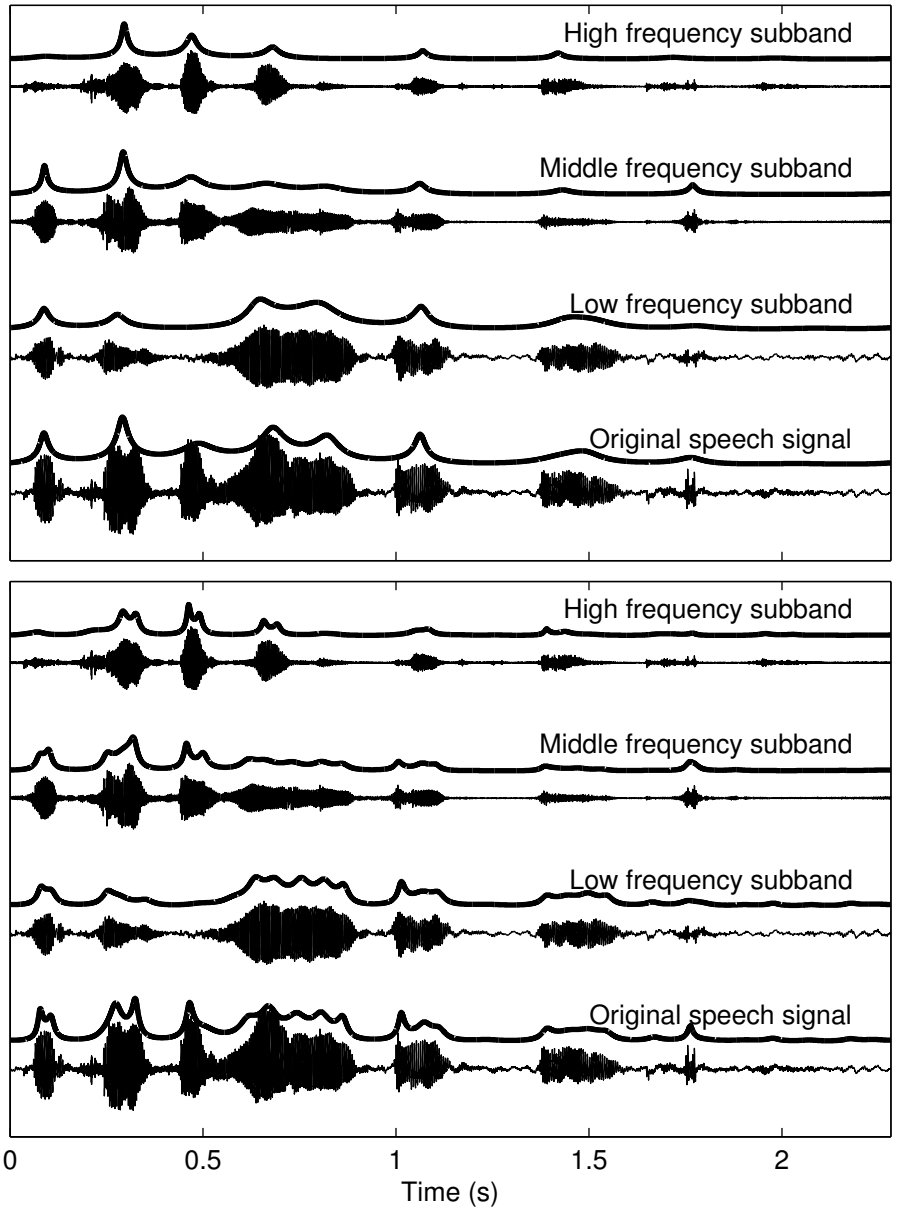

Figure 16: Estimates of the absolute value signals of a speech signal and its subbands obtained by using IDFT-FDLP with the model orders $p = 10$ (top) and $p = 30$ (bottom). These estimates are scaled by a factor 1.4 and they are shifted upwards to prevent overlapping with the signals.

The drawback of IDFT-FDLP is that it produces complex valued predictor coefficients. So even though the model order can be halved, the actual number of model parameters stays the same (complex numbers are composed of real and imaginary parts). By symmetrizing the signal $x[n]$ before IDFT, it is possible to make IDFT-FDLP work

like DCT-FDLP in a sense that it yields real valued predictor coefficients and also symmetrized estimates. In the case of symmetrized signals, the equation (48) would turn into

$$|\mathrm{DFT}\{\mathrm{IDFT}\{\mathrm{Sym}\{x[n]\}\}\}| = |\mathrm{Sym}\{x[n]\}|.$$

The reason why we get real valued predictor coefficients is that the symmetrization causes the predicted signal $\mathrm{IDFT}\{\mathrm{Sym}\{x[n]\}\}$ to be real valued.

Another problem with IDFT-FDLP is that it seems to produce envelopes that have the same value at the beginning and at the end of the signal. Therefore, if the endpoints of the signal's envelope differ, then IDFT-FDLP does not perform well near the endpoints. The use of the symmetrized version of IDFT-FDLP fixes this problem as seen from the Fig. 17.

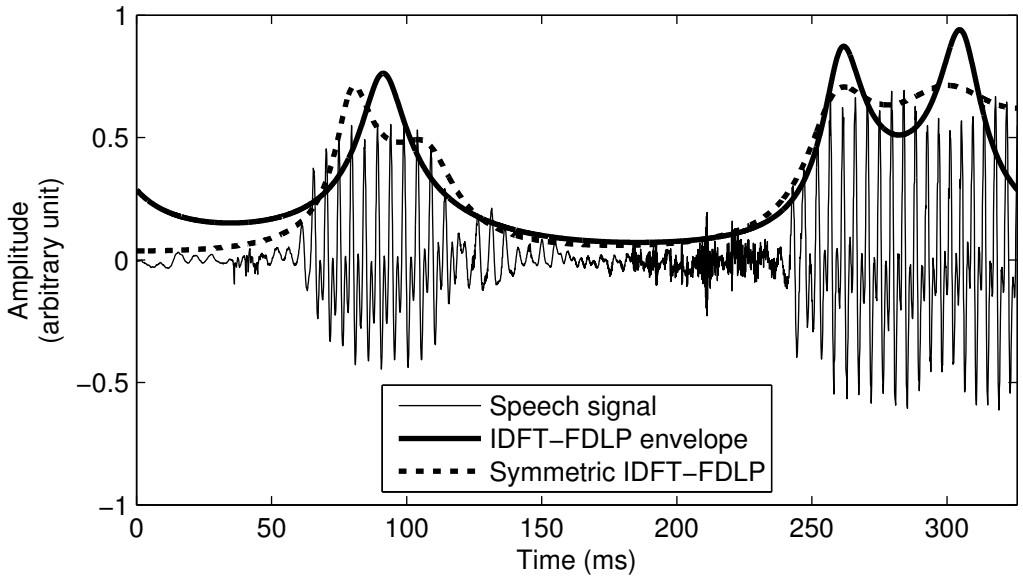

Figure 17: Envelopes of a speech signal obtained by using IDFT-FDLP and its symmetric version. The latter produces better envelopes near the endpoints.

4.7 FDLP-modeling of positive valued signals

In the previous sections, we showed how the DCT-FDLP and the IDFT-FDLP can be used to model signal's *envelope*. Let us next focus on modeling the *signal itself*, when signal is positive valued. With IDFT-FDLP, this seems straightforward, since the method models the modulus of the signal; in the case of positive valued signals, modulus signal and the signal itself are equal. On the other hand, when using the DCT-

FDLP method, we must consider what happens to the Hilbert envelope when the signal is positive valued.

Let d be a non-negative (real-valued) constant and let $x[n] = y[n] + d$, where $y[n]$ is a real-valued time-varying signal having values in a positive and bounded range, for example from 0 to 1. To get some insight how the Hilbert envelope settles on the different signal levels, let us study what happens to the Hilbert envelope of $x[n]$ when the value of d varies.

Since the Hilbert envelope is given by the modulus of the analytic signal, we turn our attention to the real and imaginary parts of the analytic signal. By Definition 4.2, the real part is the signal $x[n]$ itself, so the values of the real part change as d changes. The imaginary part, in turn, is the Hilbert transform of $x[n]$ and it does not change when d changes. This follows from the observation that only the first DFT-coefficient of $x[n]$ changes when d changes (as a constant value, d does not have effect to the oscillating components of the signal) and the value of the first DFT-coefficient does not have an effect on the Hilbert transform as can be seen from Definition 4.1.

As explained in Section 4.1, the Hilbert envelope of $x[n]$ is calculated as follows:

$$|\mathrm{An}\{x[n]\}| = \sqrt{x[n]^2 + h[n]^2},$$

where $h[n] = \mathcal{H}\{x[n]\}$. Since $x[n] \to \infty$ as $d \to \infty$ and $h[n]$ stays constant when $d \to \infty$ for all n, if follows that $\sqrt{x[n]^2 + h[n]^2} \to x[n]$ as $d \to \infty$ for all n. In other words, the higher the signal level, the more closely its Hilbert envelope matches to the signal. This phenomenon is demonstrated in Fig. 18.

Another issue that we need to consider is the condition (15) that we used to determine the gain factor of the all-pole estimate. In the case of IDFT-FDLP (DCT-FDLP), this condition means that the average ratio between the squared signal (squared Hilbert envelope) and its squared estimate is one. This causes the estimate to have a higher mean value when compared to the original signal. To elaborate, consider the following example: Suppose that we have a constant signal $\{1, 1, 1, 1, 1\}$. If the estimate has a value of 0.5 at one point, then the squared ratio at this point is $1^2/0.5^2 = 4$. To keep the average squared ratio at one, the estimate can then have, for example, value 2 at four points resulting in squared ratios $1^2/2^2 = 0.25$ in each of these four points. Then the average squared ratio is $(4 + 4 * 0.25)/5 = 1$ as required, but the mean value of

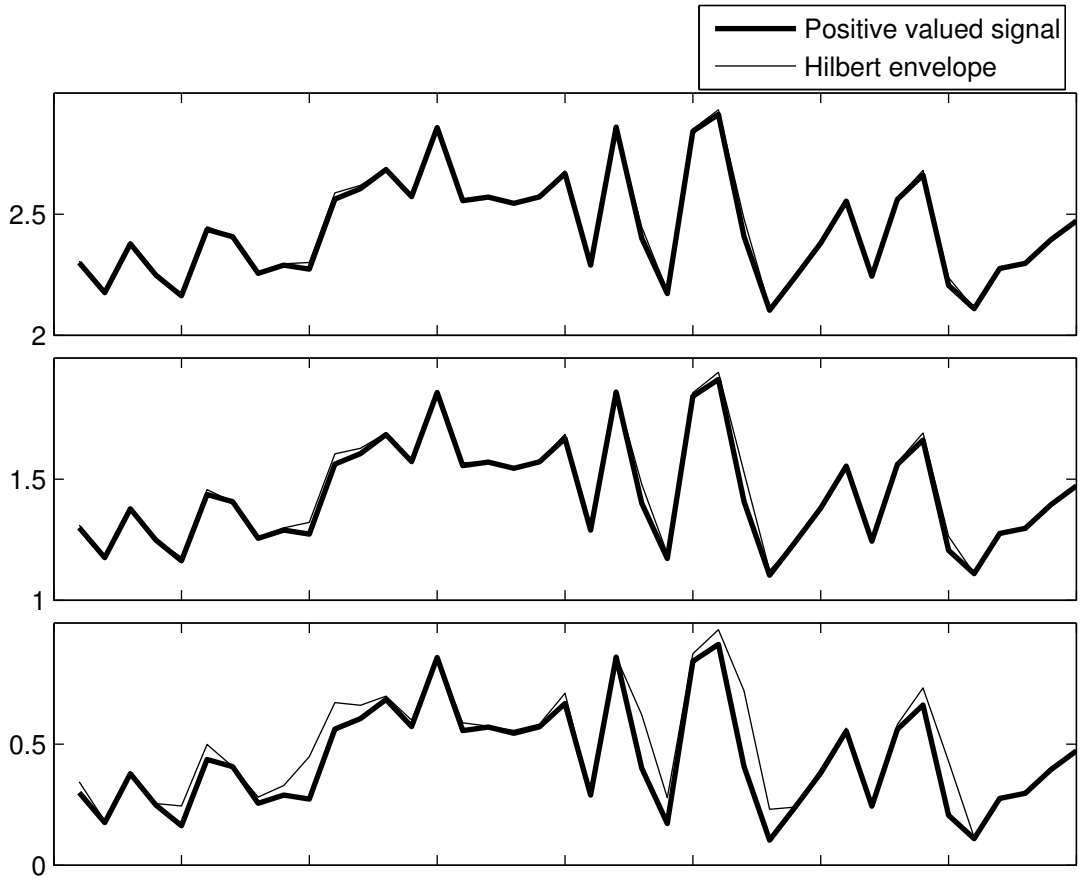

Figure 18: A signal shifted to three different amplitude levels. As the signal level increases, the Hilbert envelope converges towards the signal.

the estimate is $(0.5 + 4 * 2)/5 = 1.7$, which differs considerably from 1, which is the mean of the original constant signal.

As a practical remedy, we can shift the signal upwards by an arbitrary constant. By doing so, the difference between the mean value of the signal and the mean value of the signal's estimate gets smaller. To see this, consider a constant signal $\{5, 5, 5, 5, 5\}$. If the estimate has a value of 4.5 at one point (the same absolute error as in the first example), the resulting squared ratio is $5^2/4.5^2 \approx 1.23$. This is counterbalanced if the estimate is for example 5.7 at another point, which leads to a squared ratio $5^2/5.7^2 \approx 0.77$. Then the mean of the estimate is $(4.5 + 5.7 + 3 * 5)/5 \approx 5.04$ assuming that the estimate has the correct value at the three other points. The resulting mean (≈ 5.04) is only slightly higher than the mean of the original signal ($= 5$).

The observations made about the Hilbert envelope and about the mean values of the all-pole estimates suggest that the FDLP-modeling of positive valued signals might perform better when the signal is shifted upwards before the modeling.

5 LSF track modeling experiments

In Section 3.7, we introduced the line spectral frequencies (LSFs). Recalling that the LSFs are real and positive-valued, we are able to benefit from the methods of Chapter 4 to model their temporal characteristics. The temporal dimension for the LSFs is obtained by first splitting the speech signal into frames and by calculating the LP coefficients, and from them the LSFs, for each frame. The LP model order determines how many LP coefficients and LSFs are obtained for each frame. If the model order is p, we obtain p LSFs per frame. From these values, we can then form p curves in the time domain, which we will refer to as *LSF tracks*.

In Fig. 19, a model order $p = 10$ was used, resulting in ten LSF tracks. These tracks were formed from a 400 ms long speech signal. The signal was divided into 25 ms frames with a 10 ms overlap, leading to a total number of 28 frames. Thus, each of the LSF tracks in Fig. 19 contains only 28 data points, which makes these LSF tracks look rather jagged.

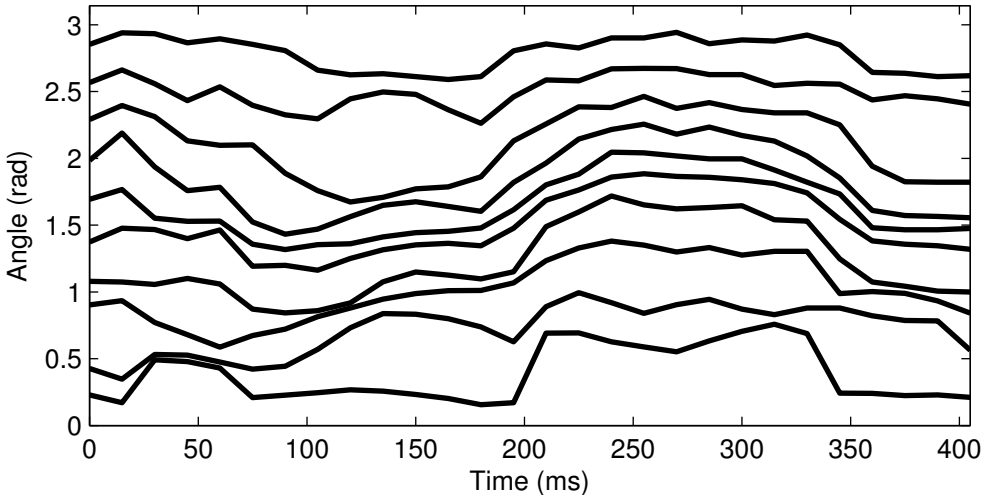

Figure 19: Ten LSF tracks obtained from a recording of a word 'greasy'.

In the next few sections, we describe the methods that we use to obtain parametrized representations of the LSF tracks. To measure the goodness of these parametrizations, we consider three different objective error measures, which we will present in Section 5.4. At the end of this chapter, we describe the LSF track modeling experiment on which we tested the different parametrization methods. And finally, the results and the related discussion of this experiment are given.

5.1 LSF track fitting with DCT

One possible parametric representation of a single LSF track is given by the discrete cosine transform (DCT). To reduce the parameter number to any given number r, we simply zero out all but the first r DCT-coefficients. The zeroing of the (high frequency) DCT-coefficients results in LSF track estimates that do not have sharp variations. Listings 1 and 2 show Matlab code examples of the parametrization process using the type II and type I odd DCTs (see Section 4.3), respectively. The type II version takes advantage of Matlab's[1] built-in `dct`-function and the type I odd transformation is implemented as a matrix multiplication.

```
function estLSF = dct2Estimates(LSF, nParams)
% Columns of LSF are the LSF tracks
d = dct(LSF);
d(nParams + 1 :  end, :) = 0;
estLSF = idct(d); %Track estimates are given by the IDCT
end
```

Listing 1: Obtaining LSF track estimates using type II DCT.

```
function estLSF = dct1Estimates(LSF, nParams)
% DCT-I-odd matrix:
N = size(LSF, 1); %Length of LSF tracks
M = 2 * N - 1;
D = 4 / M * cos((0 : N-1)' * (0 : N-1) * 2 * pi / M);
D(1, :) = D(1, :) / 2;
D(:, 1) = D(:, 1) / 2;
% DCT-I-odd inverse matrix:
DI = cos((0:N-1)' * (0:N-1) * 2 * pi / M);

d = D * LSF;
d(nParams + 1 :  end, :) = 0;
estLSF = DI * d;
end
```

Listing 2: Obtaining LSF track estimates using type I odd DCT.

[1] Matlab version: R2014a

Fig. 20 shows the LSF-estimates given by both types of DCT. As expected, these estimates differ only slightly from each other. Both of them capture the shapes of the LSF tracks reasonably well considering the fact that only 7 parameters are used to represent 28 values.

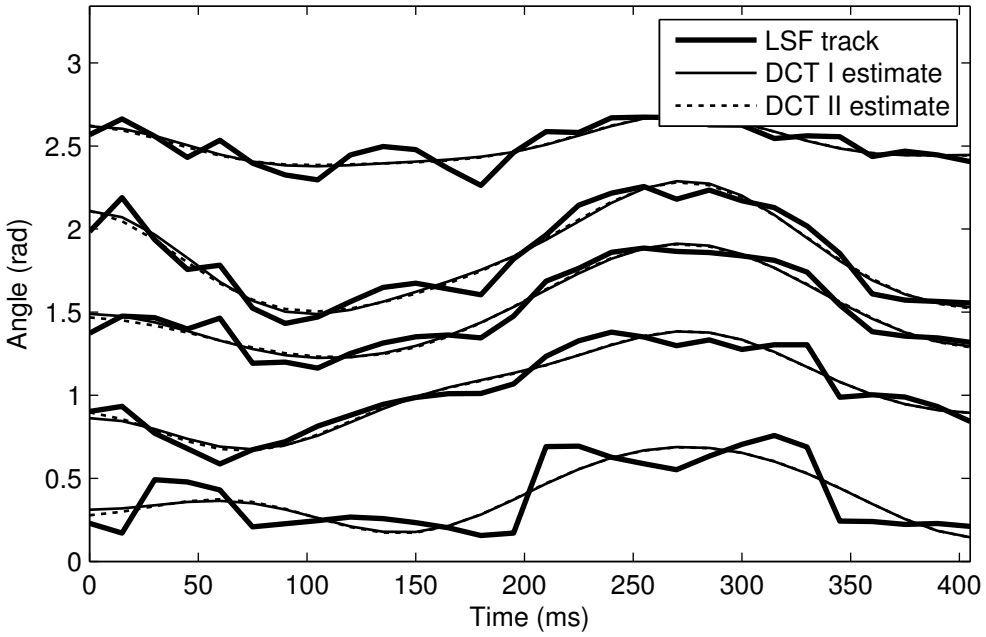

Figure 20: LSF track estimates obtained using type I and type II DCTs. Each of the tracks is represented with 7 parameters. Type I and type II estimates have only minor differences between them. To increase the readability of the figure, only every other LSF track is shown.

5.2 LSF track fitting with DCT-FDLP

The theoretical basis of the DCT-FDLP method was explained in Chapter 4. A Matlab implementation of this method is given in Listing 3. This implementation begins with a calculation of zero-padded (type I) DCTs for each track (lines 6–12). Then the autocorrelation sequences are computed efficiently by using the *fast Fourier transform* (FFT) (lines 15–16). This way of computing autocorrelation values is based on a relation between the autocorrelation sequence and the signal's power spectrum, which is given as follows [9, 19]:

$$\mathrm{DFT}\{R[l]\} = |X[k]|^2.$$

In the case of DCT-FDLP we calculate the autocorrelation values from the DCT-transformed signals. Next, the predictor coefficients are solved from the Toeplitz sys-

tem by using the *Levinson-Durbin algorithm* (line 18). Then, the gain coefficients are computed for each track, and finally the track estimates are formed from the computed parameters by evaluating the right-hand side of Equation (14). The denominator of (14) contains the DFT of the LP coefficient vector $a[l]$ where $a[0] = -1$. Thus, it can be calculated efficiently by using FFT. Note that Matlab's `levinson`-function outputs predictor coefficients that have opposite signs compared to our definition.

```
1  function estLSF = dctFdlpEstimates(LSF, nParams)
2  [N, nTracks] = size(LSF); %Columns of LSF are the LSF tracks
3  p = nParams - 1; %LP model order; one param is the gain factor
4
5  % DCT-io-forward matrix
6  M = 2 * N - 1;
7  D = 4 / M * cos((0:N-1)' * (0:N-1) * 2 * pi / M);
8  D(1, :) = D(1, :) / 2;
9  D(:, 1) = D(:, 1) / 2;
10
11 d = D * LSF;
12 d = [d; zeros(N - 1, nTracks)]; %zero-padded DCTs
13
14 % Autocorrelations (R) can be calculated using DFT:
15 X = fft(d, 2^nextpow2(2*M-1));
16 R = ifft(abs(X).^2);
17
18 A = levinson(R, p); %Predictor coefficients
19 G = sqrt(sum(A' .* R(1:p+1, :), 1)); %Gain factors
20
21 estLSF = repmat(G, M, 1) ./ abs(fft(A', M)); %Track estimates
22 estLSF = estLSF(1:N, :); %Cutting out the symmetric parts
23 end
```

Listing 3: Obtaining LSF track estimates using DCT-FDLP.

Fig. 21 shows how the DCT-FDLP estimates fit to the LSF tracks. The estimates of the lowest tracks have, on average, slightly too high values, while the estimates of the highest tracks fit better. This goes along with the observations made in Section 4.7. In Fig. 22, the lowest LSF track in Fig. 21 is artificially shifted to different levels (see the y-axis values) and after that, the DCT-FDLP estimates are fitted. An interesting observation was made while trying to fit DCT-FDLP estimates at different LSF track

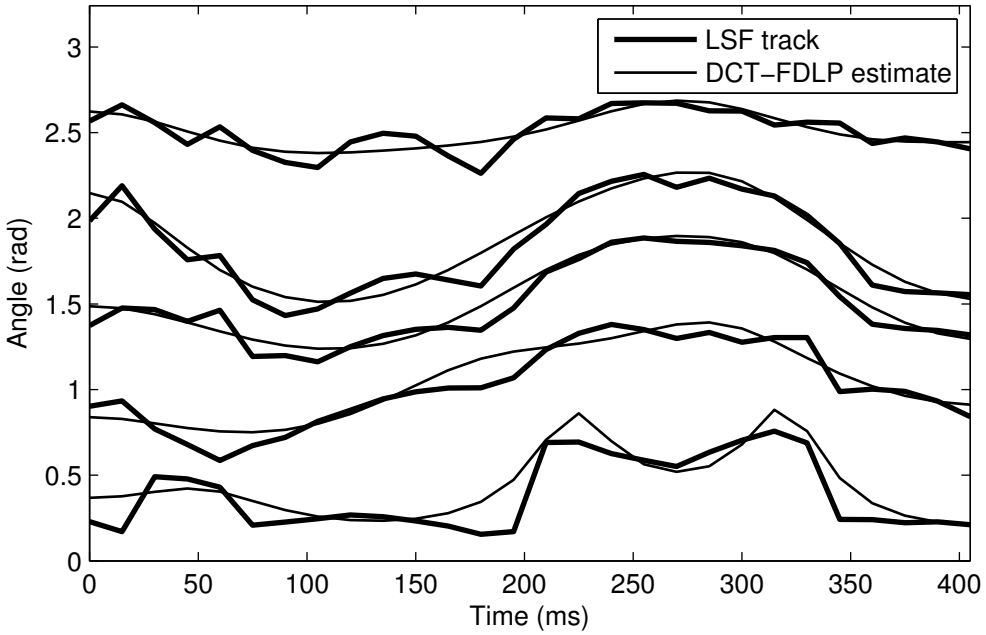

Figure 21: LSF track estimates obtained using DCT-FDLP. Each of the tracks is parametrized with six predictor coefficients and one gain coefficient.

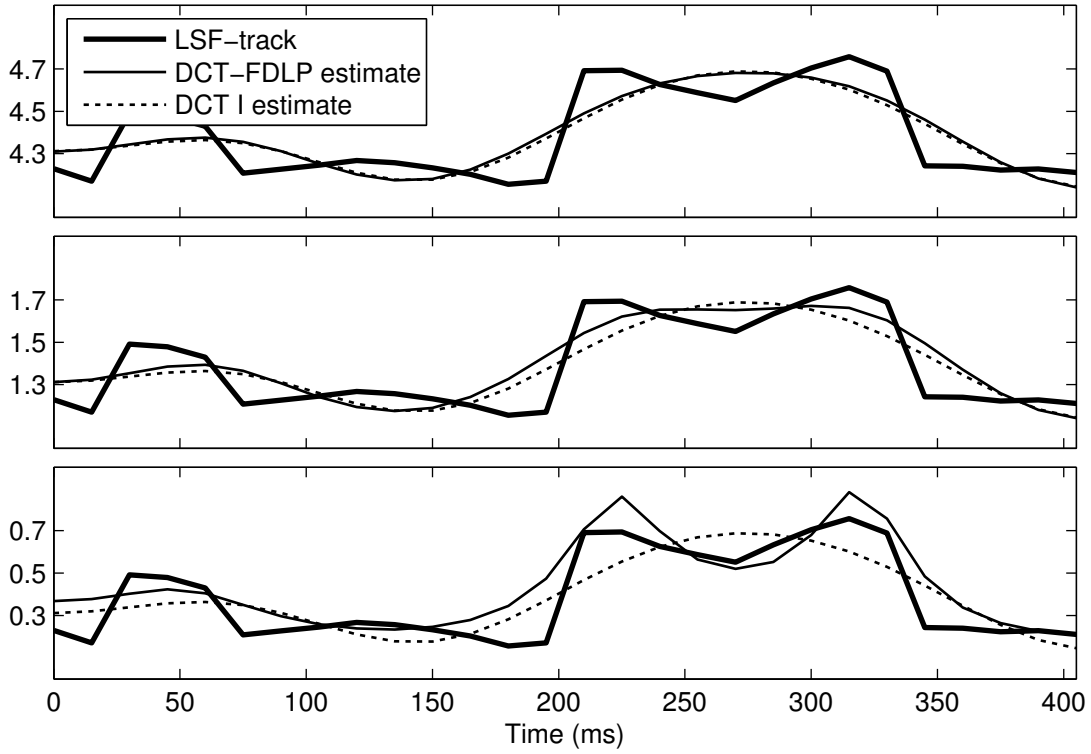

Figure 22: The DCT-FDLP and the DCT I estimates of the lowest LSF track in Fig. 21 after shifting the track to different levels. Shifting does not have any effect to the DCT estimate, unlike to the DCT-FDLP estimate, which seems to converge towards the DCT estimate as the signal level rises.

levels: The higher the LSF track lies, the more its DCT-FDLP estimate starts to resemble the type I DCT estimate (see Fig. 22). Based on these experimental observations, it seems to be possible that the DCT-FDLP estimate converges towards type I DCT estimate when the modeled signal is shifted upwards. Note that this only happens when signal moved upwards by adding a constant value. If the addition is replaced by a multiplication, the DCT-FDLP estimate retains its original shape.

5.3 LSF track fitting with IDFT-FDLP

The next method that we use to obtain LSF track parametrizations is IDFT-FDLP, which was discussed in Section 4.6. A Matlab implementation is provided in Listing 4. It uses the symmetric version of IDFT-FDLP, which results in better fitting estimates near the endpoints (see the end of Section 4.6).

```
1  function estLSF = idftFdlpEstimates(LSF, nParams)
2  p = nParams - 1; %LP model order; one param is the gain factor
3  N = size(LSF, 1); %Length of the LSF tracks
4  M = 2 * N - 1;
5
6  symLSF = [LSF; flipud(LSF(2:end, :))];
7  f = ifft(symLSF);
8  f(2:end, :) = 2 * f(2:end, :);
9
10 X = fft(f, 2^nextpow2(2*M-1));
11 R = ifft(abs(X).^2); %Autocorrelation sequences
12 A = levinson(R, p); %Predictor coefficients
13 G = sqrt(sum(A' .* R(1:p+1, :), 1)); %Gain factors
14 estLSF = repmat(G, M, 1) ./ abs(fft(A', M)); %Track estimates
15 estLSF = estLSF(1:N, :); %Cutting out the symmetric parts
16 end
```

Listing 4: Obtaining LSF track estimates using IDFT-FDLP.

In line 8 of Listing 4, all but the first IDFT-coefficient are multiplied by two. This is done because in the experiments, we found that IDFT-FDLP estimates tend to be too flat. The first IDFT-coefficient represents the mean value of the LSF track, so leaving it unchanged keeps the estimate at the correct level. By multiplying the rest of the IDFT-components, we multiply the oscillating components, resulting in a larger range

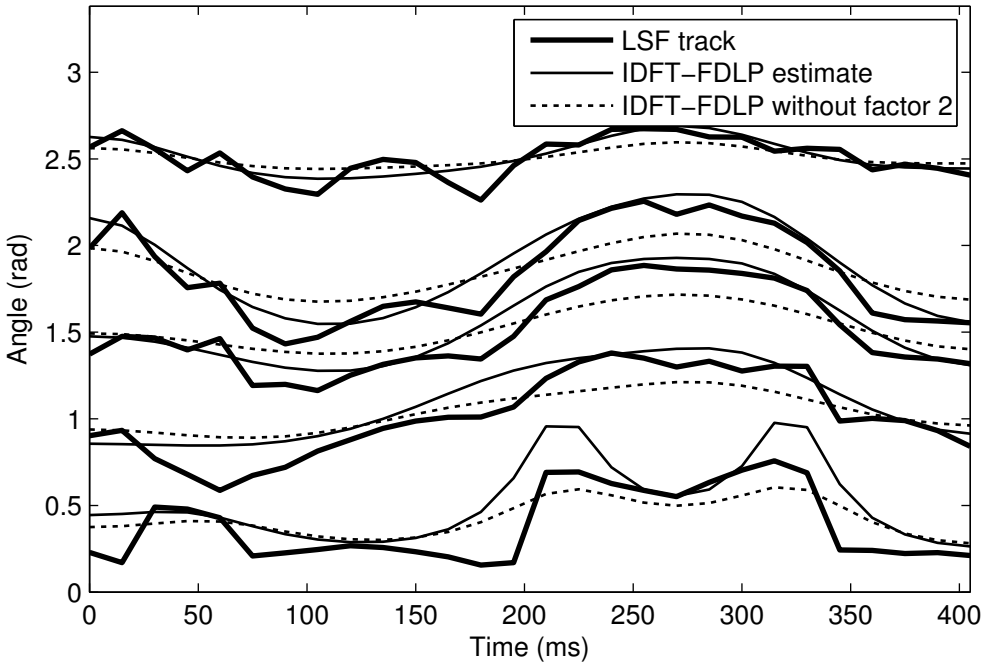

Figure 23: LSF track estimates obtained using IDFT-FDLP with and without the IDFT-coefficient factor 2 (Line 8 in Listing 4). If the IDFT-coefficients are not multiplied by two, estimates are too flat.

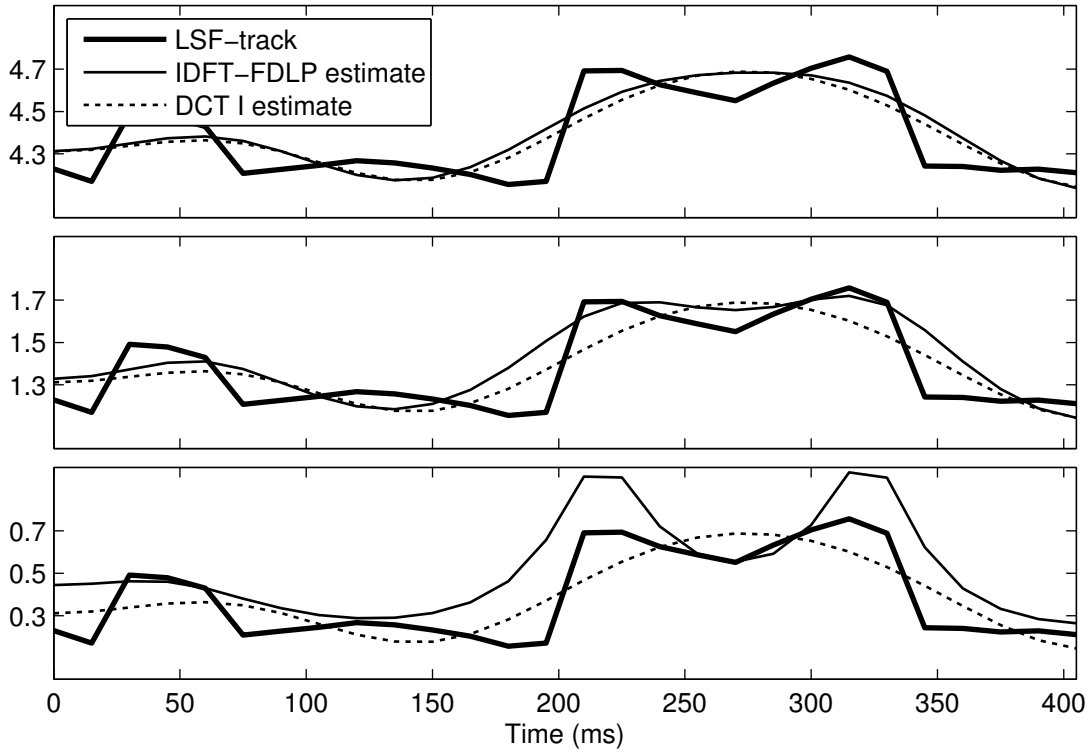

Figure 24: The IDFT-FDLP and the DCT I estimates of the lowest LSF track in Fig. 23 after shifting the track to different levels. Like the DCT-FDLP estimate, the IDFT-FDLP estimate seems to converge towards the DCT estimate as the signal level rises.

of values in the estimate.

Fig. 23 shows the IDFT-FDLP fittings with and without line 8 in Listing 4. The estimates obtained without the IDFT-coefficient multiplier 2 are too flat. Both estimates for the lowest LSF tracks are located too high, as was the case with DCT-FDLP. As the signal level increases, the IDFT-FDLP estimate (with factor 2) starts to resemble more and more the DCT type I estimate (see Fig. 24).

5.4 Error measures

Up to this point, we have studied different LSF track modeling variants using illustrative examples. To study their differences objectively, we will compare them on a large set of speech signals and adopt three error measures to analyze the goodness of the different LSF track modeling methods.

The first two error measures are the *mean absolute error* (MAE) and the *root mean squared error* (RMSE) between the original LSF track and its estimate. These error measures are defined as follows: Let $L[n], n = 1, \ldots, N$, be the original LSF track and let $\tilde{L}[n]$ be its estimate. Then the MAE (e_1) and the RMSE (e_2) measures between $L[n]$ and $\tilde{L}[n]$ are defined as

$$e_1 = \frac{1}{N} \sum_{n=1}^{N} \left| L[n] - \tilde{L}[n] \right|,$$

$$e_2 = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \left(L[n] - \tilde{L}[n] \right)^2}.$$

Since we have many LSF tracks for each speech segment, we further take the mean value of the MAEs and RMSEs of the tracks as an overall summary value.

The third error measure compares the original short-term all-pole spectra to the estimated all-pole spectra, which are formed from the LP coefficients that are obtained by converting the modeled LSF tracks back to LP coefficients. The difference between the all-pole spectra is determined by calculating their *log spectral distance* (LSD), which is the RMSE of the spectra in decibel scale. The exact definition of the LSD is given as follows: Let $S[n], n = 1, \ldots, N$, be an all-pole magnitude spectrum and let $\tilde{S}[n]$ be

its estimate. Then the LSD between $S[n]$ and $\tilde{S}$ is

$$e_2 = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(10\log_{10}S[n] - 10\log_{10}\tilde{S}[n]\right)^2} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(10\log_{10}\frac{S[n]}{\tilde{S}[n]}\right)^2}.$$

Again, since we have all-pole spectra for all the frames in a speech segment, we finally take the mean value of all the LSDs. Listing 5 contains a Matlab script in which we calculate the mean LSD value when modeling the LSF tracks with the type I DCT. The beginning of the script shows how to form the LSF tracks from the predictor coefficients.

```
% A and G contain parameters for the original all-pole spectra.
% Each row of A contains LP coefficients for one frame.
% G is a row vector containing gain factors for each frame.
% M is the frame size, N = floor(M / 2) + 1.

[nFrames, p] = size(A);
p = p - 1;

% Creating LSF tracks from LP coefficients:
LSF = zeros(nFrames, p); % Columns of LSF are the LSF tracks
for i = 1 : nFrames
    LSF(i, :) = poly2lsf(A(i, :)'); %LPC to LSF conversion
end

nParams = 10;
estLSF = dct1Estimates(LSF, nParams); %DCT I estimates of LSF tracks

estA = lsf2poly(estLSF'); %LSF to LPC conversion

spectra = repmat(G, M, 1) ./ abs(fft(A', M)); %Original spectra
spectra = spectra(1:N, :); %Cutting out the symmetric parts
estSpectra = repmat(G, M, 1) ./ abs(fft(estA', M)); %Estim. spectra
estSpectra = estSpectra(1:N, :); %Cutting out the symmetric parts

LSDs = sqrt(sum((10 * log10(spectra ./ estSpectra)).^2, 1) ./ N);
meanLSD = mean(LSDs);
```

Listing 5: Calculation of LSDs when when modeling the LSF tracks with the type I DCT.

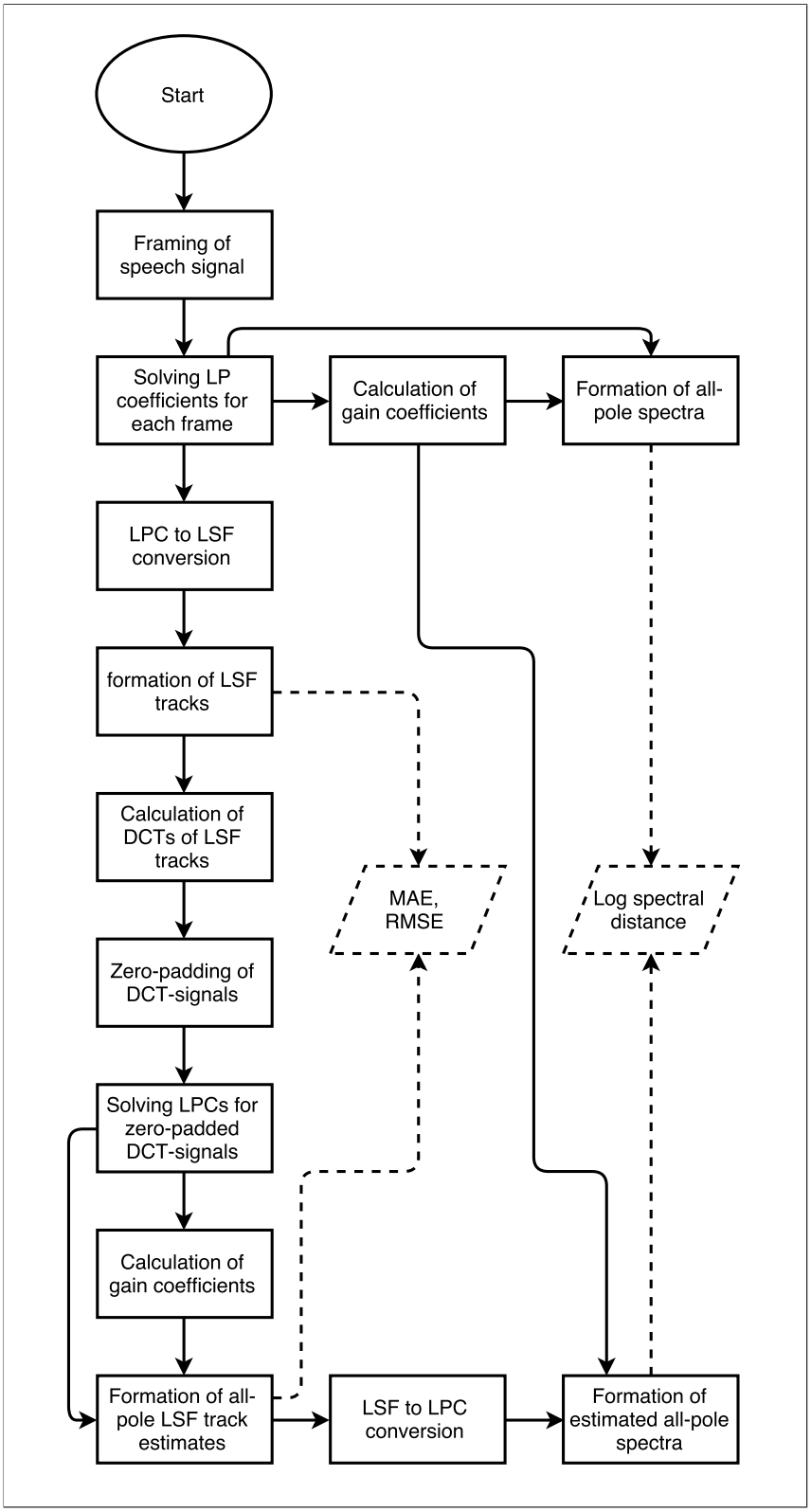

Figure 25: A flow chart showing the steps needed to form the LSF tracks and their estimates, and the steps needed to calculate the errors.

In Fig. 25 we present a flow chart that shows the process of forming LSF tracks and their estimates, and the steps needed to calculate the presented errors.

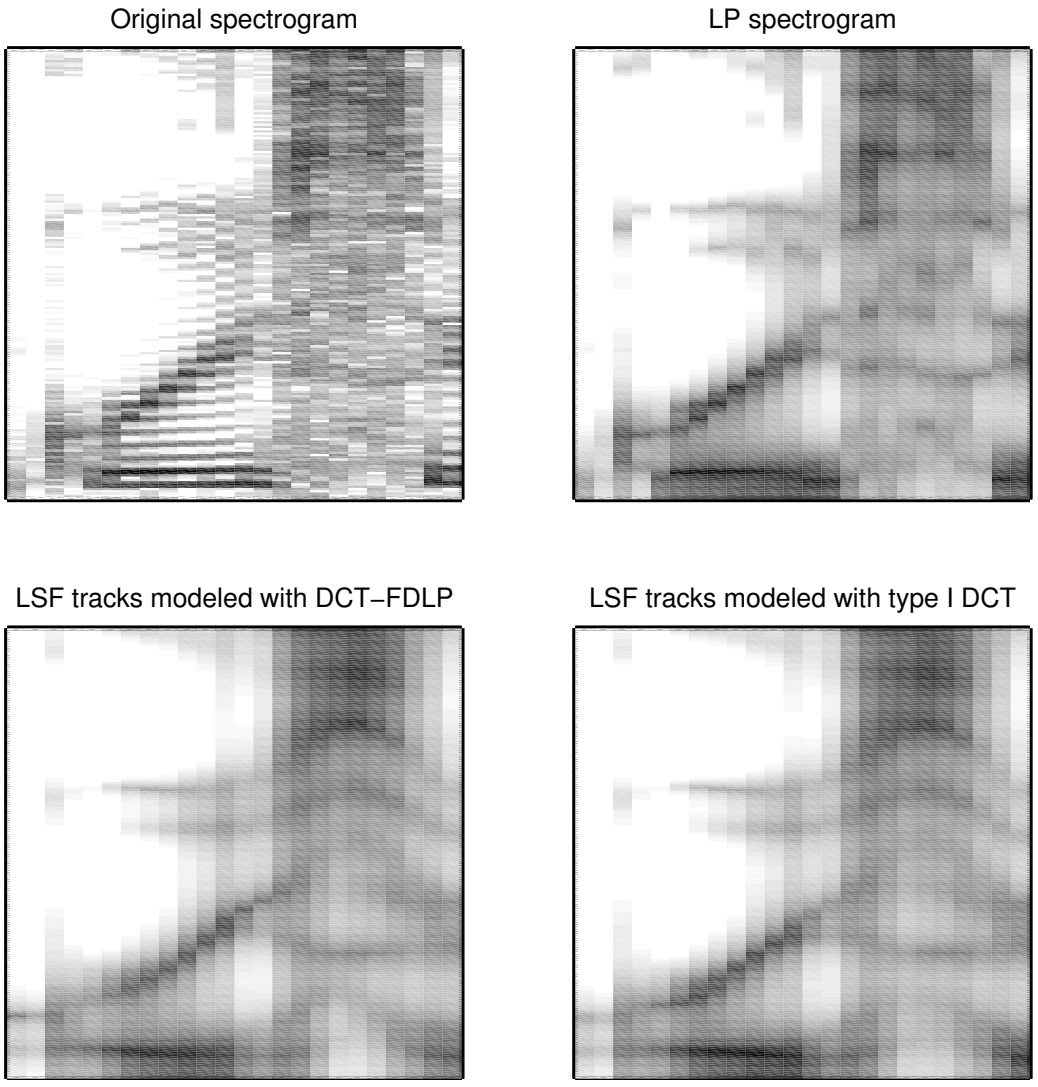

Figure 26: Spectrograms of a word 'greasy' after different modeling steps. The LP spectrogram was obtained using Hamming windowed 25 ms long frames with a 10 ms overlap. In both LSF track modeling methods, six parameters were used to model the tracks.

An example that shows the changes in the frequency data after different modeling steps is given in Fig. 26. The top-left spectrogram is a typical spectrogram, where the Hamming window is applied to each frame before calculating the DFT. On the top-right spectrogram (LP-spectrogram), each vertical line represents an all-pole spectrum that is obtained by using linear prediction for each (Hamming-windowed) frame. We can think that the LP-modeling step blurs the original spectrogram in the vertical dimension. The two spectrograms in the bottom show how the data is modified after the LSF

track modeling step. In these steps the blurring effect is mostly horizontal. To obtain the bottom spectrograms we needed to convert modeled LSF tracks back to LP coefficients as done in Listing 5. From these two spectrograms, we gain some confidence that the LSF-modeling phase will not disturb the data so drastically that it would lose its most prominent characteristics.

5.5 Experiment design

We used the TIMIT corpus [14] to compare the previously described LSF track models. The speech data in TIMIT is recorded in a quiet room (noise-free) with a high quality microphone [29, p. 14] and it is sampled at 16 kHz. We randomly chose 50 male and 50 female speakers from the corpus. For each speaker, we selected the two common sentences (denoted by SA1 and SA2 in the TIMIT corpus) that were spoken by all the speakers. These sentences were further split to individual words with the help of annotation files available in TIMIT. We then modeled the LSF tracks of these words with different methods. The resulting errors were averaged over different speakers while keeping different words and genders separated.

We compared six different LSF track models. Two of them used the type I and type II DCTs as described in Section 5.1. Then the DCT-FDLP and IDFT-FDLP methods were used as described in Sections 5.2 and 5.3, respectively. The last two methods were modified versions of the FDLP methods. In these methods, all the LSF tracks are first shifted to the level of the highest LSF track and only after that FDLP is applied. The shifting is done as follows: If the number of the LSF tracks is p, then the second highest LSF track is shifted π/p radians upwards, the third highest LSF track is shifted $2\pi/p$ radians, the fourth highest is shifted $3\pi/p$ radians, and so on. Since LSF tracks are typically quite uniformly spaced between zero and π, this procedure moves all the LSF tracks close to the highest LSF track, which lies just below π.

The reason why we shift LSF tracks upwards was explained in Section 4.7. Figs. 22 and 24 show examples of the effect of the shifting. By shifting LSF tracks to the π-level, we expect to have similar results than with the DCT methods. Then, the differences between the DCT and FDLP methods are mainly in the parameters that the methods produce (DCT-coefficients versus LP coefficients).

Typically, magnitude spectra of speech segments have an overall decreasing slope to-

wards higher frequencies. We pre-emphasized all of the speech samples in order to make the spectra more flat. Without pre-emphasizing, LSF tracks tend to be more densely located at the low levels, but after pre-emphasizing, LSF tracks are more uniformly spaced between zero and π. The explanation for this was given in the end of Section 3.7.

After pre-emphasis, the words were Hamming-windowed to 25 ms long frames with a 10 ms overlap. For each frame we then computed the LP coefficients using a model order $p = 20$, resulting in 20 LSF tracks. The total number of parameters were set to six in each of the LSF track modeling methods.

5.6 Results and discussion

Table 1 shows LSF track modeling errors of different methods for three words spoken by all the 100 speakers (50 for both genders). As hypothesized, the FDLP-methods, where the LSF tracks are shifted just below π-level, produce very similar results to the DCT-variants. The differences between the two types of the DCTs are negligible as are the differences between the DCT methods and the shifted FDLP methods. The FDLP methods, where the tracks are not shifted, yield larger errors. When modeling the lowest LSF tracks, IDFT-FDLP performs worse than DCT-FDLP, and therefore there exists clear the differences in the results between the the FDLP methods in the non-shifted cases. We also find that all three error measures tend to give consistent results with each other.

In Table 1 the average word lengths for each word and gender is given. Since we used six parameters in LSF track fitting no matter what the word length was, the errors are, in general, higher for the words that have longer duration. Despite of this, the word “dark” has the highest error values (at least for women) even if it has the shortest average duration.

Table 1: LSF track modeling errors for words “greasy”, “dark” and “suit”. The FDLP-methods marked with π are the ones where the LSF tracks are shifted below the π-level.

Word "greasy"						
Gender	Female			Male		
Avg. length (ms)	402			390		
	MAE	RMSE	LSD	MAE	RMSE	LSD
Type I DCT	0.0309	0.0391	0.5718	0.0292	0.0368	0.5414
Type II DCT	0.0311	0.0391	0.5743	0.0292	0.0367	0.5420
DCT-FDLP (π)	0.0310	0.0392	0.5736	0.0290	0.0367	0.5389
IDFT-FDLP (π)	0.0310	0.0392	0.5735	0.0291	0.0368	0.5403
DCT-FDLP	0.0321	0.0407	0.5945	0.0301	0.0381	0.5630
IDFT-FDLP	0.0358	0.0453	0.6762	0.0345	0.0433	0.6592

Word "dark"						
Gender	Female			Male		
Avg. length (ms)	336			332		
	MAE	RMSE	LSD	MAE	RMSE	LSD
Type I DCT	0.0322	0.0412	0.5858	0.0291	0.0375	0.5392
Type II DCT	0.0322	0.0410	0.5859	0.0292	0.0374	0.5415
DCT-FDLP (π)	0.0322	0.0411	0.5865	0.0291	0.0374	0.5408
IDFT-FDLP (π)	0.0322	0.0412	0.5871	0.0292	0.0375	0.5406
DCT-FDLP	0.0326	0.0415	0.6008	0.0297	0.0380	0.5583
IDFT-FDLP	0.0350	0.0452	0.6587	0.0324	0.0418	0.6234

Word "suit"						
Gender	Female			Male		
Avg. length (ms)	367			345		
	MAE	RMSE	LSD	MAE	RMSE	LSD
Type I DCT	0.0265	0.0341	0.4928	0.0250	0.0323	0.4694
Type II DCT	0.0265	0.0341	0.4938	0.0251	0.0323	0.4706
DCT-FDLP (π)	0.0265	0.0342	0.4937	0.0250	0.0324	0.4695
IDFT-FDLP (π)	0.0265	0.0343	0.4940	0.0250	0.0324	0.4696
DCT-FDLP	0.0270	0.0353	0.5046	0.0259	0.0340	0.4866
IDFT-FDLP	0.0310	0.0406	0.5789	0.0310	0.0406	0.5850

6 Application to automatic speaker verification

Typically, automatic speaker verification (ASV) systems utilize feature vectors extracted from short-duration frames that are about 25 milliseconds long [10, p. 295]. In this chapter, we present ASV systems that are based on extracting features from substantially longer frames (up to 250 ms). In these systems, we form the feature vectors by parametrizing the LSF-tracks (see Chapter 5) of longer "superframes" that are formed from consecutive short-term frames.

We begin this chapter by explaining the framing and feature extraction processes. Then, we describe the used speaker model and the arrangement of the test data, after which we investigate how the various parameter settings for the proposed ASV systems perform. Then, we introduce *mel frequency cepstral coefficient* (MFCC) and LSF based short-term features so that we can compare their performances against the long-term features. At the end of this chapter, we also present the results obtained by fusing different types of ASV systems with each other.

6.1 Feature vector extraction

Let us go through the feature extraction steps needed to obtain the proposed LSF track features. First, we consider formation of the long-duration frames of LSF tracks (*LSF frames*), from which the features are extracted. An illustration of this process is provided in Fig. 27. The first step is to create typical short-duration frames from the pre-emphasized ($\alpha = 0.97$) input signal. Here, we use 25 ms long Hanning-windowed frames with an overlap of 10 ms. Then, for each frame, we calculate the LP coefficients using the autocorrelation method explained in Section 3.3 and convert them to LSFs. From all the LSFs, we form the LSF tracks, which are finally framed with a 100 – 250 ms long rectangular window to obtain *LSF frames*. Each LSF frame contains multiple LSF tracks, or to be specific, framed segments of the original LSF tracks. The number of LSF tracks is determined by the model order used in the LP modeling step.

Fig. 28 shows how the feature vectors are formed from the LSF frames. First, each LSF track is modeled with the DCT or DCT-FDLP as explained in Sections 5.1 and 5.2, respectively. When using DCT-FDLP, we shift all the LSF tracks near to the level $\pi/2$ similarly as explained in Section 5.5. In preliminary experiments, we found that

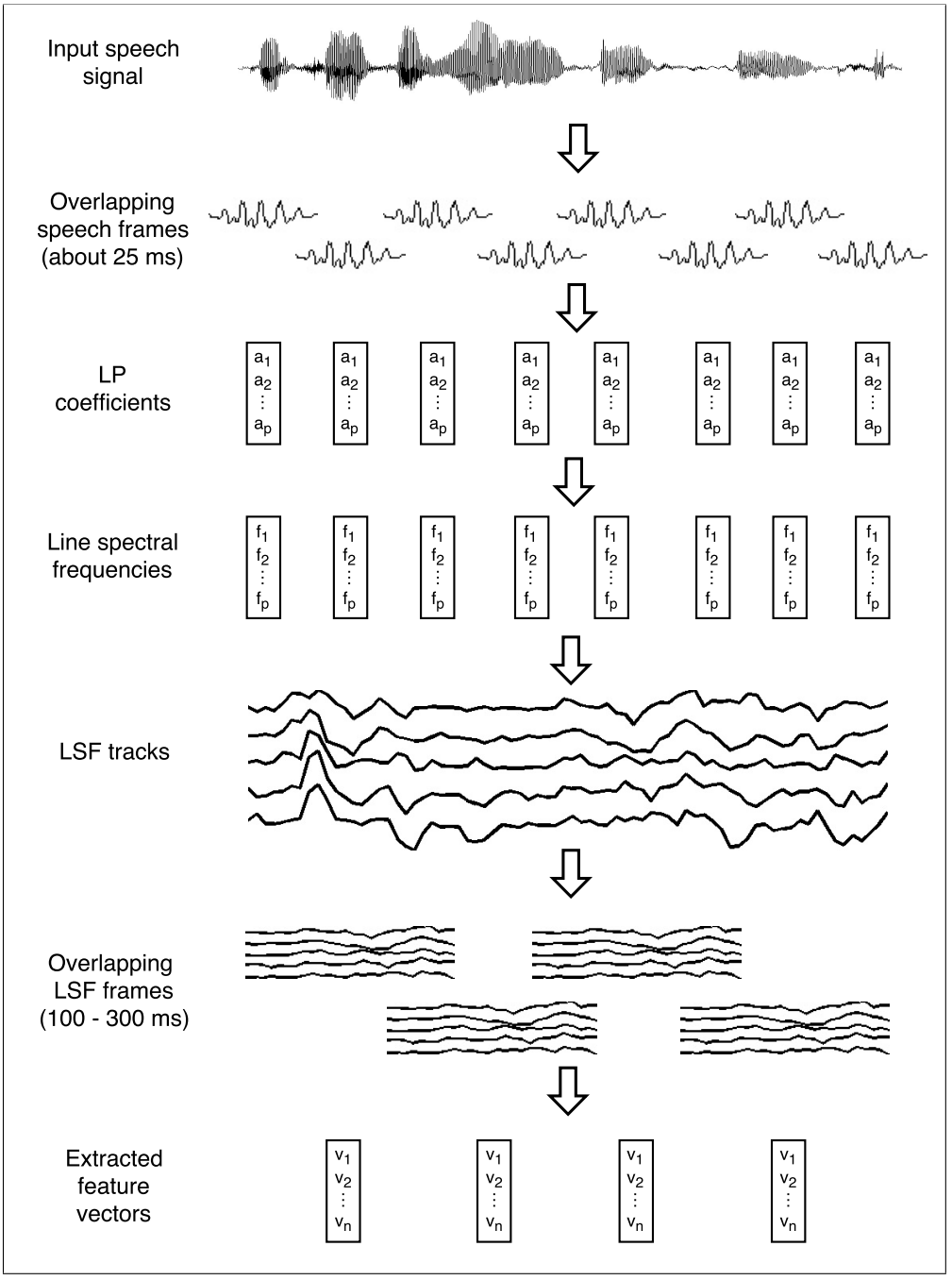

Figure 27: The formation of LSF frames. The last feature extraction step is given in more detail in Fig. 28

.

this improves the performance of the ASV-system considerably. LSF track shifting was found to be beneficial in Chapter 5, but it is still unclear for us why shifting to higher levels did not work as well.

After LSF track modeling, we concatenate the resulting parameters to a one long vector of LSF track parameters. The length of this vector is the product of the LP model order p and the LSF track model order r. As the last step, we apply DCT to decorrelate the parameters and to further reduce the dimensionality of the final feature vector. In preliminary experiments, we also tried to decorrelate the parameters by using the basis vectors given by the *principal component analysis* (PCA). The basis vectors were computed from the same data that was used to create the universal background model (UBM) (see Section 6.2). Unfortunately, we were not able to obtain as good results with the PCA as with the DCT.

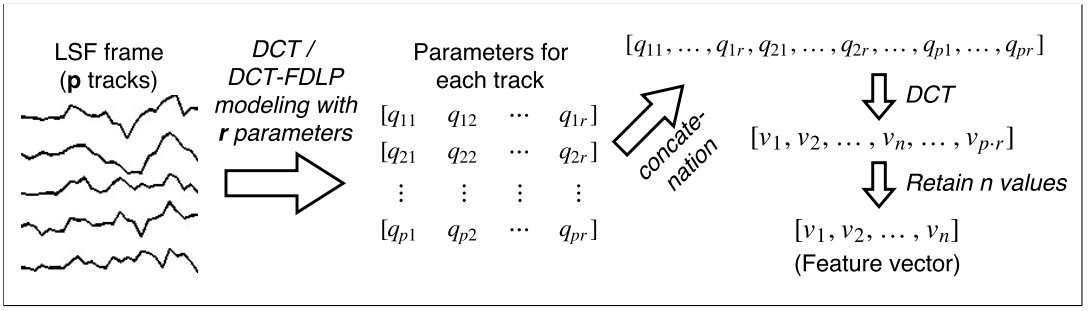

Figure 28: Extraction of a feature vector from an LSF frame. Depending on the system, either the DCT or the DCT-FDLP method is used in the modeling of the LSF tracks.

By now, it has become apparent that we have a relatively large number of parameters to be empirically determined in the ASV-system. These include the length and the overlap of the short and long duration frames, the LP model order, the LSF track modeling order and the desired feature vector dimensionality after the final DCT. Furthermore, some of these parameters are interrelated. For example, if the length of the LSF frame is increased, then the LSF track modeling order needs to be increased in order to retain the same level of modeling precision.

We simplify the parameter selection problem by fixing the frame length and overlap for the short frames to 25 and 10 milliseconds, respectively. These values are commonly used in short-term speech analysis. In our initial experiments, we found that changing these values did not cause large variations to the recognition results. We also learned that the best overlap value for the LSF frames is the maximum one. This means that the adjacent LSF frames should have only one different LSF value with each other in

each of the LSF tracks to be optimally overlapped. The use of the maximum overlap maximizes the amount of obtained feature vectors for our statistical classifier, which is the probable reason why it yielded the best results.

6.2 Experimental set-up

We selected the *Gaussian mixture model* (GMM) with *universal background model* (UBM) (popularly known as GMM-UBM in the speech processing literature) to model the speaker data. The UBM is a GMM trained from a large speaker population. It represents a speaker-independent distribution of the feature vectors and it is used as the alternative hypothesis in an ASV system [25]. We trained the UBM from a speaker population that is distinct from the evaluation speaker set, as is a common practice in ASV studies. For each speaker in the evaluation set, the GMM was generated by adapting the UBM to the speaker data. In all the experiments reported in this thesis, we used the MSR Identity Toolbox [27] implementation of GMM-UBM.

The ASV experiments were done on the speech data obtained from the popular TIMIT corpus [14]. This corpus contains recordings from 630 American English speakers with different dialects. For each speaker, TIMIT contains ten recorded utterances of different sentences. Two of the sentences (denoted by SA1 and SA2 in the original TIMIT corpus) were spoken by all the 630 speakers while the remaining eight sentences were chosen randomly from a larger set of sentences. The speech data in TIMIT is recorded in a quiet room (noise-free) with a high quality microphone [29, p. 14] and it is sampled at 16 kHz.

In our experiment, sixty percent of the speakers were randomly chosen and assigned to the UBM set while the remaining forty percent were assigned to the evaluation set. In the UBM set, all ten utterances from each speaker were used in generating the UBM. We used a single gender-independent UBM instead of generating UBMs for both genders. In the evaluation set, the utterances SA1 and SA2 were used in the testing of the ASV-system while the remaining eight utterances were used to train the speaker models. In the testing phase, we performed all but the cross-gender trials. That is, the feature vectors extracted from the male speakers were not evaluated against female speaker models and vice versa. Table 2 summarizes the details of the experiment.

We tested LSF track based ASV-systems with various parameter combinations. For

Table 2: Partitioning of the TIMIT data for the ASV experiments.

Experiment details	
Total number of speakers	630
Male / Female speakers	438 / 192
Male / Female speakers in UBM set	263 / 115
Male / Female speakers in evaluation set	175 / 77
Sentences used in UBM	All (1 – 10)
Sentences used in training	3 – 10
Sentences used in testing	1 – 2 (SA1, SA2)
Male vs. Male target / non-target trials	350 / 60900
Female vs. Female target / non-target trials	154 / 11704
Total target / non-target trials	504 / 72604

each system configuration, we calculated the ROCCH-EER (*receiver operating characteristic convex hull - equal error rate*) value introduced by Brümmer [6, p. 85]. The ROCCH-EER value is calculated from ROCCH-DET curve, which is an interpolated version the *detection error tradeoff* (DET) curve [22]. The DET curve is formed by graphing the false positive rates (FPR) against the false negative rates (FNR) for different decision thresholds used in speaker verification. The FPR and FNR are also known as the *false alarm probability* and as the *miss probability*, respectively. The EER is obtained by setting the FPR and FNR equal. We calculated the ROCCH-EER values by using the implementation in the BOSARIS toolkit [1].

6.3 Parameter choices

Next, we study how the different parameter choices affect the LSF track based ASV systems. The parameters of interest are listed in Table 3. It is challenging to to evaluate all the possible combinations of the presented five parameters even if we would use large steps in the parameter values. Therefore, we make certain constraints to the parameters based on what seems to be sensible.

First of all, we fix the number of Gaussian components to 128. In early experiments this value produced good results for various parameter combinations tested and, additionally, it allows relatively fast processing of the tests. Then, we fix the ratios between the feature vector dimensionality and the LSF frame length (d/f) and between the LSF frame length and the number of parameters in LSF track modeling (f/r). The motivation to fix these ratios is as follows. When the LSF frame length is increased, it

Table 3: Notations for the parameters in ASV systems.

Parameter	**Notation**
The number of Gaussian components (mixtures) in the GMM	g
The LP model order	p
The number of parameters in LSF track modeling	r
The length of LSF frames given as the number of consecutive short-term frames in LSF frames	f
Feature vector dimensionality after the final DCT step (the number of retained DCT coefficients)	d

is reasonable to expect that we need more dimensions to obtain representative feature vectors. Similarly, the longer the LSF frame, the more we need LSF track parameters to model the tracks with high enough precision. We test all the combinations of ratios $d/f = 4, 5, 6, 7$ and $f/r = 2, 3$.

When the ratios d/f and f/r are set, we implicitly set the values for d and r by setting the LSF frame length f. We tested the LSF frame lengths $f = 6, 8, 10, 12, 14, 16$. If $f = 6$, the LSF frame spans over 100 ms and if $f = 16$, the LSF frame is 250 ms long. Now, we have only one free parameter remaining, the LP model order (p). We tested the values $p = 20, 25, 30, 35$.

Table 4 shows the results for ASV systems where the LSF tracks are modeled with the DCT. We observe the following. Firstly, in most cases, the LP model order (p) needs to be higher than 20 to obtain good results. Next, we find that, on average, small LSF frame lengths (f) perform better. However, since f is tied to d and r, we cannot make any definitive conclusions about the preferable LSF frame lengths. And finally, what comes to the ratios d/f and f/r, it is hard to find any obvious patterns between different combinations. The minimum EER value when $d/f = 4$ is 0.34%, which is higher than with the greater d/f values. We also tested the ratio $d/f = 3$ (not shown in the table), which gave the minimum EER of 0.39%. So, it seems that d/f should be at least 4.

We present the results for systems using the DCT-FDLP LSF track modeling method in Table 5. The DCT-FDLP method does not work as well as the DCT method with higher LSF frame lengths. Although both methods yield similar results when fitting estimates of the LSF tracks, the parameter representation of the DCT-FDLP method seems to be less compatible with the presented ASV system. For small frame lengths (and thus for small numbers of LSF track parameters), we obtain similar results to the DCT methods.

Table 4: Verification results for the DCT based LSF track features ($g = 128$). The best result in each row and column is highlighted with bold text and background color, respectively.

$d/f = 4, \quad f/r = 2$				
	EER (%)			
f	$p = 20$	$p = 25$	$p = 30$	$p = 35$
6	0.57	**0.38**	0.49	0.42
8	0.65	0.48	**0.45**	0.46
10	0.54	0.51	0.48	**0.39**
12	0.85	**0.38**	0.46	0.55
14	0.77	0.61	**0.47**	**0.47**
16	0.90	0.60	**0.43**	0.46

$d/f = 4, \quad f/r = 3$			
EER (%)			
$p = 20$	$p = 25$	$p = 30$	$p = 35$
0.66	0.39	0.50	**0.34**
0.64	0.45	**0.44**	0.46
0.74	0.69	**0.46**	0.58
0.90	0.45	**0.41**	0.55
1.01	0.66	**0.49**	0.50
1.38	0.89	0.66	**0.47**

$d/f = 5, \quad f/r = 2$				
	EER (%)			
f	$p = 20$	$p = 25$	$p = 30$	$p = 35$
6	0.51	0.43	**0.31**	0.34
8	0.52	0.50	0.52	**0.47**
10	0.69	**0.41**	0.43	0.52
12	0.85	0.61	**0.32**	0.40
14	0.73	0.53	0.53	**0.36**
16	1.06	0.59	0.52	**0.48**

$d/f = 5, \quad f/r = 3$			
EER (%)			
$p = 20$	$p = 25$	$p = 30$	$p = 35$
0.65	0.40	0.44	**0.31**
0.63	0.53	**0.46**	0.51
0.81	0.52	**0.31**	0.47
0.96	0.82	0.53	**0.42**
1.13	0.75	0.62	**0.59**
1.40	0.80	**0.70**	0.79

$d/f = 6, \quad f/r = 2$				
	EER (%)			
f	$p = 20$	$p = 25$	$p = 30$	$p = 35$
6	0.51	**0.34**	0.48	**0.34**
8	0.55	0.39	**0.38**	0.45
10	0.62	0.49	**0.29**	0.47
12	0.73	0.49	**0.36**	0.41
14	0.99	0.63	0.65	**0.55**
16	1.19	0.70	**0.58**	0.61

$d/f = 6, \quad f/r = 3$			
EER (%)			
$p = 20$	$p = 25$	$p = 30$	$p = 35$
0.66	**0.38**	**0.38**	0.42
0.64	0.49	**0.43**	0.61
0.88	0.64	0.53	**0.45**
1.15	0.91	0.73	**0.64**
1.30	0.70	**0.59**	**0.59**
1.61	1.01	**0.68**	0.71

$d/f = 7, \quad f/r = 2$				
	EER (%)			
f	$p = 20$	$p = 25$	$p = 30$	$p = 35$
6	0.33	**0.29**	0.46	0.34
8	0.61	**0.40**	0.41	0.42
10	0.57	0.53	0.38	**0.37**
12	0.78	0.51	0.47	**0.37**
14	1.10	0.56	0.50	**0.49**
16	1.33	0.87	**0.55**	0.57

$d/f = 7, \quad f/r = 3$			
EER (%)			
$p = 20$	$p = 25$	$p = 30$	$p = 35$
–*	0.41	0.42	**0.34**
0.56	0.47	0.44	**0.38**
–*	0.70	0.56	**0.37**
–*	**0.59**	0.83	0.83
1.64	0.97	0.71	**0.56**
–*	1.50	0.79	**0.78**

*Invalid parameter configuration ($d > pr$).

Table 5: Verification results for the DCT-FDLP based LSF track features ($g = 128$). The best result in each row and column is highlighted with bold text and background color, respectively.

$\boldsymbol{d/f = 4, \quad f/r = 2}$				
	EER (%)			
f	$p = 20$	$p = 25$	$p = 30$	$p = 35$
6	0.53	0.46	**0.40**	0.43
8	0.92	0.73	0.53	**0.38**
10	0.99	0.87	0.76	**0.67**
12	0.54	1.14	1.11	**0.78**

$\boldsymbol{d/f = 4, \quad f/r = 3}$			
EER (%)			
$p = 20$	$p = 25$	$p = 30$	$p = 35$
0.45	0.50	0.40	**0.35**
0.85	0.49	0.35	**0.33**
0.82	0.54	0.48	**0.41**
1.40	0.93	1.07	**0.82**

$\boldsymbol{d/f = 5, \quad f/r = 2}$				
	EER (%)			
f	$p = 20$	$p = 25$	$p = 30$	$p = 35$
6	0.69	0.48	0.36	**0.30**
8	0.92	0.70	0.63	**0.42**
10	1.18	**0.72**	0.81	0.77
12	1.19	1.10	**0.89**	0.94

$\boldsymbol{d/f = 5, \quad f/r = 3}$			
EER (%)			
$p = 20$	$p = 25$	$p = 30$	$p = 35$
0.61	0.42	0.39	**0.30**
0.60	0.52	0.58	**0.38**
1.01	0.59	0.63	**0.49**
1.45	1.12	**0.71**	0.77

$\boldsymbol{d/f = 6, \quad f/r = 2}$				
	EER (%)			
f	$p = 20$	$p = 25$	$p = 30$	$p = 35$
6	0.72	0.40	**0.35**	0.40
8	0.93	0.74	0.63	**0.55**
10	1.04	0.83	**0.69**	0.77
12	1.11	1.16	**0.78**	0.95

$\boldsymbol{d/f = 6, \quad f/r = 3}$			
EER (%)			
$p = 20$	$p = 25$	$p = 30$	$p = 35$
0.53	0.47	0.40	**0.31**
0.67	0.58	0.49	**0.36**
0.92	0.74	0.55	**0.51**
1.14	1.03	**0.88**	0.89

$\boldsymbol{d/f = 7, \quad f/r = 2}$				
	EER (%)			
f	$p = 20$	$p = 25$	$p = 30$	$p = 35$
6	0.48	**0.34**	0.37	0.43
8	0.90	0.68	**0.57**	0.69
10	1.12	0.96	**0.59**	0.71
12	1.29	0.99	**0.66**	0.79

$\boldsymbol{d/f = 7, \quad f/r = 3}$			
EER (%)			
$p = 20$	$p = 25$	$p = 30$	$p = 35$
–*	0.49	0.45	**0.29**
0.53	0.47	0.52	**0.43**
–*	0.60	**0.57**	0.61
–*	0.93	**0.79**	0.80

*Invalid parameter configuration ($d > pr$).

Next, we chose some of the promising parameter combinations from Tables 4 and 5 to re-evaluate them with a different numbers of Gaussians. The results are presented in Table 6. Decreasing the number of Gaussians down to 64 degrades performance, while increasing it to 512 or to 1024 yields improvements in a few cases.

Table 6: Verification results for different numbers of Gaussian components.

					EER (%)				
	d/f	f/r	f	p	$g = 64$	$g = 128$	$g = 256$	$g = 512$	$g = 1024$
DCT	5	2	6	30	0.53	**0.31**	0.47	0.51	0.65
	5	2	12	30	0.45	0.32	0.34	**0.30**	0.35
	5	2	14	35	0.49	0.36	0.43	**0.28**	**0.28**
	5	3	6	35	0.42	**0.31**	0.38	0.38	0.39
	5	3	10	30	0.50	**0.31**	0.39	0.37	0.40
	6	2	10	30	0.52	0.29	0.31	**0.25**	0.28
	5	3	6	35	0.43	**0.29**	0.34	0.41	0.49
DCT-FDLP	4	3	8	35	0.46	**0.33**	0.39	0.46	0.55
	5	2	6	35	0.36	**0.30**	0.34	0.36	0.37
	5	3	6	35	0.46	**0.30**	0.38	0.38	0.48
	6	3	6	35	0.37	**0.31**	0.36	0.32	0.41

6.4 Baseline short-term features

In the previous section, we discovered that the LSF track based features worked well even when the LSF frame length was low. Therefore the question arises, if we could use LSFs of a short-term frame as a feature vector, without any temporal envelope modeling. And indeed, we found that this simple construction performs slightly better than the LSF track features, as Table 7 suggests.

Table 7: Verification results for the short-term LSF features.

	EER (%)			
	$g = 128$	$g = 256$	$g = 512$	$g = 1024$
LSF ($p = 20$)	0.40	**0.26**	0.34	0.47
LSF ($p = 25$)	**0.29**	0.30	0.31	0.38
LSF ($p = 30$)	**0.26**	**0.26**	**0.26**	0.30
LSF ($p = 35$)	0.32	0.33	**0.20**	0.26
LSF ($p = 40$)	0.24	**0.22**	0.27	0.26
LSF ($p = 45$)	0.32	0.31	**0.22**	0.23

For comparative purposes, and as a reference baseline for the LSF-based features, we also included the most popular speech processing features, *mel frequency cepstral co-*

efficients (MFCCs), in our ASV experiments. The MFCCs are calculated from the short-term frames, and they provide a relatively simple way to obtain good results in speaker recognition [10, p. 295] [17]. We used the MFCC implementation [7] by Ellis. Table 8 shows the settings used in the experiments.

Table 8: The MFCC settings used in the experiments.

MFCC settings	
Frame length	25 ms
Frame overlap	10 ms
Window function	Hanning
Pre-emphasis coefficient	0.97
Liftering exponent	0.6
Min. frequency in mel filtering	0
Max. frequency in mel filtering	8 kHz (Sampling rate / 2)

Table 9: Verification results for the MFCC features.

		EER (%)			
	c	$g = 128$	$g = 256$	$g = 512$	$g = 1024$
MFCC	20	0.44	0.45	**0.31**	0.40
	25	0.35	0.31	**0.26**	0.31
	30	0.23	0.23	**0.20**	0.26
	35	0.23	0.20	0.20	**0.19**
	40	0.28	0.20	0.20	**0.19**
MFCC + Δ	20	0.74	0.67	0.55	**0.50**
	25	0.55	0.40	0.39	**0.34**
	30	0.44	0.33	0.33	**0.30**
	35	0.46	0.47	0.49	**0.31**
	40	0.38	**0.35**	0.41	0.39
MFCC + Δ + $\Delta\Delta$	20	0.74	0.57	0.61	**0.47**
	25	0.48	0.55	**0.38**	0.41
	30	0.36	**0.33**	0.44	0.42
	35	**0.37**	0.47	0.62	0.67
	40	**0.33**	0.35	0.62	0.71

To add some dynamic information about speech to the MFCCs, we tried appending *delta* (Δ) (the first order time derivatives) and *double-delta* ($\Delta\Delta$) (the second order time derivatives) coefficients [23, 8] to the base coefficients. We also tested MFCC features with varying numbers of cepstral coefficients ($c = 20, 25, 30, 35, 40$). The results for the tested MFCC configurations are given in Table 9. The preferred number

of cepstral coefficients is higher than 25 and adding the delta and double-delta coefficients degrades performance. In many applications, delta coefficients have proved to be useful and a lower number of cepstral coefficients is usually preferred. We believe that our differing findings are due to the good quality, noise-free speech data.

6.5 Comparison of ASV systems

We collected the best variants of each ASV system into Table 10. This table has two variants of the DCT method because we also wanted to include a system that uses a relatively long LSF frame length ($f = 14$). The comparison of the systems reveals that the short-term features perform better than the LSF track based features. The MFCC features have a slight advantage over the short-term LSF features and, in the case of long-term features, the DCT variant outperforms the DCT-FDLP variant.

Table 10: Verification results for different types of ASV systems and for their fusions.

Feature type		g	d/f	f/r	f	p	**EER (%)**
MFCC ($c = 35$)	**(A)**	1024	-	-	-	-	0.19
LSF	**(B)**	512	-	-	-	35	0.20
LSF tracks (DCT)	**(C)**	512	6	2	10	30	0.25
LSF tracks (DCT)	**(D)**	512	5	2	14	35	0.28
LSF tracks (DCT-FDLP)	**(E)**	128	5	2	6	35	0.30

Fusion	**Best individual**	**EER (%)**	**Relative change (%)**
A + B	0.19	0.16	-15.9
A + C	0.19	**0.12**	**-36.0**
A + D	0.19	0.16	-16.1
A + E	0.19	0.16	-18.1
B + C	0.20	0.16	-19.7
B + D	0.20	0.17	-14.9
B + E	0.20	0.18	-9.6
A + B + C	0.19	**0.11**	**-43.6**
A + B + C + D	0.19	0.13	-33.6

As the final experiment, we fused different ASV systems with each other. This was done by calculating the mean value of the verification scores of the fused systems. It turned out that the fusions of every combination of systems we tried performed better than the fused systems alone. The best result (EER $= 0.11\%$) was obtained by fusing the MFCCs, the short-term LSF features, and the long-term LSF features together (**A + B + C**). The MFCCs and the DCT-based long-term LSF features (**A + C**) form the

best performing pair of systems. Fusing MFCCs and short-term LSFs (**A** + **B**) did not yield as good improvements, even if the short-term LSF features are the second best performing feature variant. The short-term features are too similar to each other to provide a large benefit when fused.

The detection error tradeoff curves (Figure 29) assure us about the performance differences between the different ASV systems. In the case of MFCCs, one target trial resulted in a substantially lower score than other target trials, which explains the flat part in the DET curve at FNR level 0.2%.

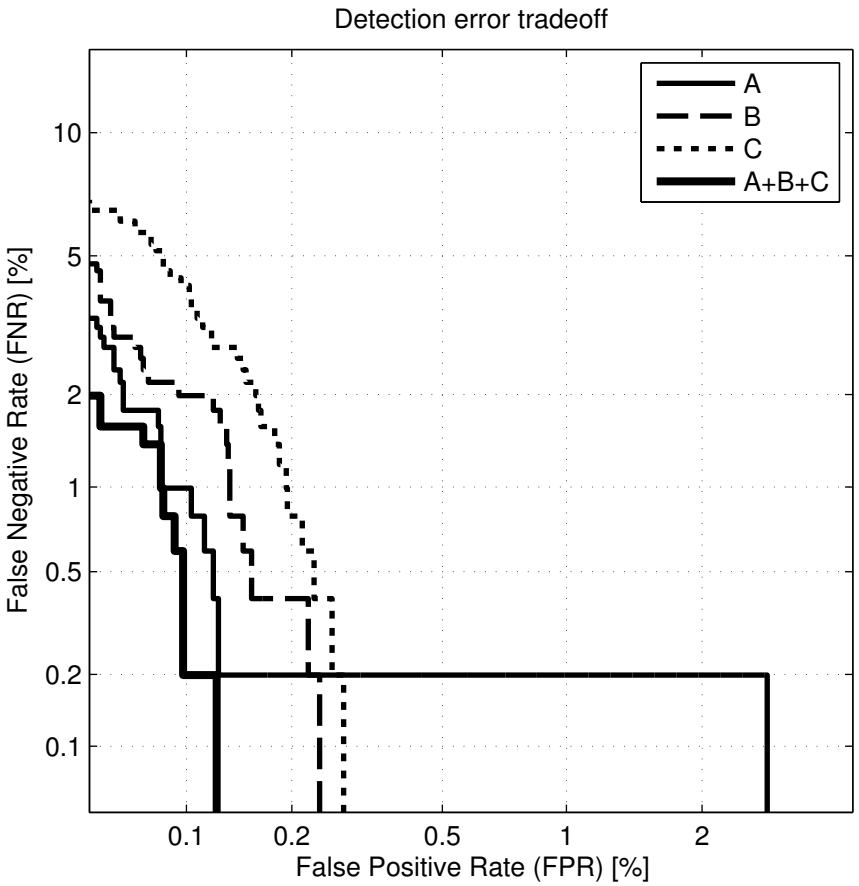

Figure 29: Detection error tradeoffs for MFCCs (A), short-term LSFs (B), long-term LSFs (C), and for their fusion.

7 Conclusions

In this thesis, we reviewed and studied various signal modeling and processing methods. Our main focus was on two LP-related topics: LSFs and FDLP. Finally, we combined most of the studied techniques and created feature vectors for a text-independent speaker verification task.

In Chapter 3, we presented a slightly different derivation for the complex valued LPCs than in [26] and [11]. The complex valued LPCs found their use in our new FDLP proposal, IDFT-FDLP, although we then found out that the real valued linear prediction would be sufficient, if IDFT-FDLP is applied to a symmetrized signal. We tested different FDLP methods in Chapter 5, where we applied them to model LSF tracks. These experiments suggested that, depending on the case, IDFT-FDLP performs similarly or worse than DCT-FDLP. Therefore the usefulness of IDFT-FDLP is questionable.

We also tried LSF track parametrization with the DCT. The DCT method is simple and seems to work for LSF track modeling. One interesting observation related to the DCT and FDLP methods is that the FDLP estimates start to resemble the DCT estimates when the modeled signal is shifted upwards. We believe that this phenomenon might be theoretically provable.

The use of the long-term LSF track based features in the speaker verification task showed promising results. We managed to improve the EER of the MFCC based ASV system from 0.19% down to 0.11% by fusing MFCC-system with systems using short- and long-term LSF features. The system using the LSF track features performed quite well as a standalone system as well (EER = 0.25%).

Because the feature extraction process consists of multiple subsequent phases and techniques, there is still room for the further development of the proposed ASV-systems. One might find improvements in the results just by adjusting the parameters that are related to the presented feature extraction process. Improvements could also be found by trying different LSF track modeling methods or different feature vector post-processing methods. Finally, further research is also needed about the robustness of the methods in less idealized conditions. The experiments of this thesis were purposefully constrained to the laboratory-quality TIMIT corpus to keep all the experimental conditions carefully controlled, but the methods should be further re-evaluated on evaluation corpora containing, for instance, intersession variation and additive noise.

References

[1] The BOSARIS Toolkit v1.06. `https://sites.google.com/site/bosaristoolkit/home`, 2012.

[2] M. Athineos and D.P.W. Ellis. Frequency-domain Linear Prediction For Temporal Features. *Workshop on Automatic Speech Recognition and Understanding*, 2003.

[3] M. Athineos and D.P.W. Ellis. Autoregressive Modeling of Temporal Envelopes. *IEEE Transactions on Signal Processing*, 55(11), November 2007.

[4] R.P. Brent. Old And New Algorithms For Toeplitz Systems. *Proceedings of SPIE*, 975, 1989.

[5] V. Britanak, P.C. Yip, and K.R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Academic Press, 2007.

[6] N. Brümmer. *Measuring, refining and calibrating speaker and language information extracted from speech*. PhD thesis, University of Stellenbosch, 2010.

[7] D.P.W. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab. `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`, 2005.

[8] S. Furui. Speaker-independent isolated word recognition based on emphasized spectral dynamics. *ICASSP*, pages 1991–1994, April 1986.

[9] S. Ganapathy, S.H. Mallidi, and Hermansky H. Robust Feature Extraction Using Modulation Filtering of Autoregressive Models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(8), August 2014.

[10] B. Gold and N. Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. John Wiley & Sons, Inc., 1999.

[11] X. Gu and J. Jiang. A complex autoregressive model and application to monthly temperature forecasts. *Annales Geophysicae*, 23:3229–3235, November 2005.

[12] F.J. Harris. On the use of windows for harmonic analysis with the discrete Fourier transform . *Proceedings of the IEEE*, 66(1), 1978.

[13] F. Itakura. Line spectrum representation of linear predictor coefficients of speech signals. *The Journal of the Acoustical Society of America*, 57, 1975.

[14] Garofolo J., Lamel L., Fisher W., Fiscus J., Pallett D., Dahlgren N., and Zue V. Timit acoustic-phonetic continuous speech corpus. Linguistic Data Consortium. `https://catalog.ldc.upenn.edu/LDC93S1`, 1993.

[15] P. Kabal and R.P. Ramachandran. The computation of line spectral frequencies using Chebyshev polynomials. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34(6), December 1986.

[16] T. Kinnunen, Wei C., Koh E., Wang L., Li H., and E.S. Chng. Temporal discrete cosine transform: Towards longer term temporal features for speaker verification. *Proc. 5th International Symposium on Chinese Spoken Language Processing (ISCSLP 2006)*, pages 547–558, December 2006.

[17] T. Kinnunen and H. Li. An overview of text-independent speaker recognition: From features to supervectors. *Speech Communication*, 52(1):12–40, January 2010.

[18] T.W. Körner. *Fourier analysis*. Cambridge University Press, 1989.

[19] J. Laroche. Autocorrelation method for high-quality time/pitch-scaling. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, October 1993.

[20] J. Makhoul. Linear Prediction: A Tutorial Review. *Proceedings of the IEEE*, 63(4), April 1975.

[21] J.D. Markel and A.H. Gray, Jr. *Linear Prediction of Speech*. Springer, 1982.

[22] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. In *in Proc. of EUROSPEECH*, pages 1895–1898, 1997.

[23] S. Memon, M. Lech, and N. Maddage. Speaker Verification Based on Different Vector Quantization Techniques with Gaussian Mixture Models. *Third International Conference on Network and System Security*, pages 403–408, October 2009.

[24] D.G. Messerschmitt. Stationary points of a real-valued function of a complex variable. http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-93.html, Jun 2006.

[25] D.A. Reynolds, Quatieri T.F., and Dunn R.B. Speaker verification using Adapted Gaussian mixture models. *Digital Signal Processing*, 10(1–3):19–41, January 2000.

[26] I. Sekita, T. Kurita, and N. Otsu. Complex autoregressive model for shape recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(4):489–496, Apr 1992.

[27] Sadjadi S.O., Slaney M., and Heck L. MSR Identity Toolbox v1.0: A MATLAB Toolbox for Speaker-Recognition Research. http://research.microsoft.com/apps/pubs/default.aspx?id=205119, November 2013.

[28] F.K. Soong and B. Juang. Line spectrum pair (LSP) and speech data compression. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1984.

[29] Y. Stylianou, M. Faundez-Zanuy, and A. Esposito. *Progress in Nonlinear Speech Processing*. Springer, 2007.

[30] F. Zheng, Z. Song, L. Li, W. Yu, F. Zheng, and W. Wu. The distance measure for line spectrum pairs applied to speech recognition. In *Proc. ICSLP*, pages 1123–1126, 1998.