2007-01-01

Channel Modeling Applied to Robust Automatic Speech Recognition

Alexander Gabriel Sklar
University of Miami, asklar@gmail.com

UNIVERSITY OF MIAMI

CHANNEL MODELING APPLIED TO ROBUST AUTOMATIC SPEECH RECOGNITION

By

Alexander G. Sklar

A THESIS

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Science

Coral Gables, Florida

December 2007

UNIVERSITY OF MIAMI

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

CHANNEL MODELING APPLIED TO ROBUST AUTOMATIC SPEECH RECOGNITION

Alexander G. Sklar

Approved:

Dr. Michael Scordilis Research Associate Professor of Electrical Engineering	Dr. Terri A. Scandura Dean of the Graduate School
Dr. Xiaodong Cai Assistant Professor of Electrical Engineering	Dr. Subramanian Ramakrishnan Associate Professor of Mathematics

SKLAR, ALEXANDER GABRIEL (M.S., Electrical & Computer Engineering)
Channel Modeling Applied to (December 2007)
Robust Automatic Speech Recognition

Abstract of a thesis at the University of Miami.

Thesis supervised by Professor Michael Scordilis.
No. of pages in text. (176)

In automatic speech recognition systems (ASRs), training is a critical phase to the system's success. Communication media, either analog (such as analog landline phones) or digital (VoIP) distort the speaker's speech signal often in very complex ways: linear distortion occurs in all channels, either in the magnitude or phase spectrum. Non-linear but time-invariant distortion will always appear in all real systems. In digital systems we also have network effects which will produce packet losses and delays and repeated packets. Finally, one cannot really assert what path a signal will take, and so having error or distortion in between is almost a certainty. The channel introduces an acoustical mismatch between the speaker's signal and the trained data in the ASR, which results in poor recognition performance. The approach so far, has been to try to undo the havoc produced by the channels, i.e. compensate for the channel's behavior. In this thesis, we try to characterize the effects of different transmission media and use that as an inexpensive and repeatable way to train ASR systems.

Dedication

This thesis is dedicated to my parents, Miriam and Julio, for their endless love and support.

Acknowledgments

I would like to thank several people without whom I could not have succeeded. My family and friends for their support; Dr. Mykel Billups for making it possible for me to attend the University through an assistantship at the Academic Resource Center. Also, Professor Reuven Lask and Guy Ravitz for referring me for that position. Special thanks go to my advisor Dr. Michael Scordilis for his encouragement and guidance. Also, thanks to Dr. Kamal Premaratne and Dr. Paul Mermelstein for their insights. Last but not least, I would like to thank IBM for their support by providing funding and equipment for the fruition of this work.

Contents

List of Tables

List of Figures

List of Algorithms

Nomenclature

$x[n]$	The n^{th} element of a vector or signal x
$\mathbf{FFT}\{x[n]\}[k]$	The Fast Fourier Transform of signal x evaluated at bin k
$\mathbf{IFFT}\{X[k]\}[n]$	The Inverse Fast Fourier Transform of X evaluated at sample n
$\mathbf{KLT}\{x[n]\}[k]$	The Karhunen-Loève transform of signal x evaluated at element k
$\mathbf{STFT}_N\{x[m]\}[p]$	The N-point STFT of the signal x from $n = m - N + 1$ to $n = m$ evaluated at frequency bin p
$X = C_1 \underset{B}{\overset{A}{\gtrless}} C_2$	Decision operator. If $C_1 > C_2$, $X = A$. Otherwise, $X = B$.
j	The imaginary unit ($j^2 = -1$)
$\mathcal{O}(f(x))$	Order of magnitude (Big-O)
$\Re\{x\}$	Real part of x
$\Im\{x\}$	Imaginary part of x
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^{\mathbb{N}}$	The set of all causal real sequences
$\mathbb{C}$	The set of complex numbers
$\overline{x}$	Complex conjugate of x
$\langle u, v \rangle$	Internal (dot) product of vectors u and v
A^T	Matrix transpose
A^H	Hermitian transpose ($A^H = \overline{A^T}$)
$\lvert\cdot\rvert$	Absolute value ($\lvert x\rvert = \sqrt{\overline{x}x}$)
$\angle z$, $\arg z$	Angle of the complex number z
$\arg\max_x f(x)$	The argument x that maximizes $f(x)$
$\max f$	Maximum of f
$\min f$	Minimum of f
$\lVert\cdot\rVert_p$	p-norm. If p is omitted, the 2-norm ($\lVert x\rVert = \sqrt{x^H x}$).
W_N	The N^{th} root of unity $\left(W_N = e^{-j\frac{2\pi}{N}}\right)$
$\delta(t)$	Dirac delta
$\delta[n]$	Discrete impulse (Kronecker Delta)
$A \backslash B$	The difference set of B and A equal to $A \cap B'$
$\mathbb{E}\{\cdot\}$	The statistical expectation operator

Chapter 1

Introduction

1.1 Background

This thesis is about System Identification and Channel Modeling techniques applied to the problem of Automatic Speech Recognition engine re-training. The most tedious part of setting up an Automatic Speech Recognition (ASR) system is the data acquisition phase. This is the phase where a very large number of signals are to be acquired so that the system can later be trained based on this data.

1.1.1 Signal Acquisition

Signal acquisition is a very cumbersome process. On one hand, setting up the environment can be a difficult task in itself. This, depending on the requirements, includes first coming up with a list of phrases that span the domain of interest. These cannot be just any phrases but must follow the right distribution of phonemes as well as have a significant number of words and phrases that are certain to occur in the system in production.

1.1.1.1 Interactive Voice Response (IVR) Systems

IVR systems are used for customer service scenarios. The user is given a number of options in the form of a spoken menu (i.e. a prerecorded message), and then he is prompted to select an option by either pressing a number key on his phone, or by speaking the number. In some cases, he may be required to answer "yes" or "no" to the questions. In most modern IVR systems, a user will most likely use his voice to interact rather than his phone's dialpad, and the system has to be able to be fed a number of possible answers for a given question. Many of these possible answers will have the same meaning, but different phrasing. Take for example, the prompt "Do you want to pay now?". A user might answer:

- "Yes"

- “No”
- “Yeah”
- “Nope”
- “Nah”
- “No, thanks”
- “Thanks, but no, thanks”
- “Umm...okay”

and so on. We can clearly distinguish there are two basic answers, “Yes” and “No”, and variations thereof. These variations can be small from the phonetic point of view (as in “Yeah” for “Yes”) or can include redundant information which complicates the task of the ASR system, since it has to “scan” for the answer (as in “Thanks, but no, thanks” or “Umm...okay”). It is common for IVR systems to ask the user to confirm what the system believes he said. This is because the ASR system behind the IVR performs something analogous to a statistical hypothesis test of the following form:

$$\begin{cases} H_0^1) & x \in S_1 \\ H_1^1) & x \notin S_1 \end{cases}$$

and

$$\begin{cases} H_0^2) & x \in S_2 \\ H_1^2) & x \notin S_2 \end{cases}$$

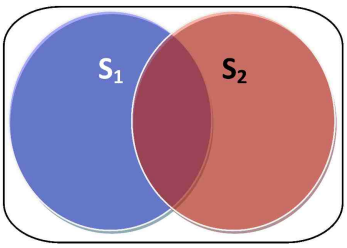


Figure 1.1: Venn diagram of the Hypothesis Test’s sets for the ASR system

In this case, we could identify S_1 and S_2 as the sets of all acceptable phrases whose meanings are “Yes” and “No” respectively. Clearly, if an answer belongs to S_1, it should

not belong to S_2 and vice versa. However, an answer can pass both hypothesis tests so that x cannot be determined to exclusively lie in either S_1 or S_2. Furthermore, an utterance can fail both tests so that x cannot be determined to lie in neither S_1 or S_2. In these cases the system will prompt the user to repeat or restate his answer. Furthermore, the hypothesis test will render a p-value associated with the chosen hypothesis. If this p-value is not large enough, we cannot assert with a large enough degree of certainty that the signal belongs to that set. In that case, the system will prompt the user for confirmation (i.e. it will ask "Did you say "yes"?").

The hypothesis test outlined above is for a two-word vocabulary, which is of not much interest. In general, the test may be stated as the maximization problem

$$S(x) = \arg\max_S \frac{P(x \in S)}{P(x \notin S)}$$

or equivalently[1]

$$S(x) = \arg\max_S \ln\left(\frac{P(x \in S)}{P(x \notin S)}\right)$$

This is called the log-likelihood maximization problem. We see that in the two-word vocabulary case, this becomes

$$S(x) = \frac{P(x \in S_1)}{P(x \in S_2 \backslash S_1)} \underset{S_2}{\overset{S_1}{\gtrless}} \frac{P(x \in S_2)}{P(x \in S_1 \backslash S_2)} \tag{1.1}$$

In the case where the two sets are disjoint, equation 1.1 reduces to the trivial maximization problem

$$\begin{aligned} S(x) &= P(x \in S_1) \underset{S_2}{\overset{S_1}{\gtrless}} P(x \in S_2) \\ &= \arg\max_{S_i} P(x \in S_i) \end{aligned}$$

1.1.2 Recording Environment

Once the phrase list is compiled, one has to set up the recording environment. In the simplest case this consists of a computer recording through a sound card and a computer program that prompts for a phrase and records the speaker's utterances. There is a small subtlety here which could make or break the ASR, and this is the recording conditions. If

[1] since $\ln x$ is a monotonous non-decreasing function

one was to record the utterances in ideal conditions, that is, infinite signal-to-noise (SNR) ratio, the resulting signal statistical properties would be different than those of a signal with broadband noise (thermal noise), narrowband noise (50 or 60 Hz AC current EM interference, air conditioning hum), babble, etc. The recording conditions have to match in this sense the conditions under which the user is expected to be. Therefore, as nice as having a clean recording of a phrase is, it is not very useful since there won't be much correlation between the training set and the sentences to be recognized.

As we will show in sections 1.2 and 1.3, the focus until now has been in recording clean speech, and then trying to clean up the user utterances so that they resemble clean speech.

1.1.3 Speech Recognition

An ASR system consists of a front-end, also called feature extractor, which takes the speech waveform and outputs a time series of feature vectors, and a back-end or statistical decoder, which produces a sequence of words from the time series. When we add the speaker and the communication channel to the ASR, we get the system depicted in figure 1.2.

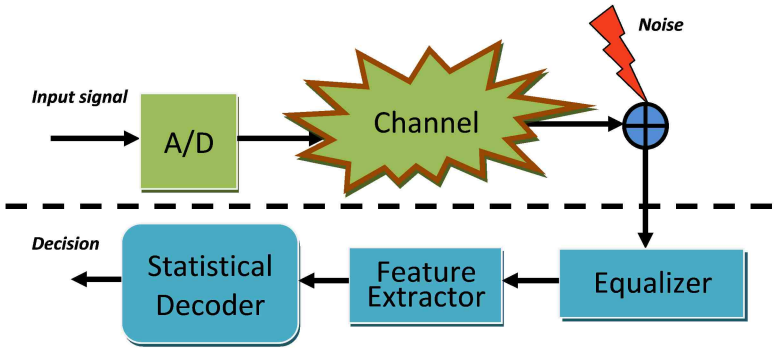


Figure 1.2: High level diagram of a Speech Recognition System

Thus, the speaker encodes a sequence of words $W_1, \ldots, W_N$ as an acoustic pressure wave. This wave is transformed to an electric signal by a microphone, and then is fed through an A/C converter, which outputs a digital waveform. The channel then distorts and filters the signal in some *a priori* unknown way.

1.1.4 The Feature Extractor

The feature extractor processes this signal and outputs sets of "features", i.e. numbers that represent some relevant magnitude for recognition purposes. As we know, speech is only stationary when viewed a few milliseconds at a time, so for the analysis to be meaningful a feature extractor outputs a feature vector every, say, 10 to 30 milliseconds. Also, we want the feature extractor to eliminate any redundant information, that is, the

feature vector elements should be independent of each other or "orthogonal". This can be more or less easily achieved by techniques discussed in sections 2.1.2.2 and 2.1.2.3. Some popular feature choices are described in sections 2.1, 2.2, 2.3 and 2.4.

In addition, we would also like the feature extractor to remove any information that is irrelevant to the task at hand (if we were doing speaker independent recognition, we would like the features to be similar for a given utterance, either spoken by a male or female speaker). As a third requirement, we would like the feature vector to be as simple and compact a representation as possible. This is because we will then have to work with this vector by carrying some sort of search algorithm in an M-dimensional space, M being the length of the feature vector.

All in all, the feature extractor will output a low-dimensionality feature vector $\{O_k\}_{k=1\ldots M}$ every τ milliseconds, with τ either being fixed or varying according to the signal statistics, but always in the 10 to 30 milliseconds order. The reason for this is that the frame cannot be too long, because the signal statistics generally change with time, and it can not be too short, because there would not be enough information contained in the frame so as to provide sensible statistics, i.e. data that is consistent with a longer portion of the signal.

1.1.5 The Statistical Decoder

The sequence of feature vectors is fed to the statistical decoder which will output a sequence of words that will hopefully be the same as the ones the speaker spoke. A canonical statement of the decoding process within the realm of probability is thus

$$W_1 \ldots W_N = \arg \max_{W'_1 \ldots W'_N} P\left(W'_1 \ldots W'_N | O_1 \ldots O_M\right)$$

or

$$\tilde{W} = \arg \max_{W \in W^*} P\left(W | \{O_k\}\right) \tag{1.2}$$

where $W^* = \{W'_1 \ldots W'_N\}$ is the set of all possible word sequences, $\{O_k\}$ is the feature vector and $\tilde{W}$ is the recognized word. Using Bayes' rule, we get

$$\tilde{W} = \arg \max_{W \in W^*} \frac{P\left(\{O_k\} | W\right) P\left(W\right)}{P\left(\{O_k\}\right)}$$

Since the feature vector is independent of the maximization problem, we rewrite the last equation as

$$\tilde{W} = \arg \max_{W \in W^*} P\left(\{O_k\} | W\right) P\left(W\right) \tag{1.3}$$

1.2 Prior Work

To estimate the probability $P(\{O_k\}|W)$ in equation 1.3, Hidden Markov Models (HMM) have been used. The second term $P(W)$ is estimated using what is referred to as a Language Model. A language model is basically a list of valid words in the language and their relative occurrence in normal speech. The language need not be a full language in itself; it can be constrained to a domain. This gives rise to domain-specific speech recognition.

Another way to model the language is by the use of a state machine, through which valid sentences are represented. In this approach, each word represents a node, and each edge represents a non-zero probability of having the originating word before the destination word. This state machine represents thus a language grammar, and greatly simplifies the search space of the words.

1.3 Motivation and Goals

To overcome the disparity in recording conditions, one has to modify the incoming speech to be recognized to match the signal conditions used for training the system. There are two possibilities: either train the system with clean speech, or train the system with "dirty" speech, i.e. a speech signal that has been distorted by the transmission media, gone through some lossy process, and/or a signal to which noise of some kind has been added. The notions of noise and "dirty" signals are deeply related, since one considers noise to be any signal which is not desired for the purpose of the system.

The production conditions will most likely not correspond to clean speech. Thus, if the system is trained on clean speech, the front-end to the system will have to filter the incoming signal to make it look like clean speech. Usually this consists of a channel equalizer and/or a noise reduction system.

A channel equalizer is a filter, which tries to undo the effects the channel had on the signal from a frequency point of view. Put another way, a channel equalizer is a filter whose magnitude response is close to the inverse of the magnitude response of the channel, or equivalently, a filter such that the combined magnitude response of the channel and the equalizer is close to unity. Equalizers can never be perfect, they have several limitations. For one, they are adaptive systems which must be trained to arrive to an optimal set of coefficients. This training has to be done on production-conditions speech. In general, Finite Impulse Response (FIR) equalizers are used for several reasons. First, the available algorithms for their adaptation are well-behaved[2] and well understood. Secondly, FIR filters are always stable, so there is no question as to whether the output of the equalizer is going to be usable or not. Finally, since speech can be modeled to some extent as an Autoregressive

[2]By well-behaved we mean that they can easily be made to converge to the optimal solution.

(AR) process (section 2.3.3), the inverse process is a Moving Average (MA) process which is realized by an FIR filter.

Chapter 2

Preliminary Notions

2.1 Lossless Signal Representation

During the conversion of a signal from an analog signal (continuous in time and range, e.g a sound wave) to a digital signal, two processes take place. First, the time axis is discretized. This process is known as sampling. Basically, a sampling device takes instantaneous values of the signal at usually regular intervals of time. Note that these values can vary continuously. If a signal is to be sampled and then reconstructed (i.e. converted back into an analog signal), then the signal must comply with the Nyquist-Shannon Sampling Theorem: the sampling frequency f_s has to be larger than twice the highest frequency present in the analog signal. Therefore, there will always be an analog lowpass filter before sampling known as an antialiasing filter, so that this condition is assured. Even if the signal to be sampled is known to be bandlimited, noise will always be present in every real situation, and that noise is present at all frequencies. Therefore, not putting an antialiasing filter will cause aliasing of the higher frequencies' noise content into the baseband, producing a distortion which cannot be alleviated.

The second process the signal undergoes is known as quantization. A quantizer is a device that restricts the range of possible values the signal can take to lie within a finite set. Since digital binary computers are the only ones in use in actuality[1], the set of possible values is usually a set of points taken in intervals of a power of two. For example, a two-bit quantizer using values from 0 to $\frac{3}{4}$ would output either 0V, $\frac{1}{4}$V, $\frac{1}{2}$V or $\frac{3}{4}$V. Whenever a signal greater in magnitude than $\frac{3}{4}$V comes, it is clamped at the output at $\frac{3}{4}$V. The behavior for negative voltages is similar; the behavior for values in between two preset values is a priori

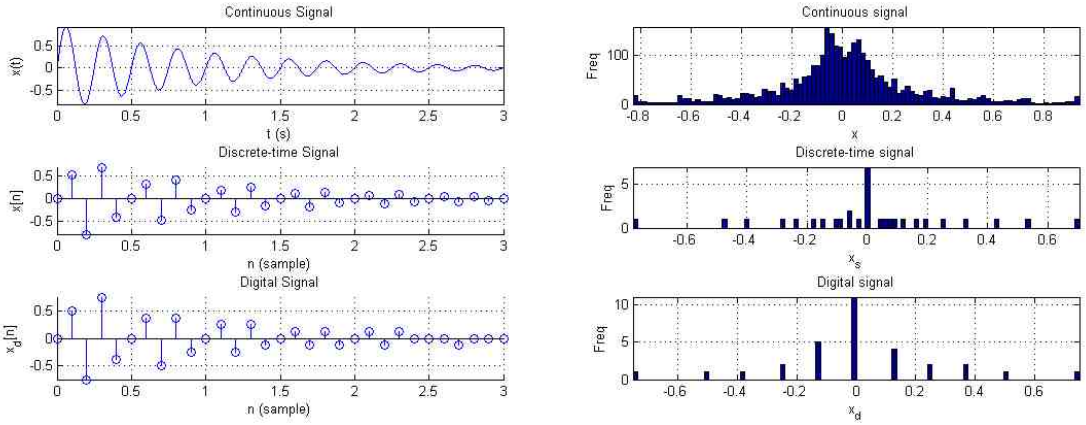
[1]Ternary computers have been known since 1840, when Thomas Fowler built one entirely out of wood. The largest ternary computer was built in the 1950s in the Soviet Union and it was called Setun. Despite the apparent demise of ternary computers, Donald Knuth predicts their comeback in the near future, based on their superior properties compared to their binary counterparts ([10, 119]).

arbitrary, and gives rise to another degree of freedom in designing quantizers. Two popular choices are known as truncation (disregarding the fractional part) and rounding (rounding it towards the closest integer). Following the quantizer, lies yet another device called encoder which as its name indicates, actually encodes the quantizer's output into a binary word. Thus, another degree of freedom is added as to how to assign codewords to values. Popular choices are Sign-Magnitude, which gives rise to Sign-Magnitude Truncation (SMT) and Sign-Magnitude Rounding (SMR), and Two's-complement Truncation (TCT).

We can say, in summation, that digitizing a signal introduces two kinds of errors. First, the signal has to be bandlimited. If the signal is not bandlimited, then the signal will contain the information contained only in the baseband, i.e. some information will be lost. This can be palliated by using a high enough sampling frequency. It is also worth noting that filters, analog or digital, can not have a brick wall characteristic and be realizable at the same time. A lowpass filter introduces attenuation at the high frequencies, starting at the passband frequency, and smoothly reaches a large attenuation at its stopband frequency. Therefore, either the stopband frequency is made to be half the sampling rate, which forces the passband frequency to lie within the baseband, or the passband frequency is chosen to be equal or greater than half the sampling frequency. In the first case, high frequencies in the baseband will be attenuated. This can be somewhat alleviated by using post-emphasis digital filters. In the second case, the actual sampling frequency has to be larger, and then once the signal is in the digital domain, it can be digitally filtered and downsampled to the necessary rate.

The second kind of error is due to the quantization process and is known as quantization noise. Different quantizer/encoder models give rise to different quantization noise statistics. In general, some assumptions are made when modeling this noise:

- The amplitude of the noise is much smaller than that of the signal (conversely, the quantizer almost never saturates)
- The noise is a sample sequence of a stochastic wide-sense stationary process, uncorrelated with the input signal and itself.
- The noise is distributed uniformly over the range of the quantization error.
- The input signal is a sample sequence of a stationary process.



(a) Continuous signal and PCM (b) Histograms

Figure 2.1: Continuous signal, sampled version and digital signal

2.1.1 Time Domain (PCM)

The most standard way to represent a signal without any loss of information (apart from noise introduced in the A/D conversion) is by just using its sample values at regular time intervals (i.e. uniform sampling). The values are encoded usually using a binary code. This is known as the waveform or Pulse Code Modulation.

The main disadvantage of using PCM is that is requires a large storage capacity, since every single value in the signal is represented. Compressed variants of PCM are ubiquitous, both in degrading (lossy) and non-degrading (lossless) versions.

On the other hand, PCM provides an advantage with respect to other representations in that it allows tracking of fast time-domain events such as sudden bursts of energy coming from a plosive, a drum, etc.

2.1.1.1 Differential PCM (DPCM)

In its simplest form, Differential PCM (DPCM, also known as Δ-PCM) will encode the first sample in a data stream and then the differences between consecutive samples. This way, the dynamic range of the resulting signal is much smaller than the original one, allowing for more efficient compression without introducing losses.

A natural extension to this simple approach is the prediction of the next sample's value based on the last few values, thus encoding the error in the prediction as well as the coefficients used for performing the prediction. This is usually achieved using Linear Prediction (see section 2.4).

2.1.1.2 Companding

The problem with directly encoding the amplitude of each sample in the time domain is that the quantization noise maximum magnitude is constant, i.e. independent of the actual sample value. In other words, for a large sample value, the SNR is high, but for sample values close to the quantization floor, the SNR is terrible. Because speech is a process with such a dynamic range, the fine details are lost when quantization is present.

To overcome this problem, one can use what is referred to as a *companding* scheme. Companding entails compressing the dynamic range of the signal so as to obtain a uniform SNR over all of the dynamic range of the signal. In particular, two companding schemes are widely used. The first is called A-law and is primarily used in Europe. The second one is called μ-law and is used in North America and Japan.

For 8-bit encoding and an original signal $x(t) : \mathbb{R}^+ \to [-1, 1]$, μ-law compression is defined as

$$\begin{aligned} F(x) &= \operatorname{sign} x \frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \\ F^{-1}(y) &= \frac{\operatorname{sign} y}{\mu}\left((1+\mu)^{|y|}-1\right) \end{aligned}$$

where $\mu = 255$.

The discrete-time μ-law is defined in the ITU-T Recommendation G.711 ([11]).

Similarly, A-law compression is defined by

$$\begin{aligned} F(x) &= \operatorname{sign} x \begin{cases} \frac{A|x|}{1+\ln A} & |x| < \frac{1}{A} \\ \frac{1+\ln(A|x|)}{1+\ln A} & \frac{1}{A} \le x \le 1 \end{cases} \\ F^{-1}(y) &= \operatorname{sign} y \begin{cases} \frac{|y|(1+\ln A)}{A} & |y| < \frac{1}{1+\ln A} \\ \frac{1}{A}\exp\left(|y|(1+\ln A)-1\right) & \frac{1}{1+\ln A} \le |y| \le 1 \end{cases} \end{aligned}$$

with $A = 87.7$ or $A = 87.6$.

2.1.2 Frequency Domain Representation

Just as time domain representations show how a signal changes in time, a frequency domain representation shows how much signal content lies within a certain frequency band. In this section we present some ways to represent or encode a signal using its frequency domain information.

2.1.2.1 Discrete Fourier Transform (DFT)

The Discrete Fourier Transform is a bijective transformation taking a sequence (signal) of generally complex numbers $x[n]$ and transforming it into a second sequence $X[k]$, which is referred to as the DFT of the first one. The two are said to form a Fourier Transform pair, and noted as $x[n] \leftrightarrow X[k]$, or simply $x \leftrightarrow X$.

$$W_N = e^{-j\frac{2\pi}{N}}$$

$$\begin{aligned} \mathbf{FFT}\{x[n]\}[k] &= X[k] \\ &= \sum_{n=0}^{N-1} x[n] W_N^{nk} \\ \mathbf{IFFT}\{X[k]\}[n] &= x[n] \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk} \end{aligned}$$

The Fast Fourier Transform (FFT) and its inverse, the Inverse Fourier Transform (IFFT), are efficient ways to implement the evaluation of a DFT. Specifically, while direct computation of the DFT requires $\mathcal{O}(N^2)$ operations, the FFT requires $\mathcal{O}(N \log_2 N)$, thus being much faster than direct computation. For a signal 1024 samples long, this represents a reduction in computations by a factor greater than 100.

Since the transformation is invertible, one can always recover the original signal from its DFT. As it is widely known, the DFT is but a sampled version of its continuous version, the Discrete-time Fourier Transform (DTFT):

$$X\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

When the infinite sum converges (either because the signal has finite length or the values of the signal decrease quickly enough), the DTFT is also called the Frequency Response. This is because it conveys information on how a sinusoid of a given frequency ω will be affected when passed through a system with impulse response given by x. Also, the DTFT

is itself a special case of another, more general transform called the z-transform:

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

A very important property of the DFT is Plancherel's Theorem: given two complex-valued discrete-time signals $x[n]$, $y[n]$, and their respective N-point DFTs $X[k]$ and $Y[k]$, it holds that

$$\sum_{n=0}^{N-1} x[n] \overline{y[n]} = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \overline{Y[k]}$$

This might be also stated as an identity in internal (dot) products:

$$\langle x, y \rangle = \frac{1}{N} \langle X, Y \rangle$$

A fundamental corollary is known as Parseval's Identity. When one takes $y = x$, the identity reduces to

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

or equivalently

$$\langle x, x \rangle = \frac{1}{N} \langle X, X \rangle$$

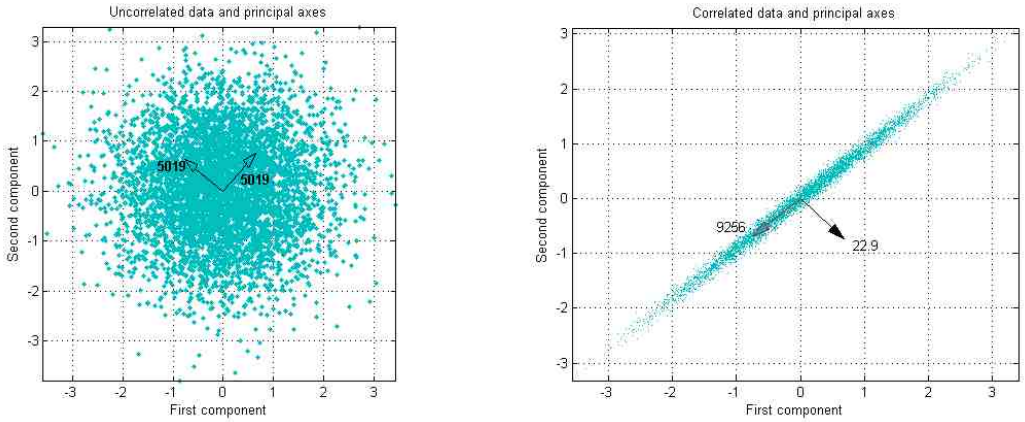
Since the dot product of a vector with itself is nothing but its squared 2-norm, we may also write

$$\|x\| = \frac{1}{\sqrt{N}} \|X\| \tag{2.1}$$

In summation, the Fourier transform preserves the norm of a signal, except for a scaling factor, and changes the cosine of the angles between signals by a factor of $\frac{1}{N}$.

Another important property the DFT has is the shifting property, which can be stated as follows: if $x[n] \leftrightarrow X[k]$, then

1. $\mathbf{FFT}\{x[n] W_N^{nm}\}[k] = X[k+m]$, and
2. $\mathbf{FFT}\{x[n-m]\}[k] = X[k] W_N^{km}$



(a) Uncorrelated data and principal axes

(b) Highly correlated data and principal axes

Figure 2.2: Correlated and uncorrelated data

For more information, see [3, 6].

2.1.2.2 Principal Component Analysis (PCA)

Principal Component Analysis is a tool in multidimensional signal processing that changes the coordinate system so as to eliminate interdependence between dimensions. This interdependence or lack thereof can be shown graphically by plotting dimensions pairwise, if one gets a set of points with no "preferred" direction as in figure 2.2(a), then those dimensions are uncorrelated. If on the other hand, the points more or less align in one direction as in figure 2.2(b), then there is some degree of dependence. The more alignment there is, the stronger the dependence.

PCA is an orthogonal linear transformation. In other words, it transforms the data it operates on, to a new coordinate system, where the individual coordinates have decreasing variances. This is useful in lossy compression, as one may reconstruct the signal to varying degrees of accuracy by keeping only the first L elements. The more elements one keeps, the better the reconstruction will be.

Suppose we have N zero-mean stochastic processes $x_n[m]$, with $n \in \{0, N-1\}$ and $m \in \{0, M-1\}$. We will arrange these values in a $N \times M$ matrix $X[n,m] = x_n[m]$. Then, we form the $N \times N$ covariance matrix C_X as

$$C_X = \frac{1}{N-1} X X^H$$

The covariance matrix contains information relating the variances (autocorrelations) of the N processes (dimensions), as well as the covariances (cross-correlations) between them. We might then calculate the canonical Jordan decomposition of the covariance matrix.

$$\begin{aligned} C_X &= V^H D V \\ V^{-1} &= V^H \\ V\, C_X\, V^H &= D \end{aligned}$$

Because of the special form of the covariance matrix, its eigenvalues will be non-negative, so we can define a square root matrix

$$\begin{aligned} B &= \sqrt{C_X} \\ &\stackrel{\text{def}}{=} V^H \sqrt{D} V \end{aligned}$$

and define a transformed signal

$$\begin{aligned} y &= B^{-1} x \\ &= \left(V^H \sqrt{D} V \right)^{-1} x \\ &= V^{-1} D^{-\frac{1}{2}} V^{-H} x \end{aligned}$$

$$y = V^H D^{-\frac{1}{2}} V\, x \tag{2.2}$$

Now, we show that this new signal, is x after being decorrelated. For that purpose, we calculate y's covariance matrix:

$$\begin{aligned} C_Y &= \frac{1}{N-1} y \cdot y^H \\ &= \frac{1}{N-1} \left(V^H D^{-\frac{1}{2}} V x \right) \left(V^H D^{-\frac{1}{2}} V x \right)^H \\ &= \frac{1}{N-1} V^H D^{-\frac{1}{2}} V x x^H V^H D^{-\frac{1}{2}H} V \\ &= \frac{1}{N-1} V^H D^{-\frac{1}{2}} V\, (N-1)\, C_X V^H D^{-\frac{1}{2}H} V \\ &= V^H D^{-\frac{1}{2}} V C_X V^H D^{-\frac{1}{2}H} V \\ &= V^H D^{-\frac{1}{2}} D D^{-\frac{1}{2}H} V \end{aligned}$$

Now, because D is diagonal and non-negative (hence, real), we have that

$$\begin{aligned} C_Y &= V^H D^{-\frac{1}{2}} D D^{-\frac{1}{2}} V \\ &= V^H V \\ &= I \end{aligned}$$

Therefore, we have shown that the transformation given in equation 2.2, actually decorrelates the input signal, i.e., leaves a signal whose covariance is trivial. This procedure actually removes the interdependence of the different processes, and leaves a new set of independent processes[2]. In light of these observations we may reinterpret equation 2.2: we first take the signal x and transform the coordinate axes to match the principal directions. Then, we scale (i.e. divide) each component by its associated standard deviation[3], and lastly, convert back to the original coordinate system. More information in [10].

2.1.2.3 Karhunen-Loève Transform (KLT)

The Karhunen-Loève Transform, is an orthogonal linear transformation similar to the PCA transformation just described. The only difference is that the resulting covariance matrix turns out to be diagonal instead of being the identity matrix. Thus, different components will have different variances, which is a way of choosing which components to keep in a lossy compression scheme, or also to decide on the number of bits to use when encoding a component.

The transformation used is $y = \mathbf{KLT}\{x\} = V \cdot x$. In that case, we have that

$$\begin{aligned} C_Y &= \frac{1}{N-1} y y^H \\ &= \frac{1}{N-1} V x x^H V^H \\ &= \frac{N-1}{N-1} V C_X V^H \\ &= D \end{aligned}$$

[2] In a way, is similar to finding an orthonormal basis in an N-dimensional space.

[3] In the one-dimensional case, this is equivalent to doing $y = \frac{x}{\sigma}$ where σ^2 is the signal variance.

According to Riesz's Representation Theorem, one can write

$$\begin{aligned} y &= \sum_{j=1}^{N} \langle x, v_j \rangle v_j \\ \tilde{y} &= \sum_{j=1}^{d} \langle x, v_j \rangle v_j \end{aligned}$$

i.e., express the resulting vector y as a linear combination of the orthonormal basis's vectors $\{v_j\}$, and define a truncated version considering only the first d basis vectors. Evidently, there is some error introduced by this truncation. It can be shown that the KLT minimizes the truncation error. This is proven as follows:

$$\begin{aligned} e &= E\left\{\|y-\tilde{y}\|^2\right\} \\ &= E\left\{(y-\tilde{y})^H (y-\tilde{y})\right\} \\ &= E\left\{\left(\sum_{j=d+1}^{N} \langle x, v_j \rangle v_j\right)^H \left(\sum_{j=d+1}^{N} \langle x, v_j \rangle v_j\right)\right\} \\ &= E\left\{\left(\sum_{j=d+1}^{N} \langle x, v_j \rangle^H v_j^H\right)\left(\sum_{j=d+1}^{N} \langle x, v_j \rangle v_j\right)\right\} \\ &= E\left\{\sum_{j=d+1}^{N} x_j^2\right\} \end{aligned}$$

Also,

$$\begin{aligned} V^H y &= x \\ v_j^H y &= x_j \end{aligned}$$

so

$$\begin{aligned} e &= \sum_{j=d+1}^{N} E\left\{v_j^H y y^H v_j\right\} \\ &= \sum_{j=d+1}^{N} v_j^H C_Y v_j \end{aligned}$$

To find the optimal orthonormal basis, we minimize (using Lagrange multipliers):

$$\sum_{j=d+1}^{N} v_j^H C_Y v_j - \sum_{j=d+1}^{N} \lambda_j \left(v_j^T v_j - 1 \right)$$

Setting the derivative to zero,

$$2 \left(C_Y v_j - \lambda v_j \right) = 0$$

Therefore, v_j is an eigenvector of the covariance matrix with associated eigenvalue λ_j. Also, the KLT has the property that it minimizes representation entropy, defined as

$$\begin{aligned} H &= -\sum_{j=1}^{d} \hat{\lambda_j} \log_2 \hat{\lambda_j} \\ \hat{\lambda_j} &= \frac{\lambda_j}{\sum_{i=1}^{d} \lambda_i} \end{aligned}$$

and also, the KLT is the optimal among all linear transformations as for energy compaction within the first d components.

More information can be found in [12, 13, 14, 81].

2.1.2.4 Discrete Cosine Transform (DCT)

As we saw in the previous section, the KLT is the optimal transform for decorrelating a multidimensional signal. However, it entails computing the covariance matrix of a possibly very large multidimensional signal, and even worse, we have to find its eigendecomposition.

Under the assumption that the signal was generated from a first-order Markov process[4], the DCT performs closely to the KLT as for signal decorrelation, but with the advantage of not having to compute and diagonalize the covariance matrix. This is achieved by using a fixed basis, which is derived by using the aforementioned Markov process model.

The DCT is defined as

$$X\left[k\right] = \sum_{n=0}^{N-1} x\left[n\right] \cos\left(\frac{2\pi}{2N} \left(n + \frac{1}{2} \right) k \right)$$

[4] A first-order Markov process is a discrete stochastic process in which the outcome at the current sample is dependent only on the value of the last sample, i.e. $P\left(X\left[t\right] | X\left[t-1\right], X\left[t-2\right], \dots\right) = P\left(X\left[t\right] | X\left(t-1\right)\right)$. An autoregressive process is a Markov process.

This can be rewritten in matrix form as

$$X = D \cdot x$$

where

$$D_{nk} = \cos\left(\frac{\pi}{2N}\left(2n+1\right)k\right) \tag{2.3}$$

It can be shown that this DCT is equivalent (up to a scale factor of $1/2$) to the DFT of $x\left[n\right]$ upsampled by a factor of 2, and made even-symmetric:

$$\begin{aligned} y\left[2n\right] &= 0 \qquad \forall 0 \leq n < N \\ y\left[2n+1\right] &= x\left[n\right] \quad \forall 0 \leq n < N \\ y\left[4N-n\right] &= y\left[n\right] \quad \forall 0 \leq n < 2N \end{aligned}$$

Some authors define the DCT so that it becomes an orthogonal transformation (i.e. $D^T D = DD^T = I$). In that case, the DCT is defined as

$$X\left[k\right] = \begin{cases} \sqrt{\frac{1}{N}} \sum_{n=0}^{N-1} x\left[n\right] & k = 0 \\ \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x\left[n\right] \cos\left(\frac{2\pi}{2N}\left(n+\frac{1}{2}\right)k\right) & k \neq 0 \end{cases}$$

Next, we investigate if there exists a version of Parseval's identity for the DCT. For this purpose, we form the product $X^H X$ to produce the 2-norm of X and we will relate it to the 2-norm of x:

$$\begin{aligned} X &= Dx \\ X^H &= x^H D^H \end{aligned}$$

Now, per equation 2.3, D is a real matrix, so its Hermitian transpose is really nothing but D^T:

$$D^H_{nk} = \cos\left(\frac{\pi}{2N}\left(2k+1\right)n\right)$$

$$\begin{aligned} X^H X &= x^H D^H D x \\ &= x^H D^T D x \end{aligned}$$

Now, we have already seen that DCT are usually defined to be orthogonal, so that $D^T D = I$. Thus we arrive at the following result

$$X^H X \;=\; x^H x$$

or more compactly

$$\|\mathbf{DCT}\{x\}\|_2 = \|x\|_2 \tag{2.4}$$

This is the DCT analogous to Parseval's identity for DFT.

There are several types of DCT, depending on the boundary conditions: each boundary can have even or odd symmetry. In addition, the symmetry can be around a data point, or the "virtual" point between two data points. Therefore, there are actually $2 \times 2 \times 2 \times 2 = 8$ types of DCT. The DCT mentioned so far is known as DCT-II and is the most commonly used. It corresponds to $x[n]$ being even around $n = -\frac{1}{2}$ and even around $n = N - \frac{1}{2}$. $X[k]$ is even around $k = 0$ and odd around $k = N$.

Another DCT which is worth mentioning, is the DCT-IV:

$$X[k] = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi}{2N}\left(n + \frac{1}{2}\right)\left(k + \frac{1}{2}\right)\right)$$

This corresponds to $x[n]$ being even around $n = -\frac{1}{2}$ and odd around $n = N - \frac{1}{2}$ and the same for $X[k]$.

For more information see [10, 15].

2.1.3 Time-Frequency Representation

Yet another way to represent a signal is by combining information from both time and frequency domains. Consider a continuous time signal $x(t)$. When the signal is represented in the time domain, we have infinite precision in that domain, i.e. we have signal information for each possible value of t. However, the information we extract actually contains information from every possible frequency for a given t. Similarly, when we take the Fourier Transform $X(f)$, we have all the possible information we might want from the frequency domain, i.e. a different datum for each possible frequency f, but in this representation, the time variable has been "summed out", and it contains information from all times for a given f. This shortcoming in single domain representations and the duality is stated in what is known as the Uncertainty Principle.

2.1.3.1 The Heisenberg Uncertainty Principle

We will now show that the resolution one can attain in both time and frequency are interrelated (in most signals of interest, anyway). The way we will do this is by defining two signal parameters: effective duration and effective bandwidth. These parameters are representative of how long the signal is, and how wide its spectrum is.

Assume $f(t)$ and its Fourier transform $F(\omega)$ are absolutely integrable, that is the integrals $\int_{\mathbb{R}} |f(t)|\, dt$ and $\int_{\mathbb{R}} |F(\omega)|\, d\omega$ both converge. The effective duration Δt and the corresponding effective bandwidth $\Delta\omega$ are those which satisfy

$$\int_{-\infty}^{\infty} |f(t)|\, dt = \left|f\left(\overline{t}\right)\right| \Delta t$$

and

$$\int_{-\infty}^{\infty} |F(\omega)|\, d\omega = |F(\overline{\omega})|\, \Delta\omega$$

with

$$\overline{t} = \frac{\int_{-\infty}^{\infty} t\, |f(t)|\, dt}{\int_{-\infty}^{\infty} |f(t)|\, dt}$$

and

$$\overline{\omega} = \frac{\int_{-\infty}^{\infty} \omega\, |F(\omega)|\, d\omega}{\int_{-\infty}^{\infty} |F(\omega)|\, d\omega}$$

If we regard $|f(t)|$ as an unnormalized probability distribution, then $\overline{t}$ is the mean of the random variable whose probability distribution is $|f(t)|$; likewise $\overline{\omega}$ is the mean of the random variable whose probability distribution is $|F(\omega)|$.

From the definition of the Fourier Transform we have

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} f(t)\, e^{-j\omega t} dt \\ |F(\omega)| &\leq \int_{-\infty}^{\infty} \left|f(t)\, e^{-j\omega t}\right| dt \quad \forall\omega \\ &= \int_{-\infty}^{\infty} |f(t)|\, dt \end{aligned}$$

and

$$\begin{aligned} f(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\, e^{j\omega t} d\omega \\ |f(t)| &\leq \frac{1}{2\pi}\int_{-\infty}^{\infty} \left|F(\omega)\, e^{j\omega t}\right| d\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} |F(\omega)|\, d\omega \end{aligned}$$

Therefore, evaluating at $\overline{\omega}$ and $\overline{t}$ we have

$$\begin{aligned} |F(\overline{\omega})| &\leq \int_{-\infty}^{\infty} |f(t)|\, dt \\ &\leq \left|f\left(\overline{t}\right)\right| \Delta t \\ \left|f\left(\overline{t}\right)\right| &\leq \frac{1}{2\pi}\int_{-\infty}^{\infty} |F(\omega)|\, d\omega \\ &= \frac{1}{2\pi} |F(\overline{\omega})| \Delta\omega \\ \frac{1}{2\pi}\Delta t\Delta\omega\, |F(\overline{\omega})| \left|f\left(\overline{t}\right)\right| &\geq \left|f\left(\overline{t}\right)\right| |F(\overline{\omega})| \\ \Delta t\Delta\omega &\geq \frac{1}{2\pi} \end{aligned}$$

Define the signal energy as

$$E = \int_{-\infty}^{\infty} |f(t)|^2\, dt$$

We also have according to Parseval's Identity that

$$E = \frac{1}{2\pi}\int_{-\infty}^{\infty} |F(\omega)|^2\, d\omega$$

Define the effective duration and effective bandwidths as

$$(\Delta t)^2 = \frac{\int_{-\infty}^{\infty} t^2 |f(t)|^2\, dt}{E}$$

$$(\Delta\omega)^2 = \frac{\int_{-\infty}^{\infty} \omega^2 |F(\omega)|^2\, d\omega}{2\pi E}$$

If $\lim_{t\to\pm\infty} t\left|f\left(t\right)\right|^2 = 0$, then $\Delta t\Delta\omega \geq \frac{1}{2}$.

We recall the Cauchy-Schwartz Inequality for the $\mathcal{L}^2$ Hilbert space. For any $\mathcal{L}^2$ functions $z\left(x\right)$ and $w\left(x\right)$ defined in the interval $[a,b]$,

$$\left|\int_a^b z\left(x\right)w\left(x\right)dx\right|^2 \leq \int_a^b \left|z\left(x\right)\right|^2 dx \int_a^b \left|w\left(x\right)\right|^2 dx \tag{2.5}$$

which in vector notation becomes

$$\left\langle Z,W\right\rangle^2 \leq \left\langle Z,Z\right\rangle \left\langle W,W\right\rangle$$

or equivalently

$$\left\langle Z,W\right\rangle \leq \left\|Z\right\| \left\|W\right\|$$

We now prove the following

Equation 2.5 implies that

$$\left|\int_{\mathbb{R}} t\,f\left(t\right)\frac{df\left(t\right)}{dt}dt\right|^2 \leq \left(\int_{\mathbb{R}} t^2 f\left(t\right)^2 dt\right)\left(\int_{\mathbb{R}}\left|\frac{df\left(t\right)}{dt}\right|^2 dt\right)$$

Let

$$\begin{aligned} A &= \int_{\mathbb{R}} t\,f\left(t\right)\frac{df\left(t\right)}{dt}dt \\ &= \int_{\mathbb{R}} t\,\frac{d\left(\frac{f(t)^2}{2}\right)}{dt}dt \\ &= t\,\frac{f\left(t\right)^2}{2}\Bigg|_{-\infty}^{\infty} - \int_{\mathbb{R}}\frac{f\left(t\right)^2}{2}dt \end{aligned}$$

By hypothesis, $\left|t\right|f\left(t\right)^2 \to 0 \Rightarrow A = -\frac{E}{2}$.

We now recall that $\frac{d}{dt}f\left(t\right) \iff j\omega F\left(\omega\right)$, we then have using Parseval's Identity that

$$\int_{\mathbb{R}}\left|\frac{df\left(t\right)}{dt}\right|^2 dt = \frac{1}{2\pi}\int_{\mathbb{R}}\omega^2\left|F\left(\omega\right)\right|d\omega$$

Therefore,

$$\begin{aligned}
\left|-\frac{E}{2}\right|^2 &= \left|\int_{\mathbb{R}} t\, f(t)\, \frac{df(t)}{dt} dt\right|^2 \\
&\leq \int_{\mathbb{R}} t^2 f(t)^2\, dt \cdot \frac{1}{2\pi} \int_{\mathbb{R}} \omega^2 \left|F(\omega)\right|^2 d\omega \\
&= E(\Delta t)^2 \frac{1}{2\pi} \cdot 2\pi \cdot E(\Delta\omega)^2 \\
\frac{E^2}{4} &\leq E^2 (\Delta t)^2 (\Delta\omega)^2 \\
\Delta t \Delta\omega &\geq \frac{1}{2}
\end{aligned}$$

As we previously mentioned, when we take the Fourier Transform of a signal and evaluate it at any frequency, that value actually counts contributions from every instant in time when the signal has content. Similarly, at all times, $x(t)$ counts contributions from every possible frequency. While this might be useful for stationary signals, where the frequency content is constant over time, in non-stationary environments the Fourier Transform will represent frequency content "averaged" over all times. The reason for this all-or-nothing property of the Fourier Transform is that the basis functions over which the decomposition is performed, are complex exponentials (i.e. sines and cosines). While these functions are perfectly localized in frequency, they extend from $t = -\infty$ to $t = \infty$, as illustrated in equation 2.6.

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t} dt \tag{2.6}$$

Similarly, the inverse transform ponders each differential contribution of the frequency content by a basis function which is perfectly localized in time and extends from $\omega = -\infty$ to $\omega = \infty$ (equation 2.7).

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega)\, e^{j\omega t} d\omega \tag{2.7}$$

To overcome this extreme localization, several tools have been developed to have finite localization in both time and frequency. We will first analyze the Short-Time Fourier Transform.

2.1.3.2 Short-Time Fourier Transform (STFT)

The Short-time Fourier transform takes finite length blocks of a signal (called frames) and calculates the Fourier Transform over that block. Usually, frames must have some amount of overlap in order for the transform to be invertible. In addition, taking a frame

of finite length, is actually windowing the signal with a rectangular window. So in the frequency domain, this rectangular window will produce smearing or leakage within the spectrum. To overcome this smearing, we might use different windows before taking the transform. The STFT is defined in continuous time as

$$\begin{aligned} \mathbf{STFT}\left\{x\left(t\right)\right\}\left(\tau,\omega\right) &= X\left(\tau,\omega\right) \\ &= \int_{-\infty}^{\infty} x\left(t\right) w\left(t-\tau\right) e^{-j\omega t} \end{aligned}$$

In discrete time we have the continuous frequency variant

$$\begin{aligned} \mathbf{STFT}\left\{x\left[n\right]\right\}\left[m,\omega\right) &= X\left[m,\omega\right) \\ &= \sum_{n=-\infty}^{\infty} x\left[n\right] w\left[n-m\right] e^{-j\omega n} \end{aligned}$$

and the much more frequently appearing discrete-time, discrete-frequency variant, which stems from using the DFT instead of the DTFT used in the continuous frequency variant:

$$\begin{aligned} \mathbf{STFT}\left\{x\left[n\right]\right\}\left[m,k\right] &= X\left[m,k\right] \\ &= \sum_{n=m-N+1}^{m} x\left[n\right] w\left[n-m\right] W_N^{nk} \end{aligned}$$

For simplicity, we will use this STFT variant and note it as either

$$\mathbf{STFT}\left\{x\left[m\right]\right\}\left[k\right]$$

or

$$\mathbf{STFT}_N\left\{x\left[m\right]\right\}\left[k\right]$$

when the dependence upon the FFT length is non-obvious.

It can be proven that

$$X\left(\omega\right) = \int_{-\infty}^{\infty} X\left(\tau,\omega\right) d\tau$$

and also that

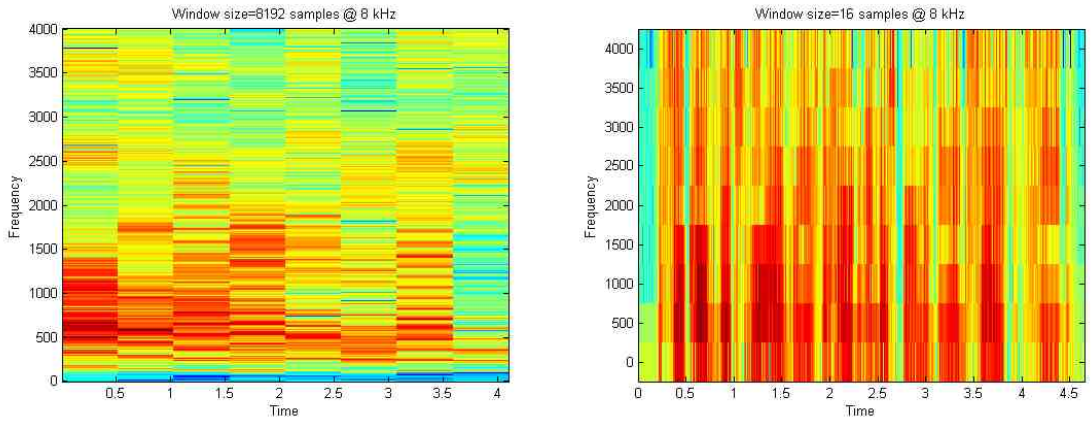
$$x\left(t\right) w\left(t-\tau\right) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X\left(\tau,\omega\right) e^{j\omega t} d\omega$$

In other words, if certain conditions are met on the window $w(t)$, then the STFT is an invertible transformation. These conditions are

$$\begin{aligned} w(t) &\neq 0 \quad \forall t \in I \\ \int_{-\infty}^{\infty} w(t)\, dt &= 1 \end{aligned}$$

where I is the set of values on which the window actually operates. That is the window cannot be zero for values "inside" of it.

Because of the Heisenberg Uncertainty Principle we discussed in section 2.1.3.1, the resolution we can get in the frequency domain for each processed frame, is related to the length of the frame (and consequently, of the window). The longer the window, the more accurate our frequency domain information will be (figure 2.3(a)). On the other hand, we will not be able to accurately pin-point fast time-domain events (our time resolution will be roughly the size of the window). Similarly, if we take a very short window in the time domain, we will be able to detect very quickly occurring events in the time domain, but our frequency resolution will be very poor (figure 2.3(b)).



(a) Long window

(b) Short window

Figure 2.3: Short-time Fourier Transform with different window lengths

Because the attainable frequency resolution is mandated by the resolution in the time domain, the STFT is a constant bandwidth analysis technique, which depending on the application may or may not be suitable. Constant Q analysis, another technique, is explored below.

2.1.3.3 Modified Discrete Cosine Transform (MDCT)

The MDCT is another invertible transform, based on the type IV DCT. However, unlike the DCT, is a lapped transform, meaning that it is meant to be applied to consecutive blocks of data, where consecutive blocks have an overlap of at least 50%. Because of this overlapping, and the good energy compaction properties of the DCT, make the MDCT attractive for audio compression, since artifacts produced by block boundary effects are avoided.

Mathematically, the MDCT is defined as a transform from a $2N$-samples long signal into a N-samples long signal:

$$X\left[k\right] = \sum_{n=0}^{2N-1} x\left[n\right] \cos\left(\frac{2\pi}{2N}\left(n + \frac{1}{2} + \frac{N}{2}\right)\left(k + \frac{1}{2}\right)\right)$$

Because the number of samples in the transform is half the number of samples of the input, one might wonder if it is invertible. Fortunately, by adding overlapping contiguous IMDCTs, one can perfectly reconstruct the original signal. This is known as Time Domain Aliasing Cancellation (TDAC). The IMDCT transforms the N numbers present in the MDCT, into a length $2N$ signal:

$$y\left[n\right] = \frac{1}{N} \sum_{k=0}^{N-1} X\left[k\right] \cos\left(\frac{2\pi}{2N}\left(n + \frac{1}{2} + \frac{N}{2}\right)\left(k + \frac{1}{2}\right)\right)$$

So, except for a factor of $\frac{1}{N}$ the IMDCT is the same as the MDCT.

Because of leakage produced by considering only a small block of the signal in the time domain, we will usually window the signal so that its frequency domain properties aren't affected by the finite duration of the analysis block[5]. In other words, we want to avoid block boundary effects, i.e. we want the data to smoothly go to zero at $n = 0$ and $n = 2N$. Hence, the procedure is to window the data before the MDCT *and* after the IMDCT. However, for the windowed MDCT to be invertible, the window has to satisfy certain properties:

1. The window has to be symmetric, i.e. $w\left[n\right] = w\left[2N - 1 - n\right]$

[5]By taking a finite block of data, one is already using a window: the rectangular or "box car" window. The frequency response of this window is convolved with the frequency response of the signal data. The rectangular window has the smallest main lobe, but the smallest stopband attenuation among all windows. Hence, the signal will have leakages at each frequency, from neighboring frequencies.

2. The window has to satisfy the Princen-Bradley condition:
   $w[n]^2 + w[n+N]^2 = 1 \; \forall 0 \leq n < N$.

Some popular choices for windows are

$$w[n] = \sin\left(\frac{\pi}{2N}\left(n + \frac{1}{2}\right)\right)$$

which is used in MP3, and

$$w[n] = \sin\left(\frac{\pi}{2}\sin^2\left(\frac{\pi}{2N}\left(n + \frac{1}{2}\right)\right)\right)$$

which is used in Vorbis.

Claim The windows used in MP3 and Vorbis satisfy the Princen-Bradley condition

Proof

MP3

$$\begin{aligned} w[n+N] &= \sin\left(\frac{\pi}{2N}\left(n + \frac{1}{2} + N\right)\right) \\ &= \sin\left(\frac{\pi}{2N}\left(n + \frac{1}{2}\right) + \frac{\pi}{2}\right) \\ &= \cos\left(\frac{\pi}{2N}\left(n + \frac{1}{2}\right)\right) \end{aligned}$$

$$\begin{aligned} w[n]^2 + w[n+N]^2 &= \sin^2\left(\frac{\pi}{2N}\left(n + \frac{1}{2}\right)\right) + \cos^2\left(\frac{\pi}{2N}\left(n + \frac{1}{2}\right)\right) \\ &= 1 \end{aligned}$$

Vorbis

$$\begin{aligned}\sin^2\left(\frac{\pi}{2N}\left(n+N+\frac{1}{2}\right)\right) &= \sin^2\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)+\frac{\pi}{2}\right)\\ &= \cos^2\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)\\ &= 1-\sin^2\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)\end{aligned}$$

$$\begin{aligned}w\left[n+N\right]^2 &= \sin^2\left(\frac{\pi}{2}\left(1-\sin^2\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)\right)\right)\\ &= \sin^2\left(\frac{\pi}{2}-\frac{\pi}{2}\sin^2\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)\right)\\ &= \cos^2\left(\frac{\pi}{2}\sin^2\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)\right)\end{aligned}$$

$$\begin{aligned}w\left[n\right]^2+w\left[n+N\right]^2 &= \sin^2\left(\frac{\pi}{2}\sin^2\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)\right)\\ &\quad+\cos^2\left(\frac{\pi}{2}\sin^2\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)\right)\\ &= 1\end{aligned}$$

It is worth noting that for even N, the MDCT is similar to a DCT-IV where the input is shifted by $\frac{N}{2}$ and two N-length blocks of data are transformed together.

For more information, see [16, 17].

2.1.3.4 Wavelets

Wavelet analysis is sometimes referred to as constant Q analysis. The Q refers to the quality factor, a parameter which appears when studying resonance systems. It is a measure of bandwidth of a system but relative to the frequency at which the bandwidth is measured:

$$Q = \frac{f_c}{\Delta f}$$

A high-Q and a low-Q second order filters are shown in figure 2.4.

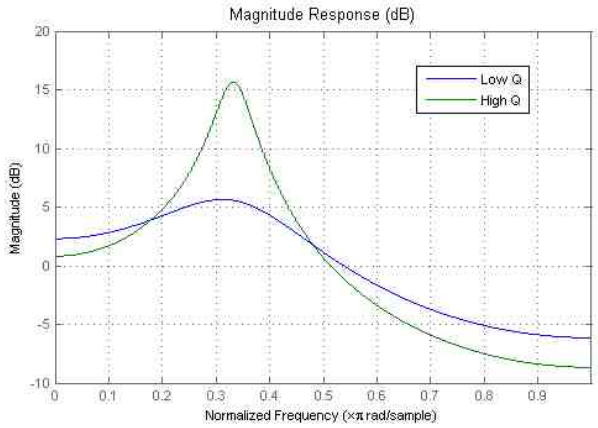


Figure 2.4: High-Q and low-Q filters

Wavelets perform an analysis that has a greater bandwidth at high frequencies, and lower bandwidth at lower frequencies. That way, the resolution is not constant over frequencies. This is very useful since a difference of 10 Hz in resolution at a central frequency of 30 Hz is 30%, but 10 Hz at a central frequency of 2000 Hz is only 0.5%. In the first case, the measurement error is so big it renders the analysis unusable, and in the second case the resolution is an overkill. With wavelets, we can have a resolution that has a constant ratio with respect to the central frequency, such as having 1 Hz at 200 Hz and 10 Hz at 2000 Hz, etc.

The Continuous Wavelet Transform is defined as

$$\gamma(t,s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} f(t)\, \overline{\psi\left(\frac{t-\tau}{s}\right)} dt \tag{2.8}$$

Several clarifications are in order. First, the Wavelet transform, being a time-frequency representation, converts a function of one variable (time) into a two-dimensional signal (time t and scale s). Scale plays a role similar to that of frequency[6] We see that equation 2.8, looks a lot like a decomposition (analysis) equation. Indeed, this equation actually projects the signal $f(t)$ into the space generated by a set of basis function $\psi_{\tau,s}(t) = \psi\left(\frac{t-\tau}{s}\right)$. Thus, the basis functions are called wavelets, in similarity to the sinusoidal waves used in Fourier Theory[7].

[6] Actually, scale is inversely proportional to frequency, so a more accurate statement would be to identify scale with period, although for non-periodic signals this identification loses its meaning.

[7] Actually, the name "wavelet" stems from its first usage in French, "*ondelette*", by Morlet and Grossman in the 1980s ([10]).

It is worth noting that all the basis functions, $\psi_{\tau,s}(t)$ originate from a single basic function ψ by translations and changes of scale. This ψ function is for that reason called the "mother wavelet".

One may wonder now, can any function be used as a mother wavelet? The answer is no, as there are several conditions a function must satisfy in order to be considered a wavelet:

1. $\int_{-\infty}^{\infty} \psi(t)\, dt = 0$ (zero mean condition)
2. $\int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < +\infty$ (*admissibility* condition)

The CWT has the problem of squaring the effective size of the representation of the signal, since it operates on two continuous variables. To overcome this, the Discrete Wavelet Transform (DWT) was devised:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{|s_0|^j}} \psi\left(\frac{t}{s_0^j} - k\tau_0\right)$$

It is desirable that the analysis wavelets be orthogonal to each other, for that way we guarantee a compact invertible representation. Thus, the wavelets must verify

$$\int \psi_{j,k}(t)\, \overline{\psi_{m,n}(t)} dt \quad = \quad \delta_{j-m,k-n}$$

where $\delta_{a,b}$ is the two-dimensional Kronecker Delta. The inverse DWT is thus

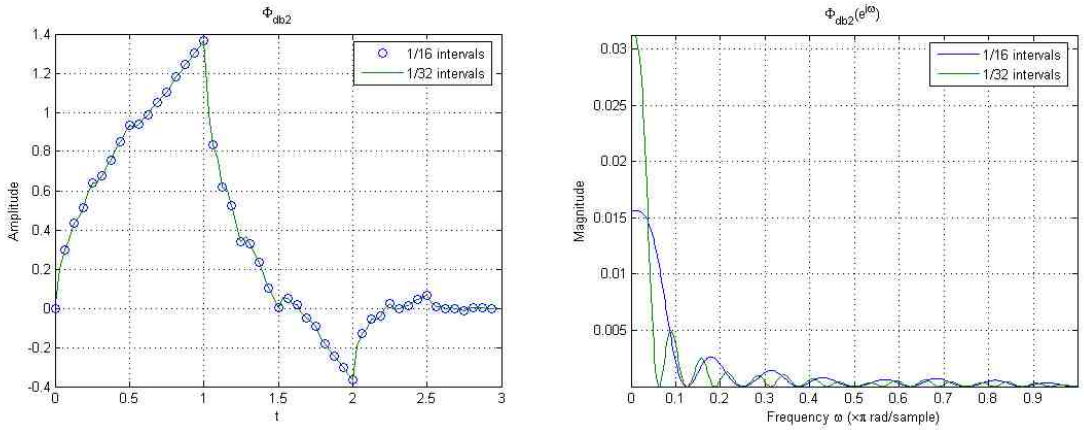
$$f(t) = \sum_j \sum_k F[j,k]\, \psi_{j,k}(t)$$

The DWT may be used for discrete-time signals, which results in a very compact and useful representation for a variety of areas, from pattern recognition, feature extraction, signal manipulation, etc. In this case we get

$$\begin{aligned} \lambda_{j-1}[k] &= \sum_m h[m-2k]\, \lambda_j[m] \\ \gamma_{j-1}[k] &= \sum_m g[m-2k]\, \gamma_j[m] \end{aligned}$$

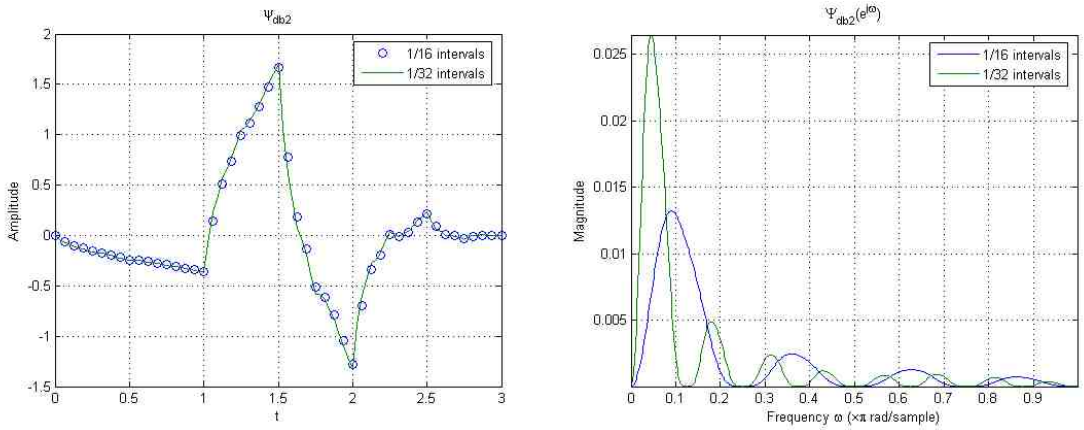
The coefficients λ_j and γ_j are the low-pass and high-pass energies of the signal at different frequency bands, respectively. The filter $h[n]$ is referred to as the scaling filter

and the $g[n]$ as the wavelet filter. In figure we see the Daubechies 4-tap wavelet (*db2*) scaling function in the time domain and its frequency response.



(a) Impulse response sampled at $\frac{1}{16}$ and $\frac{1}{32}$ intervals (b) Magnitude response

Figure 2.5: Scaling function $\phi(t)$ for the *db2* wavelet



(a) Impulse response sampled at $\frac{1}{16}$ and $\frac{1}{32}$ intervals (b) Magnitude response

Figure 2.6: Wavelet function $\psi(t)$ for the *db2* wavelet

The scaling function satisfies $\Phi(0) = 1$ while the wavelet function satisfies $\Psi(0) = 0$. This means that while the scaling function represents "averages", the wavelet represents "differences" in the analyzed signal.

The theory of Wavelets and the transforms associated with them is both recent and involved. Vaidyanathan ([18]) as well as Daubechies ([19]) have thoroughly approached the subject. A study on the use of wavelet coefficients as features for phoneme recognition is found in [21] where it is shown that the sampled continuous wavelet transform coefficients

(SCWT, also known as a scalogram) perform similarly to MFCCs in recognition tasks. More information on the Uncertainty Principle as well as on the STFT can be found in [8, 10].

2.1.4 Cepstrum

The Cepstrum is defined as the the FT of the log-spectrum of a signal, i.e.

$$\hat{X} = \mathbf{FFT}\left\{\log \mathbf{FFT}\left\{x\right\}\right\}$$

If we denote $x \leftrightarrow X$, we have that

$$\hat{X}\left[k\right] = \sum_{p=0}^{N-1} \log\left(X\left[p\right]\right) e^{-jp\omega_k}$$

The Cepstrum thus takes advantage of the periodicity in the spectrum. It is an invertible representation[8]:

$$\begin{aligned} X\left[p\right] &= \exp\left(\frac{1}{N}\sum_{k=0}^{N-1} \hat{X}\left[k\right] e^{jk\omega_p}\right) \\ &= \sqrt[N]{\prod_{k=0}^{N-1} \hat{X}\left[k\right] e^{jk\omega_p}} \end{aligned} \tag{2.9}$$

Just as in the time domain we have the index n and in the frequency domain the frequency bin index p, in the cepstral domain we use the index k. This index corresponds to the "quefrency" magnitude: low quefrency represents frequency components in the spectrum that vary at a low quefrency, that is, slowly varying components. Similarly, high-quefrency components are signal components which vary quickly. The cepstral representation has been proven to be a powerful tool in several applications. For example, in speech modeling, an approximation of the glottal filter might be obtained by removing the high-quefrency components of the signal. This is an application of homomorphic filtering theory, and specifically corresponds to a filter in the frequency domain, where the operation is known as "liftering". More information on cepstrum can be found in [8].

[8]Note, in formula 2.9, that the introduction of a radical in the process complicates matters, since in general the radicand will be a complex number. Therefore, we must be careful which of the N roots we choose for each of the vector components. For that purpose one usually unwraps the phase and chooses the root that makes the phase look most continuous.

2.2 Lossy Representations

When the ultimate goal of the application is analyzing the signal such as for classification, detection etc., some more compact representations can be used. These representations do not preserve all of the signal information, but for the purposes of that application might be sufficient. Furthermore, since synthesis is not a concern, the representation can be tailored to describe only the features of the signal that are relevant. For example, a very simple activity detector might use the average signal energy within a frame to discern between speech and silence. Therefore, instead of using a number of values equal to the frame length, it uses just one value (the RMS energy). It is obvious that the samples can not be recovered from this feature, but for the purposes of this very simple VAD, it might be enough.

In summation, lossy representations are useful in analysis scenarios: the usefulness usually arising from the compactness of the representation. To arrive at a representation suitable for a given application, we must have some criteria that drives our choice.

2.2.1 Optimality Criteria

The classic criteria for measuring the goodness of a representation for a signal, is to use some form of distance between the representation and the signal itself. If we call the signal x and its representation s using some vector space S, we have that

$$s = \arg\min_{s_i \in S} \|x - s_i\|$$

In general, the search (and factibility thereof) for this optimum depends heavily on two factors:

1. The cardinality (size) of the search space S
2. The tractability of the norm used to measure distances

For example, the search space might be finite-dimensional ($\mathbb{R}^n$) or infinite-dimensional ($\mathcal{L}_2$, $\mathbb{R}^{\mathbb{N}}$). A mathematical procedure to obtain the optimum would be to evaluate

$$s = \operatorname{sol} \frac{d}{ds_i} \|x - s_i\|$$

when the norm in question is a function of class $\mathcal{C}^1$ with the typical norm properties:

1. $\|v\| \geq 0 \quad \forall v \in S$ (non-negativity)
2. $\|v\| = 0 \iff v = 0$ (discernibility)
3. $\|v_1 + v_2\| \leq \|v_1\| + \|v_2\| \quad \forall v_1, v_2 \in S$ (triangle inequality)
4. $\|a \cdot v\| = |a| \, \|v\| \quad \forall a \in \mathbb{C}, v \in S$ (positive homogeneity)

The classical choice for a norm to make derivations mathematically tractable, is the 2-norm. In $\mathbb{R}^n$ the 2-norm is defined as

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

and for functions in $\mathcal{L}_2$

$$\|f\|_2 = \sqrt{\int_{\mathrm{dom}\, f} |f(x)|^2 \, dx}$$

2.2.2 Partial Correlation Coefficients (PARCOR)

Partial correlation coefficients or PARCOR, are a useful tool to model close-to-autoregressive processes, since they are derived from the autoregression coefficients. Given a process/signal $\{s[0] \ldots s[t-1]\}$, its forward AR model of order m is given by

$$s[l] = \sum_{i=1}^{m} a_m[i] \, s[l-i] + e_m^{\text{fwd}}[l] \tag{2.10}$$

where $\{a_m[i]\}_{i=1}^{m}$ are the forward AR coefficients and $e_m^{\text{fwd}}[l]$ are the forward prediction errors.

In an analogous manner, the backwards AR model is given by

$$s[l-m-1] = \sum_{i=1}^{m} b_m[l] \, s[l-i] + e_m^{\text{back}}[l] \tag{2.11}$$

which "predicts" past samples in terms of future samples.

As will be explained in section 2.3.3, the AR coefficients a_m and b_m are determined so as to minimize the mean square total forward and backward prediction error respectively. This minimizes the energy contained in that error: though the error might have large values, it can only have them for short periods of time, lest the 2-norm would greatly increase.

Compare this with using for instance the ∞-norm, where we minimize the maximum error. Although this usually is more desirable, it is much more difficult to attain because of the intractability of the $\max$ operator, especially the inability to take its derivative. The characteristic effect of minimizing the ∞-norm is to obtain errors which are fairly constant with a sinusoid-like behavior. This is called maximum ripple and is used in the design of FIR filters (Remez exchange and Parks-McClellan Algorithm).

In perfectly AR signals, the backwards coefficients b_m are the same as a_m only reversed. However, in reality there will always be some degree of non-autoregressivity, be it from the process source itself, or ambient noise. Thus, the PARCOR provide a unified forward-backward representation of a signal. The forward and backwards prediction coefficients, as well as the PARCOR can be estimated using the Levinson-Durbin recursion.

It is interesting to note that the partial correlation coefficients are closely related to the reflection coefficients of an acoustic tube, which measure the amount of energy reflected at each discontinuity inside an acoustic tube. Another interesting feature of PARCORs is that they always lie in the range $[-1, 1]$. Moreover, they have been shown to possess quantization properties very superior to the prediction coefficients, which when quantized, can render a stable filter unstable. Lastly, the PARCORs of a filter allows one to construct a very efficient filter structure known as a (reflection) lattice, analogous in structure to an electrical transmission line. More information can be found in [5, 8].

2.2.3 Real Cepstrum

In general, the spectrum of a signal will be a complex number, consisting of a magnitude and phase

$$X[p] = |X[p]| \exp\left(j\left(\angle X[p] + 2\pi m\right)\right) \quad \forall m \in \mathbb{Z}$$

Therefore, the logarithm of that will be

$$\log X[p] = \log |X[p]| + j\left(\angle X[p] + 2\pi m\right)$$

which is not unique (note the dependency on m). The phase is a continuous quantity so phase jumps must be taken care of by adding/subtracting multiples of 2π. This process is known as phase unwrapping.

The problem of phase unwrapping along with taking the logarithm of a potentially small quantity, poses several computational/numerical problems. Since usually the phase is of little interest in analysis applications (though we still need the phase for reconstruction),

the real cepstrum is defined as

$$\hat{X}\left[k\right] = \sum_{p=0}^{N-1} \log\left|X\left[p\right]\right| e^{-jk\omega_p}$$

As we said, this representation exploits the periodicity of the spectrum while discarding the phase. More information can be found in [8].

2.2.4 Linear-Scale Energies

One way to characterize a signal is by using the energy contained within a certain set of non-overlapping frequency bands. A particular choice for that set is to use M bands of equal width. This is known as a linear-scale energy representation. When one takes the N-point FFT, one actually is calculating the energy within each of the N frequency bands, each of which has width $\frac{2\pi}{N}$. Therefore, the energies are nothing but

$$E_p = \left|\sum_{n=0}^{N-1} x\left[n\right] e^{-jp\omega_n}\right|$$

Furthermore, we might take the logarithm of the magnitude of the energies to reduce the dynamic range of the representation.

2.2.5 Non-Linear Scale Energies

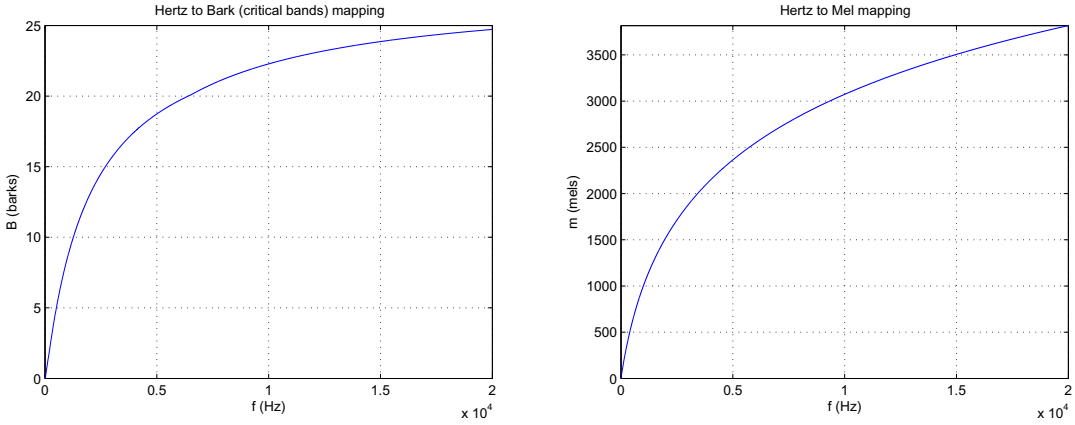
When dealing with perceptual applications, i.e. those which try to simulate or model our auditory system, we often resort to using energies on a non-linear scale. One such scale is the Bark scale proposed by E. Zwicker ([30]). The Bark scale is composed of 24 Barks, and each Bark spans a critical band bandwidth. A critical band is the area in the basilar membrane that resonates in response to a sine wave. Tones that lie on the same Bark will produce psychoacoustical phenomena such as beating, roughness, and auditory masking.

The mapping from linear frequency to Barks is given by

$$B = 13\arctan\left(0.00076f\right) + 3.5\arctan\left(\frac{f^2}{7500^2}\right)$$

Another scale used in psychoacoustics and perceptual analysis is the mel scale. It is comprised of tones which are judged to be equidistant from one another. The conversion is shown in figure 2.7 and is given by

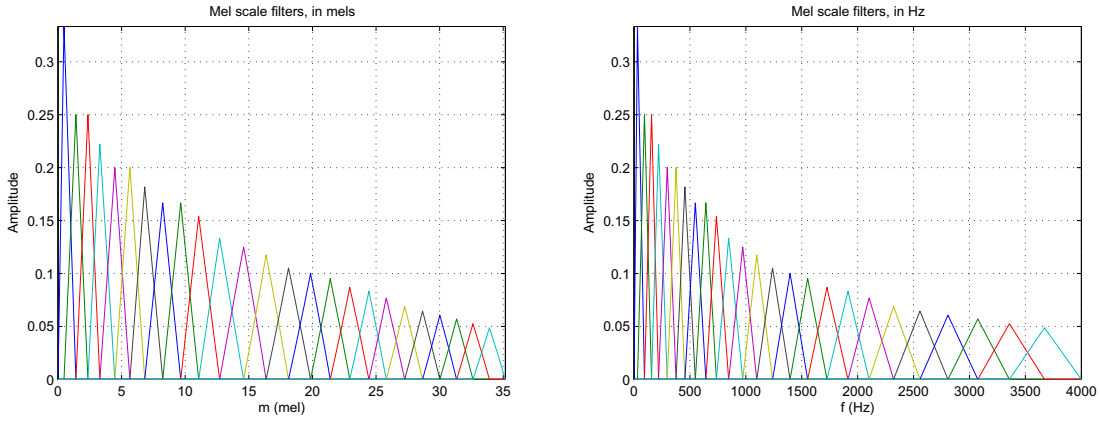
$$m = 1127.01048\ln\left(1 + \frac{f}{700}\right)$$



(a) Critical band mapping (Bark scale)

(b) Mel scale mapping

Figure 2.7: Non-linear scale mapping



(a) in mels

(b) in linear scale

Figure 2.8: Mel scale weighting filters

The mel scale is roughly linear until 1000 Hz and logarithmic from then on.

Just as we used linear scale energies, and driven by perceptually-based motivation, we can use the energy within a non-linear scale's bands as a feature vector. For that matter, the spectrum of a frame is calculated and then filtered through a number of filters to render energy values within each critical band. In the mel scale, the filters are triangular in shape, with bandwidth increasing with frequency, and conversely the gain of each filter is decreased with increasing frequency so that the energy per unit bandwidth is kept the same over all frequencies. Figure 2.8 depicts the filters' gain in the frequency domain.

More information can be found on [30, 31, 32].

2.2.6 Mel-Frequency Cepstral Coefficients (MFCC)

Davis and Mermelstein introduced MFCC as a speech recognition feature in [97]. MFCC can be viewed as a decorrelated version of the energies on each frequency bin, measured in a mel scale. Indeed, MFCC are defined as

$$M_x = \mathbf{DCT}\left\{\mathrm{Hz2Mel}\left\{\log\left|\mathbf{FFT}\left\{x\right\}\right|\right\}\right\}$$

Because of the Hertz-to-mel warping, MFCCs are usually a vector of about 13 real numbers, which usually exhibit a decaying trend. In [97], MFCC are computed on a frame-by-frame basis, so that a 13-dimensional vector is obtained every 30 ms or so. This turns out to be a powerful method to recognize voiced sounds, especially vowels, since different sounds will lie in mostly separable regions in this $\mathbb{R}^{13}$ space.

2.2.7 MFCC Differentials

To capture dynamic effects and artifacts in speech, such as articulation and speech modulation effects as well as to improve the recognition success, we can augment the 13-dimensional vector by appending the difference between consecutive MFCC vectors. Moreover, one may also include the different between consecutive MFCC differentials, which are called second-order differentials. Thus, we are left with a 39-dimensional vector for each frame.

Table 2.1: MFCC augmentation through its time differentials

$$\begin{cases} M_0\left[n\right] &= \text{vector of MFCCs for time-frame } n \\ M_1\left[n\right] &= M_0\left[n\right] - M_0\left[n-1\right] \\ M_2\left[n\right] &= M_1\left[n\right] - M_1\left[n-1\right] \\ &= M_0\left[n\right] - 2M_0\left[n-1\right] + M_0\left[n-2\right] \end{cases}$$

2.3 Speech Modeling

In order to gain some understanding of the problem at hand, we need to study to some extent the sound production mechanism and useful ways to mathematically model the processes that take place in speech.

2.3.1 The Vocal Tract

Speech sounds can be classified in many ways. One such classification pertains to whether the air flow through the vocal tract makes the vocal cords vibrate. If it does, we

say the sound it produces is voiced; otherwise, it is unvoiced. Voiced sounds have pitch, which is the frequency at which the folds vibrate, whereas unvoiced sounds are noise-like. Thus, voiced sounds are quasi-periodic. A vocal tract model is depicted in figure 2.9.

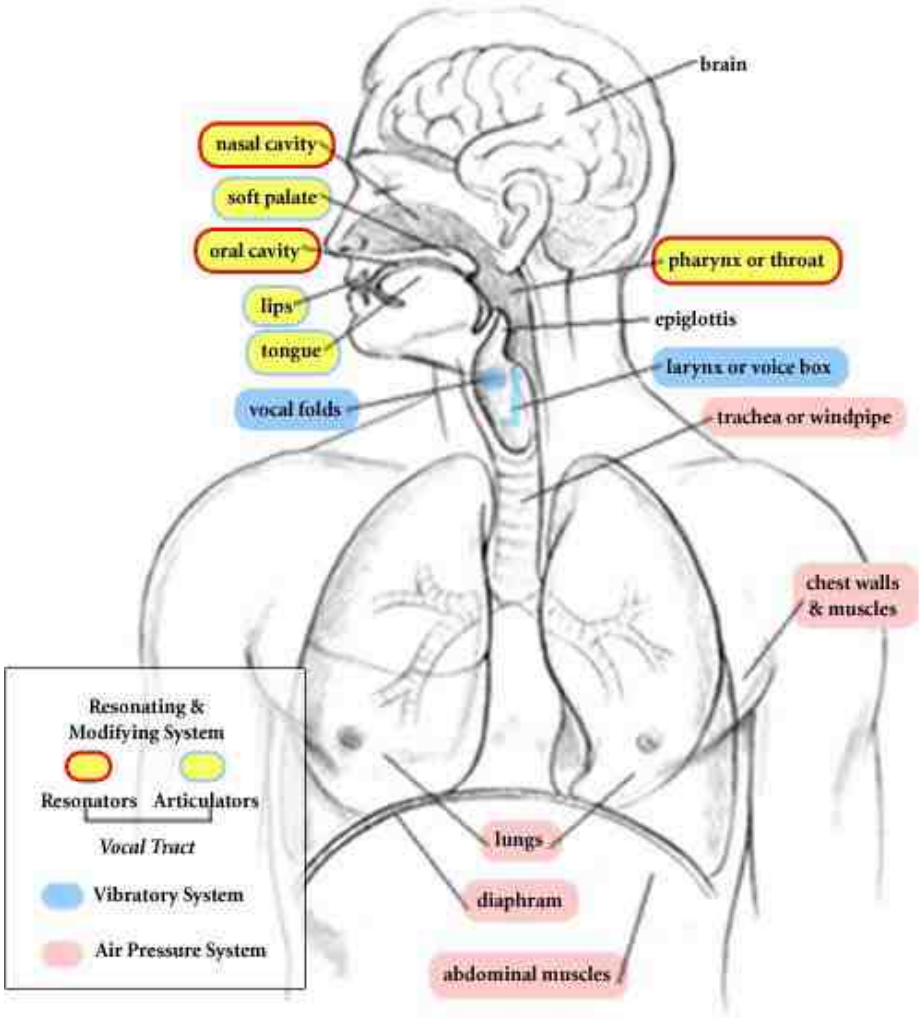


Figure 2.9: Vocal tract diagram (taken from [34])

2.3.2 Source-Filter Model

A useful model for the vocal tract is called the source-filter. In the source-filter model, the vocal tract is modeled as a system with two component: a source which can either be periodic (thus creating voiced sounds) or noisy (creating unvoiced sounds), and a filter which expresses the frequency response of the vocal tract. Therefore, it separates the excitation (airflow pressure) from the cavity resonant characteristics. This separation allows for speech processing techniques such as the vocoder, which changes the excitation and filter independently, thus rendering such effects as the well-known “singing-guitar”.

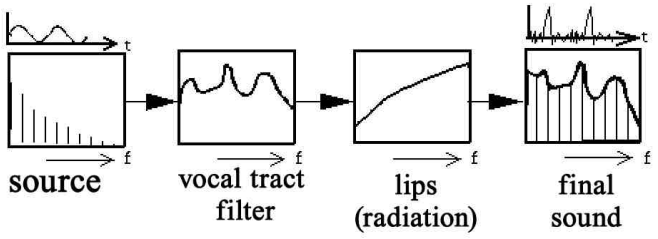


Figure 2.10: The source filter model

2.3.3 Autoregressive Processes

As already shown by equations 2.10 and 2.11, one can define coefficients $a_m[n]$ and $b_m[n]$ so that the m^{th}-order prediction residual is minimized in the mean square sense. This means that we choose $a_m[n]$ in

$$s[n] = \sum_{i=1}^{m} a_m[i]\, s[n-i] + e_m^{\text{fwd}}[n]$$

so that

$$\sum_{n=0}^{\infty} \left| e_m^{\text{fwd}}[n] \right|^2$$

is minimized. The problem of solving for the optimal autoregression coefficients is called Linear Prediction and reduces to solving a special kind of linear system of equations, which entails inverting a Toeplitz matrix. It is extensively studied in [5], where a procedure known as the Levinson-Durbin Recursion is used to efficiently and recursively calculate the coefficients. As stated in section 2.2.2, this procedure can also be used to extract the partial correlation coefficients.

Autoregressive filters are identifiable with all-pole IIR filters. This means that signals or filters with one or more zeros in the z-plane, will never be perfectly represented by an AR process. Nasalized sounds, such as /m/, /n/ or /ŋ/ exhibit zeros, arising from the coupling between the vocal tract and the nasal cavity: the nasal cavity acts as an energy trap for certain frequencies, at which the zeros occur. In addition to this, a shunting effect in the sinuses contribute to this energy trapping mechanism (see [40]).

2.3.4 Moving Average Processes

MA processes are processes where the output at a given time depends only on the current and previous value of the input. The past outputs are never fed back to the process, only the input gets processed as it moves forward in time, hence the name “moving aver-

age". MA filters are identifiable as FIR filters. These filters have a finite length impulse response and conversely their z-transform is a finite-order polynomial in z^{-1}. Therefore, the only poles are at $z = 0$, hence making these filters unconditionally stable. MA processes can try to approximate AR processes as the latter can try to approximate the former, but always with similar limitations. In this case, we can never successfully approximate a resonance. When trying to approximate a damped peak, the MA estimate will exhibit ripples and deviations from the AR response. More information in [5].

2.3.5 ARMA Processes

ARMA processes are processes which combine both past inputs and past outputs to produce the current sample's output. They are more general processes than MA and AR in the sense that they usually need less coefficients to fit a certain given response than their MA or AR counterparts.

2.4 Linear Prediction

Linear prediction is the process of estimating the future behavior of a signal based on a number of past values. The process can be stated as follows: given a signal $x = \{x[n]\}$ and a positive integer N, calculate the numbers $a_k \quad \forall\, k = 1, \ldots N$ so that the prediction error defined as

$$e[n] \;=\; x[n] - \sum_{k=1}^{N} a_k x[n-k]$$

has minimal energy

$$\begin{aligned}
\{a_k\} &= \arg\min \left(x[n] - \sum_{k=1}^{N} a_k x[n-k] \right)^2 \\
&= \arg\min \left(x[n]^2 + \left(\sum_{k=1}^{N} a_k x[n-k] \right)^2 - 2 \sum_{k=1}^{N} a_k x[n]\, x[n-k] \right) \\
&= \arg\min \left(\sum_{k=1}^{N} a_k x[n-k] \right)^2 - 2 \sum_{k=1}^{N} a_k x[n]\, x[n-k]
\end{aligned}$$

For this purpose we can set the derivative of the error energy with respect to a_k to zero:

$$\begin{aligned}\{a_k\} &= \operatorname{sol}\left\{2\left(\sum_{k=1}^{N} a_k x\left[n-k\right]\right)x\left[n-k\right] - 2x\left[n\right]x\left[n-k\right]\right\} \\ &= \operatorname{sol}\left\{\sum_{k=1}^{N} a_k x\left[n-k\right] - x\left[n\right]\right\}\end{aligned}$$

which is a self-referential equation. The linear prediction parameters are known as LPC and efficient algorithms to solve the problem such as the Levinson-Durbin recursion exist.

The set of LPC and even the cepstrum resulting from the LPC can successfully be used as features in a speech recognition system. More information in [5, 8].

2.5 Adaptive Filters

Adaptive filters are a class of iterative algorithms which try to estimate a set of optimal coefficients so that a certain objective function, whose arguments are those coefficients as well as input-output data, is minimized. Formally, the problem is as follows

Algorithm 1 Formal statement of an adaptive algorithm

Given data D, and a criterion function J, find coefficients w_0 so that

$$w_0 = \arg\min_w J_w\left(D\right)$$

1. Start with an initial guess w_0^0.
2. Based on the current estimate w_0^n, *calculate* w_0^{n+1}

   $$w_0^{n+1} = F\left(w_0^n, D^n, \dots\right)$$

   so that

   $$J_{w_0^{n+1}}\left(D\right) \leq J_{w_0^n}\left(D\right)$$

3. If the estimates have been within a certain tolerance for a sufficient number of iterations or after a certain number of iterations, stop. Otherwise go to step 2.

From this statement, it is clear that adaptive algorithms search for the solution to an optimization problem, namely the minimization of an objective function, sometimes referred to as the cost function.

In the following sections we examine a few adaptive filters and their properties. They will be a crucial part of our solution to approach the simulation problem. More information about the topics in this section can be found in [5].

2.5.1 Adaptive Filters In Our Context

For our purposes, we will restate the adaptive filter problem within the bounds of our context. Given an input signal $x = \{x[n]\}$, and a desired output signal $d = \{d[n]\}$, an FIR adaptive filter finds coefficients h so that the error signal $e[n]$ is minimized in the mean-square sense. Formally speaking, we have that

$$\begin{aligned} y[n] &= (h * x)[n] \\ e[n] &= y[n] - d[n] \\ h &= \arg\min \mathbb{E}\left\{e^2[n]\right\} \end{aligned}$$

where $\mathbb{E}\{\cdot\}$ is the statistical expectation operator.

2.5.1.1 The Wiener Filter

Expanding the previous equation we get

$$\begin{aligned} \mathbb{E}\left\{e^2[n]\right\} &= \mathbb{E}\left\{(y[n] - d[n])^2\right\} \\ &= \mathbb{E}\left\{y^2[n] + d^2[n] - 2y[n]\,d[n]\right\} \\ &= \mathbb{E}\left\{\left(\sum_{i=0}^{N} h_i x[n-i]\right)^2\right\} + \mathbb{E}\left\{d^2[n]\right\} - 2\mathbb{E}\left\{\sum_{i=0}^{N} h_i x[n-i]\,d[n]\right\} \end{aligned}$$

If the input and output sequences are wide-sense stationary processes[9], we can write

$$\begin{aligned} R_x[m] &= \mathbb{E}\{x[n]\,x[n+m]\} \\ R_{xd}[m] &= \mathbb{E}\{x[n]\,d[n+m]\} \end{aligned}$$

In order to obtain the optimal h_i, we must solve $\frac{\partial}{\partial h_i}\mathbb{E}\{e^2[n]\} = 0$. Carrying on with our manipulations we have

$$\begin{aligned} \frac{\partial}{\partial h_i}\mathbb{E}\left\{e^2[n]\right\} &= 2\sum_{j=0}^{N} R_x[j-i]\,h_j - 2R_{dx}[i] \quad \forall i = 0 \ldots N \\ \sum_{j=0}^{N} R_x[j-i]\,h_j &= R_{dx}[i] \end{aligned}$$

[9] This assumption is a bit strong for speech signals, since they tend to be stationary only for very short periods of time. Nevertheless, we will later develop algorithms that do not make such strong assumptions.

This last equation can be rewritten more compactly in matrix form as

$$\begin{bmatrix} R_x[0] & R_x[1] & R_x[2] & \dots & R_x[N] \\ R_x[1] & R_x[0] & R_x[1] & \dots & R_x[N-1] \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ R_x[N] & R_x[N-1] & \dots & R_x[1] & R_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \\ h_N \end{bmatrix} = \begin{bmatrix} R_{dx}[0] \\ R_{dx}[1] \\ \vdots \\ R_{dx}[N-1] \\ R_{dx}[N] \end{bmatrix}$$

These are called the Wiener-Hopf equations.

$$\mathcal{R}h = c \tag{2.12}$$

If the autocorrelation matrix $\mathcal{R}$ is nonsingular, the Wiener filter is then

$$h = \mathcal{R}^{-1}c \tag{2.13}$$

and it is optimal in the sense that it minimizes the expectation of the energy of the error (i.e. the mean-square error). Note that the Wiener filter is not an iterative algorithm in the sense that it needs complete knowledge of the input and output data to produce the filter coefficients estimate, and it is a one step process. In the following sections we derive algorithms to perform such estimates iteratively, adapting to the incoming data (hence the term "adaptive filter").

2.5.2 Steepest Descent (SD) Method

The steepest descent method is a very well-known method to find numerical solutions to a wide range of mathematical problems. It can be summarized as follows

Algorithm 2 Steepest descent

1. Start with an initial filter coefficient vector guess h^0
2. Calculate the gradient vector of the cost function with respect to the filter coefficients at the current guess h^n
$$v^n = \nabla_h J\left(D, h^n\right)$$
3. Advance a small distance in the direction of maximum decrease in cost function value
$$h^{n+1} = h^n - \frac{1}{2}\mu v^n$$
4. If cost function is close to final desired value $\left|J\left(D, h^{n+1}\right)\right| \leq \varepsilon$, stop. Otherwise go to step 2.

Since the gradient vector always points in the direction of maximum increase in value, its negative points in the direction of maximum decrease, hence the name "steepest descent". It is as if we want to climb down a mountain by always going through the steepest paths downhill.

To convert the Wiener filter into a steepest descent algorithm, one can note that the gradient of the cost function is

$$\begin{aligned}
\frac{\partial J}{\partial h_i^n} &= -2c\left[i-1\right] + 2\sum_{j=1}^{N} h_j^n R_x\left[j-i\right] \\
&= -2\mathbb{E}\left\{e\left[n\right]x\left[n-k\right]\right\} \\
\nabla J &= -2\mathbb{E}\left\{e\left[n\right]\underline{\mathrm{x}}\left[n\right]\right\}
\end{aligned}$$

where $\underline{\mathrm{x}}\left[n\right]$ is the column vector $\left[x\left[n\right], x\left[n-1\right], \ldots, x\left[n-N+1\right]\right]$

$$\begin{aligned}
h^{n+1} &= h^n - \frac{\mu}{2}\nabla J\left(n\right) \\
&= h^n + \mu\mathbb{E}\left\{e\left[n\right]\underline{\mathrm{x}}\left[n\right]\right\}
\end{aligned}$$

$$\begin{aligned}
e\left[n\right] &= d\left[n\right] - \underline{\mathrm{x}}^T\left[n\right]h^n \\
h^{n+1} &= h^n + \mu\mathbb{E}\left\{\underline{\mathrm{x}}\left[n\right]\left(d\left[n\right] - \underline{\mathrm{x}}^T\left[n\right]h^n\right)\right\} \\
&= h^n + \mu\mathbb{E}\left\{\underline{\mathrm{x}}\left[n\right]d\left[n\right]\right\} - \mu\mathbb{E}\left\{\underline{\mathrm{x}}\left[n\right]\underline{\mathrm{x}}^T\left[n\right]\right\}h^n \\
&= \left(I - \mu\mathcal{R}\right)h^n + \mu c
\end{aligned}$$

$$\nabla_h J\left(D, h^n\right) = \mathcal{R}h^n - c$$

Note that whenever we will have $\nabla J = 0$, it will mean that $\mathcal{R}h^n - c = 0$, therefore $\mathcal{R}h^n = c$, which is the condition for the Wiener filter.

The μvalue controls how far we go at each iteration. A large μmeans long steps and quicker initial convergence. Small μvalues result in slower but safer convergence. Note that unless we carefully control the value of the advance step μ, we might end up with an unstable algorithm. The necessary conditions for stability in this case boil down to

$$\mu \;>\; 0$$

and

$$|1 - \mu\lambda_k| < 1 \quad \forall k$$

where λ_k are the eigenvalues of the autocorrelation matrix. Therefore,

$$0 < \mu < \frac{2}{\lambda_{\text{max}}}$$

Although the steepest descent algorithm converges to the Wiener filter under the stability conditions outlined above, it requires the exact calculation of the gradient vector at each iteration, which is impossible to perform in real time.

2.5.3 Least Mean-Squares (LMS) Filters

In order to palliate the stringent requirements of the SD method as to gradient calculation, the LMS filter was devised. The LMS filter uses an instantaneous unbiased estimate of the gradient value at each iteration step, *viz.*

$$\hat{\nabla}\left(n\right) = -2e\left[n\right]\underline{x}\left[n\right]$$

Therefore, the actual update rule becomes

$$\begin{aligned} h^{n+1} &= h^n - \frac{1}{2}\mu\hat{\nabla}\left(n\right) \\ &= h^n + \mu e\left[n\right]\underline{\mathbf{x}}\left[n\right] \end{aligned}$$

where the error signal is defined as $e\left[n\right] = d\left[n\right] - \underline{\mathbf{x}}^T\left[n\right]h^n$. An LMS filter's signal flow graph is depicted in figure 2.11.

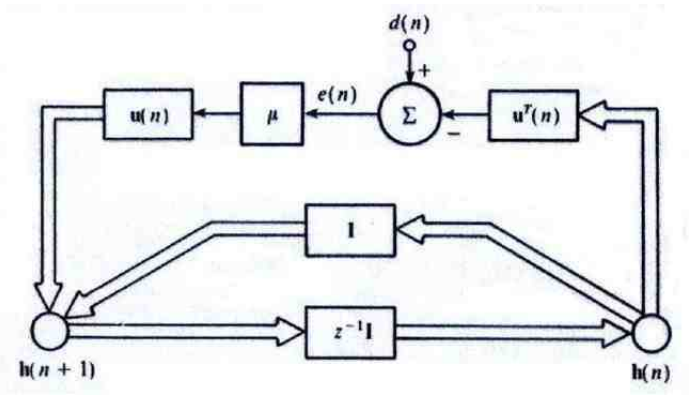


Figure 2.11: LMS signal-flow graph

To do away with the dependence on the step value μ and signal scales, we normalize the step value by the energy of the input frame $\underline{\mathrm{x}}\,[n]$ to render the Normalized Least mean square variant:

$$\mu_{NLMS} = \frac{\mu}{\left\|\underline{\mathrm{x}}\,[n]\right\|^2 + \delta}$$

The introduction of δ is due to the fact that if the frame contains little energy, the step size would exceed acceptable limits. Thus δ acts as a damping coefficient and is usually set to a very small number. μ is usually $\frac{1}{2}$. In this case the algorithm becomes

Algorithm 3 Normalized LMS

1. Choose h^0
2. $e\,[n] = d\,[n] - \underline{\mathbf{X}}^H h^n$
3. $h^{n+1} = h^n + \frac{1}{\underline{\mathbf{X}}^H[n]\underline{\mathbf{X}}[n]} e\,[n]\,\underline{\mathrm{x}}\,[n]$

The NLMS algorithm converges much faster than its simple LMS counterpart at little extra cost.

More information can be found in [5, 7, 39].

Property 1. The least-squares estimate of the coefficient vector approaches the optimum Wiener solution as the data length n approaches infinity, if the filter input and the desired response are jointly stationary ergodic processes.

Property 2. The least-squares estimate of the coefficient vector is unbiased if the error signal $e\,[n]$ has zero mean $\forall n$.

Property 3. The covariance matrix of the least-squares estimate h^n equals $\mathcal{R}^{-1}$, except for a scaling factor, if the error vector $e[n]$ has zero mean and its elements are uncorrelated (e.g. white noise).

Property 4. If the elements of the error vector $e[n]$ are statistically independent and Gaussian-distributed, then the least-squares estimate is the same as the maximum-likelihood estimate.

2.5.4 Recursive Least Squares (RLS) Filters

In the LMS and NLMS, we are only using information from the current input frame $\underline{\mathrm{x}}[n]$. RLS filters use information of all the input seen up until now, therefore rendering much faster convergence and accuracy, at the cost of extra computations. We now introduce an important lemma which is central to the derivation of the RLS filter.

2.5.4.1 The Matrix Inversion Lemma

Let A and B be two $M \times M$ positive definite matrices, C is an $M \times N$ matrix and D a positive definite $N \times N$ matrix such that

$$A = B + CDC^T$$

Then, we can express the inverse matrix as

$$A^{-1} = B^{-1} - B^{-1}C\left(D^{-1} + C^T B^{-1} C\right)^{-1} C^T B^{-1}$$

We now modify the autocorrelation matrix by adding a small damping constant ε to the diagonal. This step is necessary to insure positive-definiteness:

$$\mathcal{R}'_n[k,m] = \sum_{i=1}^{n} x[i-m]\, x[i-k] + \varepsilon\delta_{km}$$

where $\delta_{km}$is the Kronecker delta at $[k,m]$. This can be rewritten as

$$\begin{aligned}\mathcal{R}'_n[k,m] &= x[n-m]\, x[n-k] + \left(\sum_{i=1}^{n-1} x[i-m]\, x[i-k] + \varepsilon\delta_{km}\right) \\ &= x[n-m]\, x[n-k] + \mathcal{R}'_{n-1}[k,m]\end{aligned}$$

Now we let

$$\begin{aligned} A &= \mathcal{R}'_n \\ B &= \mathcal{R}'_{n-1} \\ C &= \underline{\mathrm{x}}\,[n] \\ D &= 1 \end{aligned}$$

so that when we use the matrix inversion lemma we get

$$\left(\mathcal{R}'_n\right)^{-1} = \left(\mathcal{R}'_{n-1}\right)^{-1} - \frac{\left(\mathcal{R}'_{n-1}\right)^{-1} \underline{\mathrm{x}}\,[n]\, \underline{\mathrm{x}}^T\,[n]\, \left(\mathcal{R}'_{n-1}\right)^{-1}}{1 + \underline{\mathrm{x}}^T\,[n]\, \left(\mathcal{R}'_{n-1}\right)^{-1} \underline{\mathrm{x}}\,[n]}$$

This is a recursive way to calculate the inverse correlation matrix $\left(\mathcal{R}'_n\right)^{-1}$ in terms of the previous step correlation matrix estimate $\left(\mathcal{R}'_{n-1}\right)^{-1}$ and the current input data frame.

Algorithm 4 Recursive Least-Squares (RLS)

p	filter order
λ	forgetting factor ($\rightarrow 1$)
δ	damping constant ($\rightarrow 0$)

- $h^0 = \underline{0}$
- $P[0] = \delta^{-1} I$

1. $\underline{\mathrm{x}}[n] = (x[n], \ldots, x[n-p])$
2. Calculate the gain vector
$$k[n] = \frac{P[n-1]\,\underline{\mathrm{x}}[n]}{\lambda + \underline{\mathrm{x}}^T[n]\,P[n-1]\,\underline{\mathrm{x}}[n]}$$
3. Calculate the estimation error
$$\eta[n] = d[n] - \underline{\mathrm{x}}^T[n]\,h^{n-1}$$
4. Update the coefficient estimation
$$h^n = h^{n-1} + k[n]\,\eta[n]$$
5. Update the inverse correlation matrix estimate
$$P[n] = \lambda^{-1}\left(P[n-1] - k[n]\,\underline{\mathrm{x}}^T[n]\,P[n-1]\right)$$
6. Go to step 1

1. In the LMS algorithm, the correction that is applied in updating the old estimate of the coefficient vector is based on the instantaneous sample value of the tap-input vector and the error signal. On the other hand, in the RLS algorithm the computation of this correction utilizes all the past available information, because of its recursive nature.

2. In the LMS algorithms, the correction applied to the previous estimate consists of the product of three factors: the (scalar) step-size parameter μ, the error signal $e[n-1]$, and the tap-input vector $\underline{\mathrm{x}}[n-1]$. On the other hand, in the RLS algorithm this correction consists of the product of two factors: the true estimation error $\eta[n]$ and the gain vector $k[n]$. The gain vector itself consists of $\left(\mathcal{R}'_{n-1}\right)^{-1}$, the inverse of the deterministic correlation matrix, multiplied by the tap-input vector $\underline{\mathrm{x}}[n]$. The major difference between the LMS and RLS algorithms is therefore the presence of $\left(\mathcal{R}'_{n-1}\right)^{-1}$ in the correction term of the RLS algorithm that has the effect of **decorrelating the successive tap inputs**, thereby making the RLS algorithm *self-orthogonalizing*. Because of this property, the RLS algorithm is essentially independent of the eigenvalue spread of the correlation matrix of the filter input.

3. The LMS algorithm requires approximately $20M$ iterations to converge in mean square, where M is the number of tap coefficients contained in the tapped-delay-line filter. On the other band, the RLS algorithm converges in mean square within less than $2M$ iterations. The rate of convergence of the RLS algorithm is therefore, in general, faster than that of the LMS algorithm by an order of magnitude.

4. Unlike the LMS algorithm, there are no approximations made in the derivation of the RLS algorithm. Accordingly, as the number of iterations approaches infinity, the least-squares estimate of the coefficient vector approaches the optimum Wiener value, and correspondingly, the mean-square error approaches the minimum value possible. In other words, the RLS algorithm, in theory, exhibits zero misadjustment. On the other hand, the LMS algorithm always exhibits a nonzero misadjustment; however, this misadjustment may be made arbitrarily small by using a sufficiently small step-size parameter μ.

5. The superior performance of the RLS algorithm compared to the LMS algorithm, however, is attained at the expense of a large increase in computational complexity. The complexity of an adaptive algorithm for real-time operation is determined by two principal factors:

(a) the number of multiplications (with divisions counted as multiplications) per iteration, and

(b) the precision required to perform arithmetic operations.

The RLS algorithm requires a total of $\frac{3}{2}M\left(3+M\right)$ multiplications, which increases as M^2, the square of the number of filter coefficients. On the other hand, the LMS algorithm requires $2M+1$ multiplications, increasing linearly with M. For example, for $M = 31$ the RLS algorithm requires 1581 multiplications, whereas the LMS algorithm requires only 63.

More information can be found in [4, 5, 7].

2.5.5 ARMA RLS

All of the adaptive filters we have reviewed in the previous sections, estimate the coefficients of a tapped delay line filter structure, i.e. an FIR filter. If the filter to estimate is stable and its impulse response trails off to zero for $|n| > N$, then the filter might be approximated by an FIR filter. Namely, the conditions are

1.
$$\sum_{n=-\infty}^{\infty} |h\left[n\right]| < \infty$$

2.
$$\forall \delta > 0 \; \exists N_\delta > 0 / \; |h\left[n\right]| < \delta \; \forall \, |n| > N_\delta \tag{2.14}$$

Condition (2) may be restated in relative terms with respect to the maximum impulse response sample value as

$$\forall 0 < \delta < 1 \; \exists N_\delta > 0 / \; |h\left[n\right]| < \delta \left\|h\right\|_\infty \; \forall \, |n| > N_\delta \tag{2.15}$$

Note there may be some filters such that condition (2) is satisfied but not condition (1):

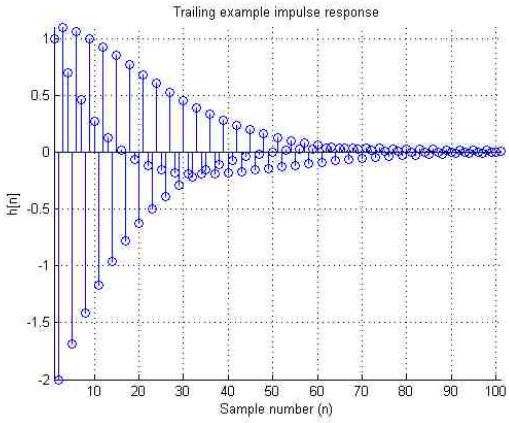
$$h[n] = \begin{cases} \frac{1}{n} & n \neq 0 \\ 0 & n = 0 \end{cases}$$

On the other hand, decaying to zero is a necessary condition for absolute summability; nevertheless, condition (2) gives an acceptable length N_δ for truncating an infinite impulse response filter and obtaining an approximately equivalent, length $2N_\delta$ FIR filter.

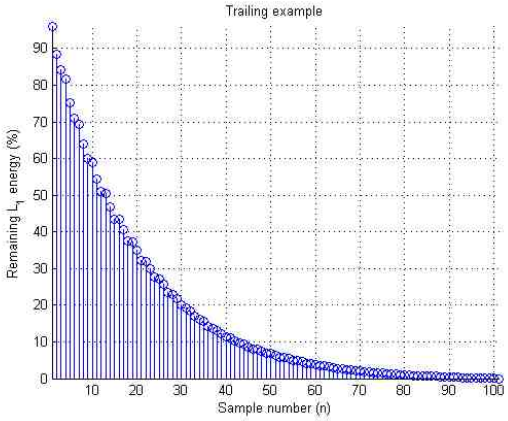
The disadvantage to use truncated IIR filters (FIR) as opposed to using IIR filters, is the necessary number of coefficients to represent the filter. Consider the filter

$$H(z) = \frac{1 - z^{-1}}{1 + z^{-1} + 0.9z^{-2}} \tag{2.16}$$

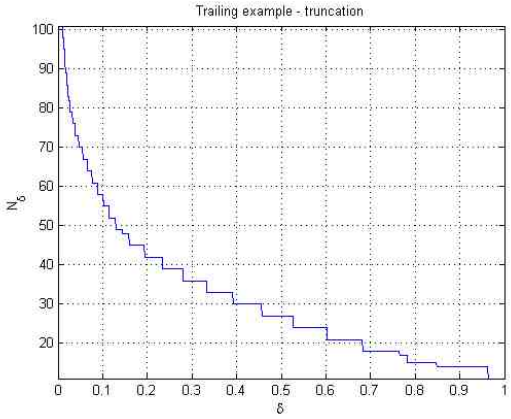
This filter can be characterized by four transfer function coefficients (1, -1, 1 and 0.9), or conversely by the location of its poles and zeros and its gain (1, $-\frac{1}{2}+\sqrt{\frac{13}{20}}j$, $-\frac{1}{2}-\sqrt{\frac{13}{20}}j$, 1) or some other parametrization; the important thing here is that the minimum number of coefficients to represent the filter is four. On the other hand, figure 2.12 depicts how the impulse response of this particular filter behaves.



(a) Impulse response



(b) Remaining L_1 energy after n samples



(c) Truncation length N_δ as defined in 2.15

Figure 2.12: Impulse response behavior of the filter defined in equation 2.16

We see how almost 60 samples are necessary to contain all impulse response samples greater than 10% of the maximum. At the same time, 60 samples leaves out about 5% of the $\mathcal{L}_1$ energy ($\sum |h[n]|$). In any case, to achieve successful representation of the filter characteristic, we need about 15 times more numbers. This ratio gets worse as the poles of the original transfer function get closer to the unit circle, for its magnitude response peaks will become sharper, which in turn lengthens the effective impulse response (this is an application of the Uncertainty Principle discussed in 2.1.3.1)

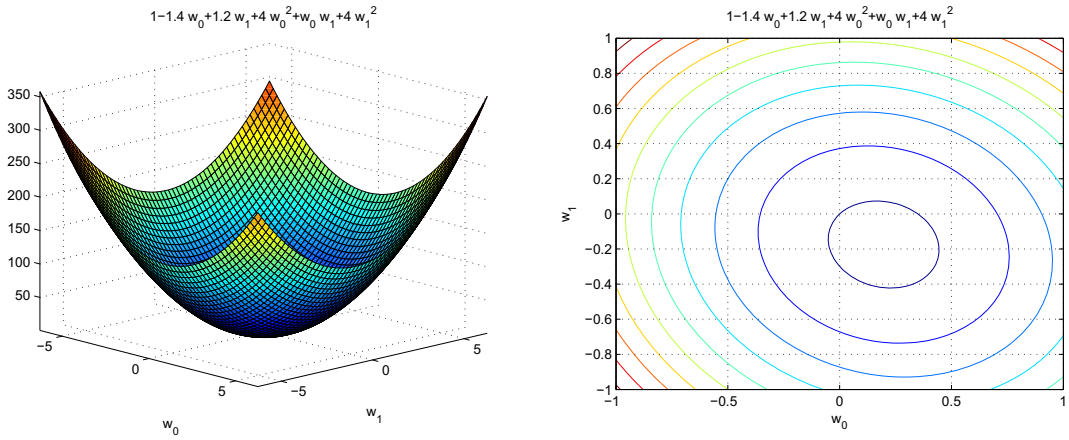
To overcome the representation inefficiency of which FIR filters suffer, one can try to fit the a given input-output realization to an ARMA model (an IIR filter). However, the procedures involved are much more complex than those for FIR estimation:

1. Since the cost function is quadratic in the error sample values and filter coefficients, its derivative will be linear in the error sample values. This gives rise to simple and tractable mathematical expressions.

2. The cost function has a nice physical interpretation as energy

3. The performance surface is smooth and has continuous derivatives

4. The performance surface is a convex n-dimensional paraboloid with a single global minimum. This means that if we start our iterations relatively near to a minimum, we will converge towards it. This idea is exploited in the Steepest descent method (figure 2.13)



(a) Two-dimensional paraboloid surface. This corresponds to a memoryless cost function in the tap-weight space.

(b) Plot of equal cost contours

Figure 2.13: FIR filter performance surface and contour plot

On the other hand, IIR filters have associated performance surfaces which exhibit multiple local minima. Consider the filter given by

$$H(z) = \frac{0.05 - 0.04z^{-1}}{1 - 1.1314z^{-1} + 0.25z^{-2}}$$

Let's assume we try to approximate this filter using a first order model, such as

$$\frac{Y_o(z)}{X(z)} = \frac{b}{1 - az^{-1}}$$

such that we have to identify the optimal parameters a and b. It can be shown that the mean-square output error for this type of models can be expressed as

$$MSOE = \sigma_d^2 - 2bH\left(a^{-1}\right) + \frac{b^2}{1-a^2}$$

From this expression we readily see that it is nonlinear in the parameters, and thus we expect the surface to have several extrema. This is the case indeed, as shown by the contour plot in figure 2.14, where bluer regions indicate minima. There is a global minima at $a = 0.906$, $b = -0.311$ and several local minima.

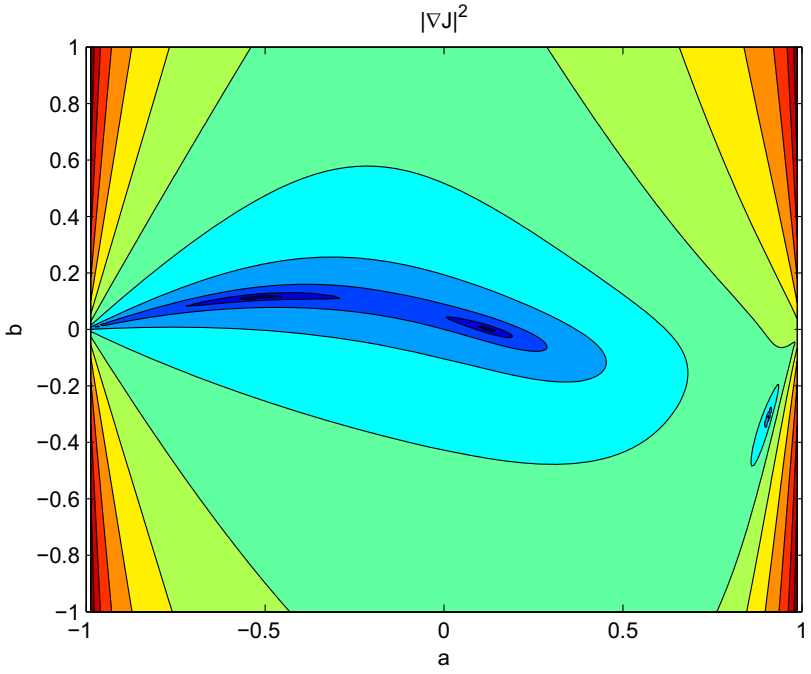
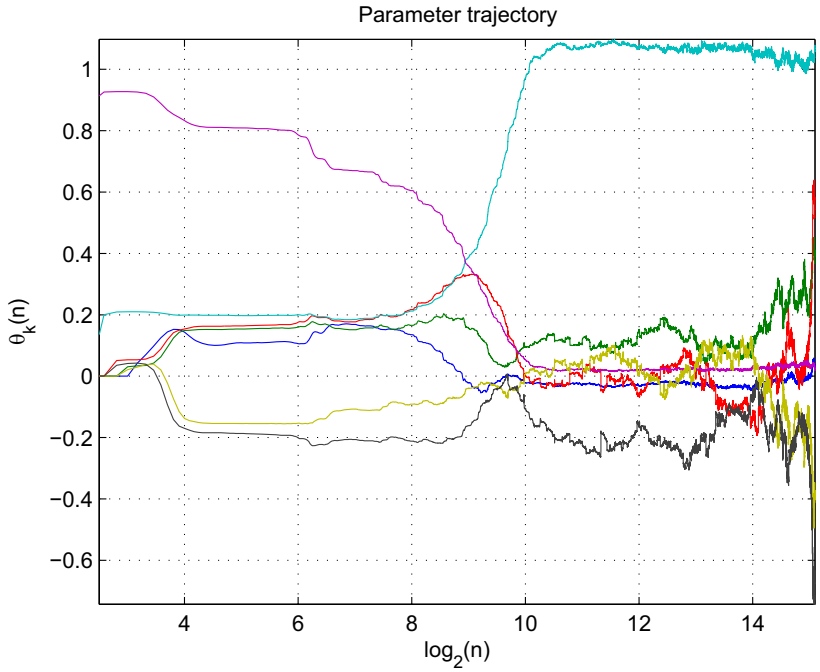


Figure 2.14: Contour plot of the IIR error-performance surface

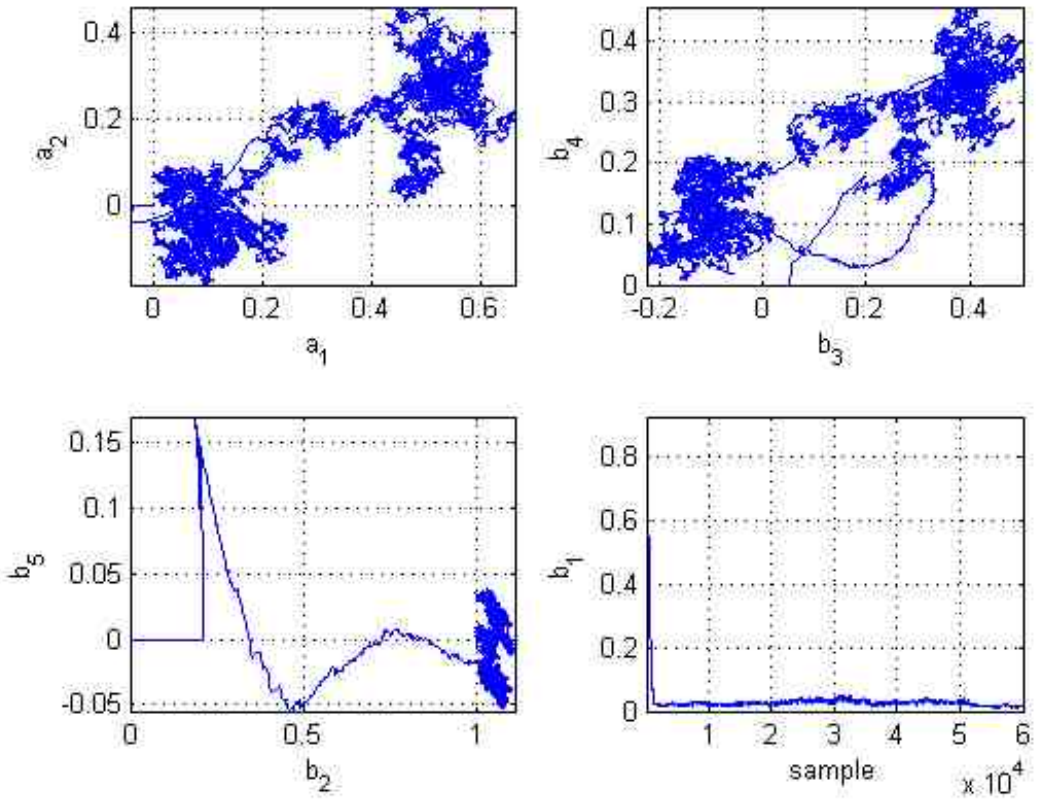
It is clear, that unlike the FIR estimation case, if one starts the adaptive filter with a parameter vector θ located near a minimum of the error-performance surface, one runs the risk of being trapped around that minimum for a long time, maybe *ad infinitum*. Moreover, an effect called breakdown is exhibited under certain conditions which renders adaptation non-unique for a given realization set. To illustrate this effect, consider the identification of an ARMA(3,5) process. There are a total of eight parameters to fit. Figure 2.15 shows the time evolution of the different parameters $\{\theta_k\}_{k=1\ldots8}$ as well as their pairwise evolution (i.e. the evolution in coefficient space, cut into bidimensional slices[10]). Notice how the

[10]Pairing is necessary since the coefficient space is isomorphic to $\mathbb{R}^8$ and therefore not representable in paper.

parameters tend to form clusters. These clusters are actually the parameter trajectories being stuck on a local minima for some number of iterations, then going to some other point until finding another local minima. During this travel to another minima, all the other parameters must vary to keep the error from increasing.



(a) Time evolution (time scale is base-2 logarithmic)



(b) Pairwise parameter evolution

Figure 2.15: ARMA(3,5) parameter evolution

A myriad of ARMA estimation methods exist. One such algorithm is explained in detail in [35], as well as the the details and reasons behind the Breakdown Effect.

2.5.6 Kalman Filtering

Kalman filtering refers to a technique used to estimate the dynamics of a system given a set of noisy measurements. For instances, it may be used to track a target in a radar: at each instant in time, the position, velocity and acceleration of an object can be measured but these measurements are usually corrupted by noise. On the other hand, there is a known relationship between the three:

$$\begin{aligned} a &= \dot{v} \\ v &= \dot{x} \end{aligned}$$

so that given measurements of the three magnitudes and these relations, one can try to cancel the noise inherent in the measurements.

Kalman filters are usually written as a state-space representation through two equations. The first equation governs the system dynamics:

$$x[k] = F[k]\,x[k-1] + B[k]\,u[k] + w[k]$$

where $x[k]$ is the state of the system at time k, $F[k]$ is a matrix modeling the dynamics of the system, which is applied to the previous state $x[k-1]$, $B[k]$ is the transfer function between the control input $u[k]$and the state and $w[k]$ is the process noise. Usually the noise is assumed to be zero-mean, multivariate Gaussian with a given (possibly time-varying) covariance matrix

$$w[k] \sim \mathcal{N}(0, Q[k])$$

The second equation describes the observable output:

$$z[k] = H[k]\,x[k] + v[k]$$

where $z[k]$ is our measured state, $H[k]$ a mapping from states to outputs, and $v[k]$ is the measurement noise, also assumed to be zero-mean and Gaussian

$$v[k] \sim \mathcal{N}(0, R[k])$$

Because of the fact that Kalman filters are order-1 recursive (only the current measurement and the immediately previous state are required to compute the current state of the system), and because of the impossibility to actually get the system state (we only have access to the system through observations), Kalman filters can be regarded as analogous to Hidden Markov Models. The only difference is in the fact that the hidden state variables are continuous in nature (as opposed to discrete in HMM). Also, HMM can represent arbitrary distributions for the next state variables, while the Kalman filter's noises are always Gaussian, mainly because of tractability issues.

More information can be found in [4, 5, 7, 22].

2.6 Other Techniques

Several methods and techniques can be used to perform non-linear time-variant mappings between input and output. One such technique uses neural networks ([41]), which allows an arbitrary mapping between the trajectories of feature vectors and vectors useful for the internal representation of data in an ASR.

Dynamic Programming is the name of a family of algorithms which uses a Divide-and-Conquer approach to solve problems, viz. that solutions to several subproblems can be combined together to render the solution to the global problem. This usually is connected with greedy algorithms.

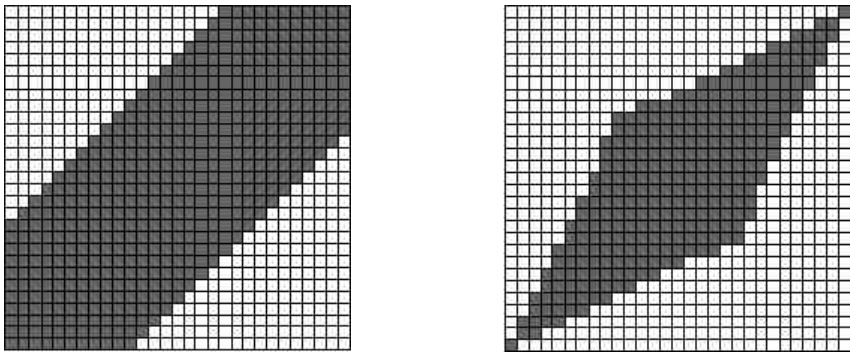
Dynamic Time Warping (DTW) is a Dynamic Programming technique that can be used to identify a distance between two feature-vector trajectories. In DTW, the problem is to find the time alignment between two signals so that the distance between the aligned signals is minimized. In other words, given the two signals $a[n]$ and $b[n]$, one tries to find monotonously-increasing mappings m_1 and m_2 such that the distance

$$d = \|a[m_1(n)] - b[m_2(n)]\|_1 \tag{2.17}$$

is minimized, while satisfying these constraints

$$\begin{aligned} m(0) &= 0 \\ m(n) &= n \\ m(n_1) &\geq m(n_2) \quad \forall n_1 \geq n_2 \end{aligned}$$

where m is either m_1 or m_2. The first and second conditions assume that the sequences are aligned at the start and end: we can usually relax these conditions by not taking into account the first and last few frames into the calculation of equation 2.17. The third equation, says



(a) Sakoe-Chuba band (b) Itakura parallelogram

Figure 2.16: Allowable regions for DTW

that time always flows in the "forward" direction, that is, we don't allow trajectories to go backwards for the sake of minimizing the distance. While in most cases this will make sense, we might want to reconsider this in scenarios where the time trajectories of the feature vectors might not be sequential in time. An example of this is when the feature vectors are transmitted over a data network that doesn't guarantee ordered packet arrival, such as a UDP or a Real-Time Transport Protocol (RTP) network connection.

Since the search space for the DTW algorithm can be quite big, we usually introduce additional restrictions on the warping function m_1 and m_2 by only allowing the trajectories to traverse a certain region, such as the ones shown in figure 2.16. More information on time warping can be found in [42].

Chapter 3

Speech Recognition Tools

In order to get a better understanding on how to improve the performance of ASR, we will study some of the tools an ASR uses.

3.1 Front End

The first component a signal encounters in its path through an ASR system is the front end. The front end's most important function is the extraction of signal characteristics relevant to recognition. In other words, the front end must transform the input signal into a representation that is at the same time meaningful for recognition purposes (i.e. similar content from the semantic point of view maps to a similar front end output), and manageable.

Since speech can be considered as having a bandwidth of approximately 4 kHz, and can be acceptably encoded at 12 bits/sample, it has a bandwidth of $B = 4 \times 10^3 \cdot 2 \cdot 12 = 96$ kbps. While this is certainly manageable for a computer, recognition algorithms involve searching a large space. If this search is to be performed on a sample-by-sample base, then the computational complexity becomes too much; if we want to perform recognition at higher sampling rates and higher bit resolutions, then the requirements are that much greater.

Finally, a time-domain representation is not appropriate for recognition purposes. That is why front ends usually work on a frame-by-frame basis, and for each of these frames, output a vector of features, like those described in previous sections.

More information can be found in [43, 44].

3.1.1 Filtering and Equalization

As mentioned, a signal undergoes certain processes throughout its path until it is received at the ASR. These processes introduce different kinds of distortion in the signal which hinder recognition. In order to palliate the effects of the channel, a number of techniques can be applied. In the rest of this discussion we will focus on linear time-invariant

systems, where the concept of a frequency-domain characterization of the channel makes sense.

All of these techniques ultimately try to equalize the signal, i.e. if a test signal such as white noise is fed through the channel, the received signal will have certain frequency-domain characteristics, viz. the noise will be colored by the spectrum of the channel[1]. By filtering the colored signal with an appropriate filter, we can try to whiten the spectrum of this noise signal to resemble the original spectrum. This filter is called a linear equalizer.

3.1.1.1 Linear Equalization

To obtain the equalizing filter, one probes the channel with a test signal $x[n]$ as just described, observes the output $d[n]$, and uses some sort of adaptive filter algorithm to adapt the filter coefficients to match the filtered output $y[n] = S\{d[n]\}$ to the channel input as close as possible, thus cancelling the channel distortion as much as possible. The structure is depicted in figure 3.1(a). In other words, one wishes to find

$$\mathcal{S} = \arg\min_{S} \|y[n] - d[n]\|$$

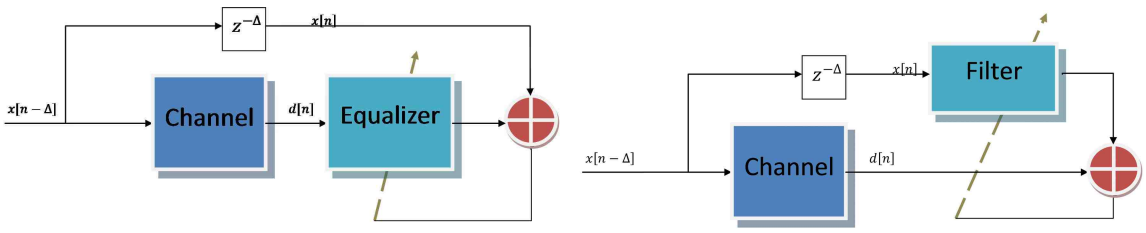
FIR filters are usually utilized for equalization, since they are always stable, and the algorithms used to find the optimal coefficients are well-behaved, mainly because the performance surface arising from the calculation of $J(S) = \|S\{x[n]\} - d[n]\|$ is unimodal (i.e. it has a unique minimum). Therefore, the mathematical statement of the problem of finding the optimal equalizer is

$$h_o = \arg\min_{h} \|x[n] - (h * d)[n]\|$$

Several comments are in order. First, the minimization is done over all filter coefficients for a **given** filter order. This filter order determines the lowest possible prediction error[2]. In general, a greater filter order for FIR filters will result in the same or lower prediction error. Secondly, the minimization is parametric in a time delay Δ. This delay is chosen so as to make the equalizer causal, and generally so that the peak of the impulse response occurs at the center of the tapped delay line. This way, the filter behaves as non-causal for a short time frame, because the output at time n will depend upon the input at time $n-1$, n, $n+1$, etc. The unconstrained, non-causal Wiener (optimal) filter for a given input x and output d

[1] Note that we may use a different test signal, as long as we know the frequency characteristics of this test signal (note that we need the signal to originate from a stationary process for the spectrum to make sense).

[2] By prediction error we mean the value of the cost function $J(h)$



(a) Inverse System Identification Configuration (b) Direct System Identification Configuration

Figure 3.1: Two equalization structures

satisfies

$$H_o(z) = \frac{\Phi_{xd}(z)}{\Phi_{xx}(z)}$$

where $H_o(z)$ is the z-transform of the filter's impulse response, $\Phi_{xy}(z)$ is the cross-PSD of x and d and $\Phi_{xx}(z)$ is the PSD of x. The filter $h_o[n]$ is therefore in general infinitely long, and non-causal. By truncating the impulse response and padding it, we obtain a causal, finite-length impulse response.

Another way to equalize the effects of the channel is depicted in figure 3.1(b). This configuration is known as system identification. When the filter is chosen to be an FIR as in the equalization configuration (also known as inverse system identification), and since one estimating the channel behavior rather than compensating for it, one obtains an all-pole equalizer.

$$h_o = \arg\min_h \|(h * x)[n] - d[n - \Delta]\|$$

More information can be found in [7, 39, 111, 112].

3.1.1.2 Cepstral Mean Subtraction (CMS)

Going back to the problem at hand, that is, eliminating the effects of a given channel, let us assume that the channel can be described as a moving average (FIR) filter:

$$y[n] = (h * x)[n]$$

By taking the cepstrum (section 2.1.4) of both sides of the equation we get

$$Y[k] = H[k] + X[k]$$

Now, we make the following observation: the channel is stationary, i.e. its characteristics will not change over time, within a certain (small) time frame. Therefore, if we

consider a frame-by-frame analysis, we get for the i[th] frame

$$Y_i[k] = H[k] + X_i[k]$$

where the index k is the quefrency bin. If we average this equation over all L frames, we get

$$\frac{1}{L}\sum_i Y_i[k] = H[k] + \frac{1}{L}\sum_i X_i[k]$$

so we now define

$$\begin{aligned} Z_i[k] &= Y_i[k] - \frac{1}{L}\sum_j Y_j[k] \\ &= H[k] + X_i[k] - \left(H[k] + \frac{1}{L}\sum_j X_j[k]\right) \\ &= X_i[k] - \frac{1}{L}\sum_j X_j[k] \end{aligned}$$

Note that removing the mean corresponds to highpass liftering the signal, thus removing the slowly varying components present in the signal. In particular, we would like to achieve infinite attenuation to the constant component, namely the channel. By using CMS, we fully remove the channel effects, **but** we introduce an artifact arising from the average cepstrum of the speech signal. In addition, we will always have an additive noise component so that

$$\begin{aligned} y[n] &= (h * x)[n] + v[n] \\ Y[k] &= H[k]X[k] + V[k] \\ \log Y[k] &= \log\left(X[k]\left(H[k] + \frac{V[k]}{X[k]}\right)\right) \\ &= \log X[k] + \log\left(H[k] + \frac{V[k]}{X[k]}\right) \end{aligned}$$

If the noise has less energy than the signal itself we can write $\log(x_0 + \delta) \approx \log x_0 + \frac{\delta}{x_0}$:

$$\begin{aligned}
\log Y[k] &\approx \log X[k] + \log H[k] + \frac{V[k]}{X[k]} \\
\hat{Y}[n] &\approx \hat{X}[n] + \hat{H}[n] + \mathbf{FFT}\left\{\frac{V[k]}{X[k]}\right\} \\
\left\langle \hat{Y}[n] \right\rangle &\approx \underbrace{\left\langle \hat{X}[n] \right\rangle}_{\text{signal artifact}} + \hat{H}[n] + \underbrace{\left\langle \mathbf{FFT}\left\{\frac{V[k]}{X[k]}\right\} \right\rangle}_{\text{deconvolved noise artifact}}
\end{aligned}$$

So as a result of CMS, on top of an artificial component from averaging the input cepstrum, we also have a component arising from averaging the noise deconvolved by the input. In mid-to-low SNR scenarios, the second foreign component can become large enough to make the whole channel estimate unreliable.

3.1.1.3 Dynamic CMS

The approach described in section 3.1.1.2 requires averaging the incoming signal over L frames, where L is a number of frames large enough to make the speech average cepstrum vanish. Usually this will make real-time processing impossible. However, by computing a running average, we can approximate this mean subtraction. This approach is called dynamic Cepstral mean normalization.

Let $\alpha \to 0^+$ and the average cepstrum and its respective z-transform be

$$\begin{aligned}
M_i[k] &= \alpha X_i[k] + (1-\alpha) M_{i-1}[k] \\
\mathcal{M}(z,k] &= \alpha \mathcal{X}(z,k] + (1-\alpha) z^{-1} \mathcal{M}(z,k] \\
\mathcal{M}(z,k] &= \frac{\alpha}{1-(1-\alpha)z^{-1}} \mathcal{X}(z,k]
\end{aligned}$$

and so we can calculate a running average

$$\begin{aligned}
\mathcal{Z}(z,k] &= \left(1 - \frac{\alpha}{1-(1-\alpha)z^{-1}}\right) \mathcal{X}(z,k] \\
&= (1-\alpha)\frac{1-z^{-1}}{1-(1-\alpha)z^{-1}} \mathcal{X}(z,k] \\
&= \beta \frac{1-z^{-1}}{1-\beta z^{-1}} \mathcal{X}(z,k]
\end{aligned}$$

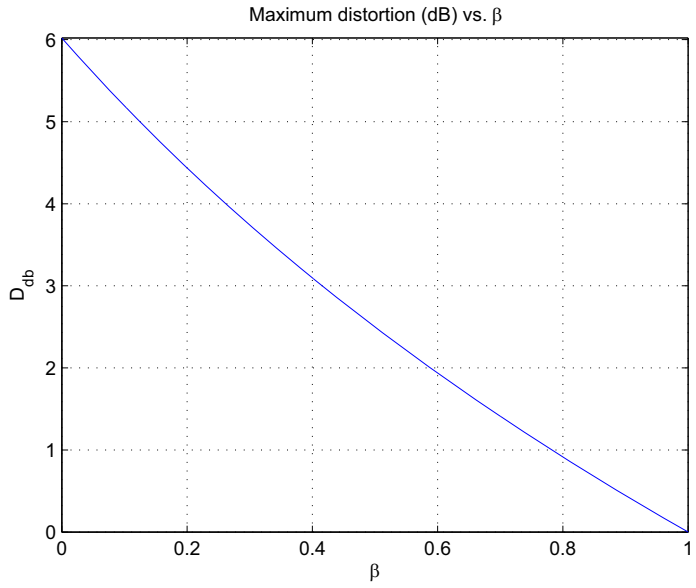


Figure 3.2: Maximum cepstral distortion in Dynamic CMS

where $\beta \rightarrow 1^{-}$. This corresponds to a filter with a zero at $\omega = 0$ and a damped pole at $\omega = 0$ too.

Note that the running average will not be very accurate at the beginning, because of the transient arising from this filter. The choice of β determines the time delay after which the mean subtraction can be considered accurate, and the distortion introduced at frequencies other than DC. It can be shown that the maximum of the magnitude response is attained at $\omega = -\ln(\beta)$, having a value of

$$\begin{aligned} D &= 4\frac{(\beta-1)^2}{(\beta^2-1)^2} \\ &= \frac{4}{(\beta+1)^2} \\ D_{\text{dB}} &= 10\log_{10}\frac{4}{(\beta+1)^2} \\ &= 20\log_{10}2 - 20\log_{10}(1+\beta) \\ &\approx 6\text{ dB} - 20\log_{10}(1+\beta) \end{aligned}$$

So, the larger β is, the smaller the maximum distortion will be (figure 3.2).

More information can be found on [79, 80].

3.1.1.4 Matched Filter

For the purpose of detection, a procedure called matched filtering may be used. This consists of a linear filter, whose impulse response is convolved with the input. In the presence of additive white noise at the input, the matched filter achieves the maximum signal-to-noise ratio at the output. To construct a matched filter, we time-reverse a template of the signal we are trying to detect. For a given noise covariance matrix R_v and template signal s, we get that the normalized matched filter is given by

$$h\left[n\right] = \frac{1}{\sqrt{s^H R_v^{-1} s}} R_v^{-1} s^H\left[N-n\right]$$

If we deal with white noise, we get $R_v = \sigma_v^2 I$ so that

$$\begin{aligned} h &= \frac{s^{BH}}{\sigma_v \sqrt{s^H s}} \\ &= \frac{1}{\sigma_v \sigma_s} s^{BH} \end{aligned}$$

where $\cdot^B$ is the backwards operator. Thus, if we have templates of all the signals of interest for detection, we might construct a bank of matched filters. Note that this would not be realizable in practice due to variabilities in one speaker (speed of speech, articulation, etc.) as well as variations among speakers.

3.1.2 Other Speech Intelligibility Enhancers

Speech enhancers are a wide class of algorithms whose intent is to increase the perceptual quality of speech data. Some general common characteristics are:

1. They usually operate on a frame-by-frame basis
2. They usually are non-linear operations, i.e. the filter cannot be expressed as an input-independent impulse response convolved with the input.
3. They are based on one or more of these models

    (a) acoustic (e.g. physical modeling of the vocal production system, noise source modeling)

(b) psychoacoustical, i.e. how the human ear perceives sound, how the stimulus is transmitted through the human hearing system into the brain, why certain frequencies are amplified, why certain sounds are masked, etc.

(c) statistical: a model that covers either long-term averages (time average), or a short-time average taken over many realization of the same process (ensemble average). Processes in which the time average and the ensemble average of a certain magnitude coincide are called ergodic

4. As the name implies, their objective is to improve the intelligibility of the speech segment. In order to evaluate their effectiveness, we have two basic options. The first one is to conduct a subjective test by letting a large enough number of people rate the intelligibility of the speech segment after being processed, compared to that of the untreated segment, on a numerical scale and then average the answers. This process is called Mean Opinion Score (MOS) or we can use a standardized algorithm such as PESQ ([120, 121]) to evaluate the perceptual quality. These algorithms also rely on psychoacoustical models, and are suitable for only certain kinds of problems. For instances, PESQ converts the input audio to a telephone-like intermediate representation. Thus, if we are trying to simulate any kind of channel, especially a phone channel, it would not be accurate to use PESQ as a measure of success of our simulation, since the algorithm is inherently discarding information. Secondly, PESQ is a perceptual measurement which tries to mimic MOS, while our simulations will not necessarily measure perceptual qualities (that is, human perception), but rather the perception of the ASR engine.

To illustrate the idea, consider the algorithm described in [82]. The algorithm's objective is basically to estimate the fundamental frequency on each frame, and then lower the magnitude of the frequencies that are most dissonant with respect to the fundamental (figure 3.3). Therefore, we have that the algorithm relies on how the ear perceives sounds as consonant or dissonant (psychoacoustical model).

It is important to note that while humans may perceive an "enhanced" segment as more intelligible than an untreated segment, recognition rates for those enhanced segments are usually not better ([83]). This is mainly because they either discard information from the signal or introduce artificial information, thus distorting the whole feature-space trajectory

on which the recognition process is based. Moreover, phase information may be changed in a way that actually worsens the recognition rate[3].

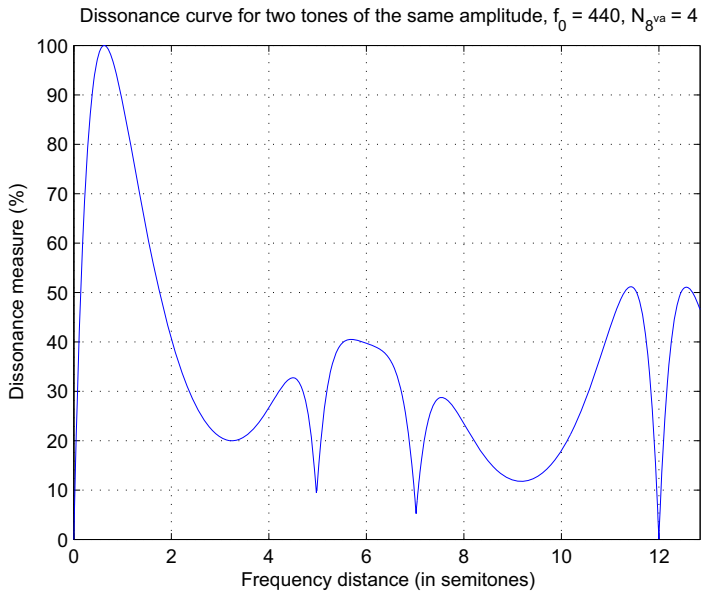


Figure 3.3: Dissonance curve for two tones of equal volume

More information can be found on [82, 91, 92, 93, 94, 95].

3.1.3 Energy Normalization

For recognition purposes, usually frames are energy normalized so that all incoming frames to the rest of the system have the same energy. This takes care of inherent volume variabilities between louder and softer parts of a sentence.

3.2 Voice Activity Detection (VAD)

In many scenarios it is crucial to be able to determine with great accuracy the beginning and ending of each word in a sentence. For example, in order to avoid feeding the system with ambient noise to recognize. Another example, which we will see in more detail in section 5.6.1, is when trying to estimate the filter that gives the closest output to a given desired output, given an input. For that particular application, it is important that input and output are synchronized. Otherwise, the estimated filter will either contain unnecessary delays (if the desired output has silence before the actual data), or fail to capture all of the channel characteristics if the input has more silence before than the desired output, since

[3]Of course this statement is true only for ASRs using features that depend on the signal's phase.

in that case the filter will have a strong anticausal component which in general is more difficult to estimate correctly.

A Voice Activity Detector classifies each frame in an audio stream as speech (S) or silence (N). To be able to synchronize and pinpoint beginnings and endings of words, there are a many methods being used. VAD systems are usually complex classifying systems, involving one or more features. The simplest one would be to use an energy threshold and if the energy within a frame exceeds that threshold, mark it as speech. This method is clearly faulty in that it doesn't contemplate differences between softer and louder speakers. Even with only one speaker, it fails to capture language dynamics which mandate certain phonemes should be stressed more than others. It also is faulted in that there are some phonemes which inherently carry less energy than others (vowels are much more energy-bearing than stops). Apart from these limitations, words tend to last for more than a few frames. Therefore, after the VAD classifies each frame as S/N, it must do a second pass eliminating spurious detections: e.g. a frame might have been classified as S because it was an aspiration before the actual word, which can carry a lot of energy.

Another decision feature might be the energy differential between current and previous frames. Note that if we start introducing information from previous frames into our decision, we are inherently introducing delays in the system, which may translate to not being able to exactly detect the onset of the first and last phonemes in a sentence.

Finally, it is interesting to note that spectral entropy can be used as a successful decision feature. Spectral entropy is calculated on a frame by frame basis, and it is defined as the information entropy of a random variable that has a probability density function given by the amplitude spectrum of the frame, normalized to unity.

Techniques such as GMM have been used as a decision algorithm, where mixtures are trained for both speech and silence and when a new frame comes in, it is compared against the mixtures to decide either S or N.

The performance of binary classifiers such as VAD algorithms can be summarized graphically in a Receiver Operating Characteristic curve. The ROC is a graphical representation of the algorithms sensitivity vs. specificity (i.e. a plot of the number of true positives (TPR) vs. false positives (FPR)). A true positive occurs when the segment was classified as S and it really is speech, while a false positive occurs when the segment was classified as S when it really is N. Figure 3.4 shows a typical ROC curve along with the so called line of no discrimination, which depicts the characteristic of an algorithm which

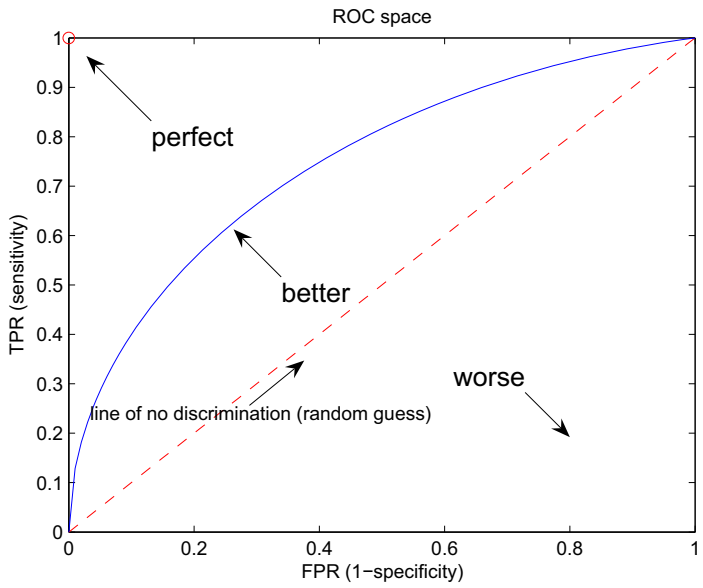


Figure 3.4: A ROC curve

assigns values randomly. An ideal perfect classifier would be represented with a point at $(0, 1)$, which means that we always get 100% true positives and 0% false positives.

3.3 Gender Detection

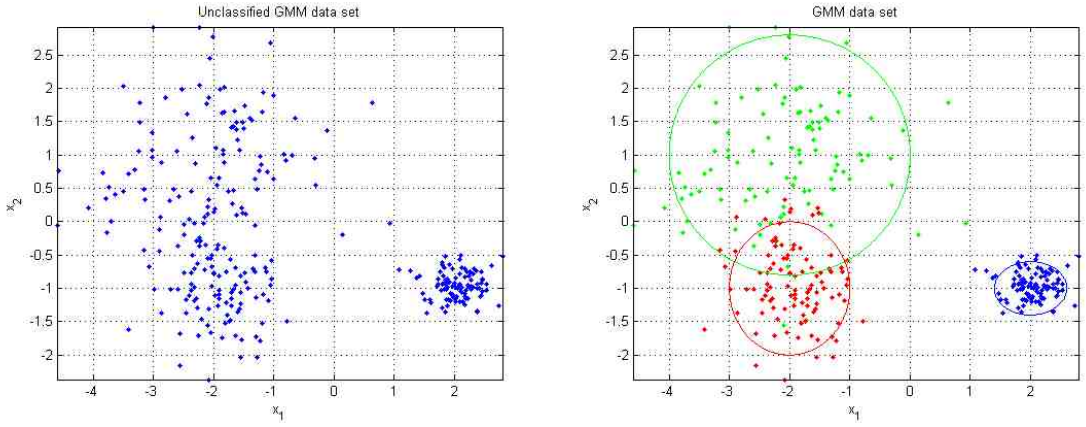
If the system can detect the speaker's gender (i.e. male, female, child), then it will most likely perform better since the features and filters used for recognition can be customized to each gender. The problem is that it is generally difficult to assert the speaker's genre based on a small number of audio frames, and since users usually expect the recognition process not to need any additional training, this splitting of the system into customized parts might not be applicable for real-time purposes.

3.4 Prosody

Prosody describes the rhythmic and musical attributes of a language. It includes the study of intonation, pitch, formant structure and similar attributes. Some languages are more prosodic than others (cf. Mandarin vs. English). Therefore, depending on the language, it may be necessary to include and analyze those pieces of information and incorporate them into the detection process.

3.5 Gaussian Mixture Models (GMM)

Gaussian mixture models are a method to "learn" about characteristics and trends within a data set. They can be used to classify the points in a data set into several categories, a



(a) Unclustered data (b) Clustered data

Figure 3.5: An example of a data set originating from three independent Gaussian distributions, and clustered by three Gaussian mixtures

process known as data clustering. Consider the two-dimensional data set in figure 3.5. To train the mixture model, we are given the green, red and blue points, but we aren't given their

To train the model means to find all of the distributions' parameters, in this case the gaussian's means μ_i and variances σ_i. To find them, we must choose those parameters which maximize the likelihood (this approach is called Neyman's Maximum Likelihood principle). To do that we have three options:

1. Set $\frac{\partial}{\partial \mu_i} \log P\left(\underline{\mathrm{x}}|\mu_1, \ldots \mu_N\right) = 0$ and solve for all the μ_i. This gives rise to difficult to solve equations.

2. Use steepest descent or some other method to find the optimum. This is a very slow approach.

3. Use the Expectation Maximization algorithm. This is the approach used in practice.

3.5.1 Expectation Maximization (EM)

Suppose we already know the probability of each class w_j, but we don't know the μ_i. Then the probability of the data given the means μ_i is

$$\begin{aligned} P\left(\underline{x}|\underline{\mu}\right) &= \prod_{i=1}^{N} P\left(x_i|\underline{\mu}\right) \\ &= \prod_{i=1}^{N}\sum_{j=1}^{k} P\left(x_i|w_j,\underline{\mu}\right) P\left(w_j\right) \\ &= \prod_{i=1}^{N}\sum_{j=1}^{k} K_j \exp\left(-\frac{1}{2\sigma_j^2}\left(x_i-\mu_j\right)^2\right) P\left(w_j\right) \end{aligned}$$

after some algebra we get that for maximum likelihood (i.e. solving for the derivative to be zero),

$$\mu_j = \frac{\sum_{i=1}^{N} P\left(w_j|x_i,\underline{\mu}\right) x_i}{\sum_{i=1}^{N} P\left(w_j|x_i,\underline{\mu}\right)} \tag{3.1}$$

These are nonlinear equations in μ_i. If, for each x_i we knew the probability that μ_j was in class w_j, i.e. $P\left(w_j|x_i,\underline{\mu}\right)$, for each class w_j, then we can straightforwardly compute μ_j. On the other hand, if we knew each μ_j then we can easily compute $P\left(w_j|x_i,\underline{\mu}\right)$ for each x_i and w_j (just use the Gaussian pdf). The alternation of both steps is called the Expectation Maximization algorithm applied to the estimation of the class's means and is described in algorithm 5.

Algorithm 5 Expectation Maximization for GMM means

On the t^{th} iteration our estimates are $M_t = \{\mu_1(t), \ldots, \mu_k(t)\}$

E-step Compute the expected classes of all data points for each class:

$$\begin{aligned} P\left(w_j|x_i,M_t\right) &= \frac{P\left(x_i|w_j,M_t\right)}{P\left(x_i|M_t\right)} \\ &= \frac{P\left(x_i|w_j,\mu_j(t),\sigma_j^2\right) p_j(t)}{\sum_{j=1}^{k} P\left(x_i|w_j,\mu_j(t),\sigma_j^2\right) p_j(t)} \end{aligned}$$

M-step Compute the maximum-likelihood $\underline{\mu}$given the last step membership distributions:

$$\mu_j(t+1) = \frac{\sum_{k=1}^{N} P\left(w_j|x_k,M_t\right) x_k}{\sum_{k=1}^{N} P\left(w_j|x_k,M_t\right)}$$

Alternate between *E* and *M* steps

- Because of the non-linear dependence on the parameters, the EM algorithm can get stuck in local minima, and empirically it does.
- In the same way we described a very simple algorithm to estimate the mixture's means, the algorithm can be applied *mutatis mutandis* to the estimation of σ_j^2.
- Note that EM works for a generic probability distribution. Gaussians are chosen for several reasons: they appear ubiquitously, mainly because of the Central Limit Theorem[4], and moreover the resulting EM equations are tractable or solvable.

After training (i.e. estimation of the mixture's parameters), whenever a new data point arrives, we must decide to which class (Gaussian) we assign the point. For that purpose, we calculate the likelihood that the point belongs to each class, and then choose the class that maximizes that likelihood. More information can be found in [107].

3.6 Recognition Performance Measures

We have mentioned the distinction between two sets of data to be used in ASR systems. On one hand, we have the training data set, which comprises the corpus of speech data based on which the system parameters (HMM parameters) are to be tuned. On the other hand, the test data set, or production data set, is the set of data upon which we will perform the recognition process. The performance of an ASR system can be characterized by the rate of successful recognition attempts, called the recognition rate, expressed as

$$R = 100\% \times \frac{\text{\# correctly recognized units}}{\text{\# tested units}}$$

where units may correspond to several levels of granularity. For example one may measure the recognition rate at a phrase level, word level, or even at a phoneme level. Because of articulation effects, and that the algorithms involved in recognition take into account the time evolution of the data, the recognition rates are generally different for different granularity levels.

Also, recognition rates may be split between significative groups, i.e. males and females, native vs. non-native speakers, and so on. This allows the system developer to

[4]The CLT states that if we average many random variables whose distributions are arbitrary (save for the fact that they have finite variance), then the resulting random variable will be distributed normally.

identify possible areas for improvement of the system; e.g. the system's performance may benefit from a gender classifier as described in section 3.3.

3.7 Distributed Speech Recognition

The idea behind Distributed Speech Recognition is to decouple the front-end processing from the rest recognition mechanism by using a client-server model over a data network. This way one can use the front-end processing on a lightweight device such as a PDA and access speech information retrieval services using a back-end recognition server. The data network is not restricted to be wire-line, e.g. GPRS[5] data-enables mobile networks making these services available to GPRS-enabled devices, like tomorrow's mobile phones. To accomplish this, the stream of acoustic information transmitted from the client to the server needs to be compacted as much as possible. Speech coding schemes designed specifically for recognition purposes can have data rates as low as 2 Kbps. More information can be found in [108, 109].

[5]General Packet Radio Service (GPRS) is a mobile data service available to users of Global System for Mobile Communications (GSM) and IS-136 mobile phones.

Chapter 4

Theoretical Framework

In this chapter we introduce a mathematical definition of the problem at hand. To generate channel data (i.e. simulated data) from our clean recordings, we will use a certain digital filter, whose structure and parameters are *a priori* unknown. Given a certain clean data set and its corresponding channel data set, we might calculate what the optimal filter coefficients are, in order to meet a certain criteria. This criteria will be to minimize the distance in some space between the channel data and the simulated data.

Therefore, we have several free variables:

- filter structure
- filter coefficients/parameters
- space in which the distance is measured
- choice of distance

4.1 Filter structure

For our purposes, and as a first approach, we will be concerned only with linear, time-invariant filters. For simplicity's sake, we will concern ourselves with FIR filters, since IIR filters can almost always be correctly approximated by sufficiently long FIR filters[1].

[1] the case in which this statement is false is when very sharp resonances or unstable poles occur in the IIR filter, which is of no interest to us anyway.

4.2 Filter Coefficients and Parameters

Since we have decided to use LTI FIR filters, the only parameters to choose are the length of the impulse response, and its sample values (or conversely, the magnitude and phase responses).

4.3 Measure Space

Since our goal ultimately is to feed the ASR system with data that "looks" as if it came from a channel, the measure space must coincide with the space in which the ASR system parametrizes its input. Put differently, the ASR sees a certain slice of reality, drawn from its input information, in a certain feature space. If the ASR system cannot tell the difference between simulated and actual channel data in that feature space, we will have attained our goal.

4.4 Norm

There are a number of interesting norms to consider. First, let's look at the quadratic norm, also known as the 2-norm. For our purposes, given a vector v in the measure space with components v_i, the squared 2-norm is defined as

$$\|v\|_2^2 = \sum_{i=1}^{N} |v_i|^2$$

First and foremost, it is a very tractable norm, whose (square's) derivative renders a linear equation which facilitates solving for the optimum. Secondly, there already are several well-established methods to do 2-norm minimization as we have already seen in section 2.5. Performance surfaces are bowl-shaped manifolds, which behave nicely because of having a unique optima. Finally, we will be heavily using and relying on Parseval's identity (equation 2.1) which is an identity relating 2-norms in a primal space and its dual space.

Depending on the choice of features, some features may actually carry more information than others. Thus, an error in that feature must contribute more to the error norm. We borrow the concept of entropy from Information Theory to get a measure of how much information a feature carries:

4.4.1 Feature Entropy

Consider a feature vector trajectory $v[k] = \{v_i[k]\}_{i=1\ldots N}$. We treat each of the N features as distinct random variables, and try to estimate the entropy for each of these r.v. For that purpose, we have basically two options: try to estimate the r.v. probability distribution (pdf) by some well-known method (a simple example of how to do this is to

produce a histogram), or calculate the spectral entropy of the feature. Let

$$V_i[t] = |\mathbf{FFT}\{v_i[k]\}[t]|$$

In general, features can be complex valued. When a feature is real valued, we can simplify the calculations by letting

$$V_i[t] = \left|\mathbf{FFT}^+\{v_i[k]\}[t]\right|$$

where the plus sign indicates we only consider positive frequency bins (since an FFT is symmetric for real valued inputs). Define the normalized spectral content as

$$V_i^N[t] = \frac{V_i[t]}{\sum V_i[t]}$$

such that V_n adds up to unity. We define the spectral entropy as

$$\eta_i = -\sum V_i^N \log_2 V_i^N$$

which can be made less sensitive to noise and quantization by damping by letting

$$V_i^N[t] = \frac{V_i[t]}{\sum V_i[t]} + \delta$$

for a very small δ. Therefore, for each feature in a trajectory, we obtain a measure of how many bits would be necessary to represent that feature. The more variability a feature's spectrum has, the more "random" it is, and therefore, its entropy will be greater. The flatter the feature's spectrum (i.e. the feature is very localized), the smaller its entropy. Hence, we can now compare how much information a feature carries. Based upon this, we may want to weigh different features differently (give more weight to features that vary less, since a difference in that feature is more noticeable; put differently, it is not expected for a very localized feature to move around a lot because of an error in that feature, while features that vary a lot aren't as affected by a small error). More information can be found in [110].

For the purpose of weighing different features differently, we can introduce what is called an anisotropic norm. The standard 2-norm is isotropic, in the sense that every component's energy is multiplied by the same number (one) and then added. An anisotropic norm can be written as

$$\|v\|_{2,\text{anisotropic}}^2 = \sum_{i=1}^{N} \alpha_i |v_i|^2$$

4.4.2 Anisotropic, Coupled 2-norm

Because of coupling between features (interdependence, as described in 2.1.2.2 and 2.1.2.3), we may want to generalize this notion by introducing a square matrix into the equation which weighs not only each feature squared by itself but also the mutual products between features:

$$\|v\|_{2|M}^2 = v^H M v \tag{4.1}$$

The requirements imposed upon the weight matrix M are basically that the matrix be positive definite, i.e. Hermitian ($M^H = M$) and all its eigenvalues are positive. In light of the definition in equation 4.1, one can now view the 2-norm as merely $\|v\|_{2|I}$. A geometrical interpretation can be given as follows: because of the Spectral Theorem, every Hermitian matrix can be diagonalized by a unitary matrix so that

$$\begin{cases} M & = & Q^H D Q \\ Q^H Q & = & I \\ Q Q^H & = & I \\ D & = & \operatorname{diag}\{d_i\} \end{cases}$$

Therefore, the anisotropic 2-norm is rewritten as

$$\begin{aligned} \|v\|_{2|M}^2 &= v^H Q^H D Q v \\ &= (Qv)^H D (Qv) \\ &= w^H D w \end{aligned}$$

So this is basically saying that the anisotropic 2-norm, looks at the vectors in a different coordinate basis (an orthogonal basis for that matter, corresponding to the eigenvectors of M), and weighs each component by a number in the diagonal of the matrix D. In other words, the 2-norm decomposes vectors along its eigenvectors and weighs these decompositions by the associated eigenvalues.

4.5 Methods and Techniques

4.5.1 Estimation

Adaptive filters represent the best approach to solving our estimation problem. They don't need dedicated or special hardware, they can be ran offline on a standard PC, a crucial characteristic. Also, they provide results which can be controlled and verified.

As to the objective function the filters try to minimize, there are some points to address. LMS filters look to minimize the expected value of the error energy at any time, i.e. $\min \mathbb{E}\left\{|e\left[n\right]|^2\right\}$. On the other hand, RLS filters take into account the past history of the error for the objective function, since they try to perform $\min \sum_{i=0}^{n-1} \lambda^{n-i} \mathbb{E}\left\{|e\left[n\right]|^2\right\}$. RLS filters converge very rapidly, but as we already said are more computationally expensive. Notwithstanding, we will pursue an RLS filter for most of our discussion.

A third and crucial point to consider is that RLS filters try to minimize the energy of the error in the time domain, i.e. the time-domain signal arising from a difference between real and simulated channels. While this is not what we are looking for, we will show how the two approaches are related and the advantages and disadvantages one has over the other.

For this purpose, we start by analyzing the difference between minimizing the error in the time domain as opposed to the frequency domain, and then try to extend that result to other spaces, especially to the cepstral space. The result, presented in section 4.7, is novel to the extent of which the author is aware.

4.5.2 Result Combination

Since we will be extracting results from many utterances, we then need to combine these partial results into one final representation. For that matter, consider the problem of, given $\{x_1, \ldots, x_N\}$, calculate $\hat{x}$ so that $\sigma^2 = \frac{1}{N}\sum \left(x_i - \hat{x}\right)^2$ is minimized (this is a condition known as minimum variance)

$$\begin{aligned}
\sigma^2 &= \frac{1}{N}\sum_{i=1}^{N} x_i^2 - 2x_i\hat{x} + \hat{x}^2 \\
\frac{\partial\sigma^2}{\partial\hat{x}} &= \frac{1}{N}\sum_{i=1}^{N} -2x_i + 2\hat{x} \\
&= -\frac{2}{N}\left(\sum_{i=1}^{N} x_i - N\hat{x}\right)
\end{aligned}$$

To reach the optimum, we need to set the derivative of the variance with respect to $\hat{x}$ to zero. This translates to

$$\sum_{i=1}^{N} x_i = N\hat{x}$$

or

$$\hat{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

which is the well-known definition of an average. Therefore, we have proven that the arithmetic average minimizes the variance of the sample data if the sample data is real valued. Now that we have this information, we proceed to generalize our result to complex valued data (since transfer functions will usually be complex). For that purpose we separate our data $\{z_i\}_{i=1...N}$ in its Cartesian decomposition, real and imaginary parts:

$$\begin{aligned} \sigma^2 &= \frac{1}{N}\sum |x_i + jy_i - \hat{x} - j\hat{y}|^2 \\ &= \frac{1}{N}\sum (x_i - \hat{x})^2 + (y_i - \hat{y})^2 \end{aligned}$$

The subtlety here is to notice that $\hat{x}$and $\hat{y}$ are independent, so we can treat the estimation of the averages for both components as independent, and get

$$\begin{aligned} \hat{x} &= \frac{1}{N}\sum_{i=1}^{N} x_i \\ \hat{y} &= \frac{1}{N}\sum_{i=1}^{N} y_i \end{aligned}$$

or more generally,

$$\hat{z} = \frac{1}{N}\sum_{i=1}^{N} z_i \tag{4.2}$$

4.6 Time- and Frequency-Domain Error Minimization Equivalence

In this section, we prove the equivalence between estimating an impulse response and estimating its transfer function, or what is the same minimizing $\mathbb{E}\left\{|e\left[n\right]|^2\right\}$ or $\mathbb{E}\left\{|\mathbf{FFT}\left\{e\left[n\right]\right\}\left[k\right]|^2\right\}$

Equivalence between impulse response error minimization and frequency response error minimization is quite straightforward

$$\begin{aligned} h^* &= \arg\min_{S_h} \|h - h_0\| \\ H^* &= \arg\min_{S_H} \|H - H_0\| \end{aligned}$$

We know that

1. The 2-norm in the frequency domain is the same as in the time domain (save a scaling factor), i.e.$\|\cdot\|_f \sim \|\cdot\|_t$
2. Linearity of FFT

So we prove that $H^* = \text{FFT}\{h^*\}$

$$\begin{aligned} H^* &= \arg\min_{H\in S_H} \|H - H_0\| \\ &= \text{FFT}\left\{\arg\min_{h\in S_h} \|\text{FFT}\{h\} - \text{FFT}\{h_0\}\|\right\} \\ &= \text{FFT}\left\{\arg\min_{h\in S_h} \|\text{FFT}\{h - h_0\}\|\right\} \\ &= \text{FFT}\left\{\arg\min_{h\in S_h} \|h - h_0\|\right\} \\ &= \text{FFT}\{h^*\} \end{aligned}$$

4.7 Cepstrum- and Frequency-Domain Error Minimization

Consider the problem of finding a FIR filter h_0 which, when applied to the input, renders a signal whose feature vector is closest to the feature vector of the output of the system. In this section we are concerned with the feature map $\mathcal{M}$ being just the cepstrum of the signal. We will try then to generalize the procedure to other features.

Given two N-dimensional vectors (signals) x and y , we define two operations: the dot product, $p = x\ y$ as

$$p[n] \stackrel{\text{def}}{=} x[n] \cdot y[n]$$

and the dot quotient as $q = \frac{x}{y}$ such that

$$q[n] \stackrel{\text{def}}{=} \frac{x[n]}{y[n]}$$

when $y[n] \neq 0$, and as $a[n] = \operatorname{sign} x[n]\infty$ when $y[n] = 0$. Then, we try to find the optimal filter given by

$$h_o = \arg\min_h \|\mathcal{M}\{(h * x)[n]\} - \mathcal{M}\{d[n]\}\|_2$$

$$\begin{aligned}
\mathcal{M}\{f[n]\} &= \mathbf{IFFT}\{\log \mathbf{FFT}\{f[n]\}\} \\
\mathcal{M}\{f[n]\} - \mathcal{M}\{d[n]\} &= \mathbf{IFFT}\{\log \mathbf{FFT}\{f[n]\}\} - \mathbf{IFFT}\{\log \mathbf{FFT}\{d[n]\}\} \\
&= \mathbf{IFFT}\{\log \mathbf{FFT}\{f[n]\} - \log \mathbf{FFT}\{d[n]\}\} \\
&= \mathbf{IFFT}\left\{\log \frac{\mathbf{FFT}\{f[n]\}}{\mathbf{FFT}\{d[n]\}}\right\}
\end{aligned}$$

Applying Parseval's identity we have that

$$\sum_{i=0}^{N-1} |f[i]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |\mathbf{FFT}\{f[i]\}[k]|^2 \tag{4.3}$$

or equivalently

$$\|f\|_2^2 = \frac{1}{N} \|\mathbf{FFT}\{f\}\|_2^2 \tag{4.4}$$

Note that while equation 4.3 expresses the equality in terms of the sample values, equation 4.4 is an equation in norms (2-norm in this case).

Next, we form the feature error vector as

$$\begin{aligned}
\mathcal{M}\{f[n]\} - \mathcal{M}\{d[n]\} &= \mathbf{IFFT}\left\{\log \frac{\mathbf{FFT}\{f[n]\}}{\mathbf{FFT}\{d[n]\}}\right\} \\
\mathbf{FFT}\{\mathcal{M}\{f[n]\}\} - \mathbf{FFT}\{\mathcal{M}\{d[n]\}\} &= \log \frac{\mathbf{FFT}\{f[n]\}}{\mathbf{FFT}\{d[n]\}} \\
\|\mathbf{FFT}\{\mathcal{M}\{f[n]\} - \mathcal{M}\{d[n]\}\}\|_2^2 &= N\|\mathcal{M}\{f[n]\} - \mathcal{M}\{d[n]\}\|_2^2 \\
&= \left\|\log \frac{\mathbf{FFT}\{f[n]\}}{\mathbf{FFT}\{d[n]\}}\right\|_2^2
\end{aligned}$$

$$\begin{aligned}
\|\mathcal{M}\{f[n]\} - \mathcal{M}\{d[n]\}\|_2^2 &= \frac{1}{N}\left\|\log \frac{\mathbf{FFT}\{f[n]\}}{\mathbf{FFT}\{d[n]\}}\right\|_2^2 \\
\|\mathcal{M}\{f[n]\} - \mathcal{M}\{d[n]\}\|_2 &= \frac{1}{\sqrt{N}}\left\|\log \frac{\mathbf{FFT}\{f[n]\}}{\mathbf{FFT}\{d[n]\}}\right\|_2
\end{aligned}$$

$$f[n] \stackrel{\text{def}}{=} (h * x)[n]$$

$$\begin{aligned} \|\mathcal{M}\{f[n]\} - \mathcal{M}\{d[n]\}\|_2 &= \frac{1}{\sqrt{N}} \left\| \log \frac{\mathbf{FFT}\{h[n]\}\,\mathbf{FFT}\{x[n]\}}{\mathbf{FFT}\{d[n]\}} \right\|_2 \\ \epsilon &\stackrel{\text{def}}{=} \|\mathcal{M}\{f[n]\} - \mathcal{M}\{d[n]\}\|_2 \end{aligned}$$

$$\begin{aligned} X[k] &\stackrel{\text{def}}{=} \mathbf{FFT}\{x[n]\}[k] \\ D[k] &\stackrel{\text{def}}{=} \mathbf{FFT}\{d[n]\}[k] \\ H[k] &\stackrel{\text{def}}{=} \mathbf{FFT}\{h[n]\}[k] \end{aligned}$$

$$\begin{aligned} \frac{\partial \epsilon^2}{\partial h_i} &= \frac{1}{N} \frac{\partial}{\partial h_i} \left\| \log \frac{H[k]\,X[k]}{D[k]} \right\|_2^2 \\ &= \frac{1}{N} \frac{\partial}{\partial h_i} \sum_{k=0}^{N-1} \left(\log \frac{H[k]\,X[k]}{D[k]} \right)^2 \\ &= \frac{1}{N} \sum_{k=0}^{N-1} 2 \log \frac{H[k]\,X[k]}{D[k]} \frac{\partial}{\partial h_i} \left(\log \frac{H[k]\,X[k]}{D[k]} \right) \end{aligned}$$

$$\begin{aligned} E_i[k] &\stackrel{\text{def}}{=} \frac{\partial}{\partial h_i} \log \frac{H[k]\,X[k]}{D[k]} \\ &= \frac{\partial}{\partial h_i} \log \left(\sum_{n=0}^{N-1} h[n]\, W_N^{nk} \frac{X[k]}{D[k]} \right) \\ &= \frac{\partial}{\partial h_i} \left(\log \left(\sum_{n=0}^{N-1} h[n]\, W_N^{nk} \right) + \log \frac{X[k]}{D[k]} \right) \\ &= \frac{\partial}{\partial h_i} \log \left(\sum_{n=0}^{N-1} h[n]\, W_N^{nk} \right) \\ &= \frac{W_N^{ik}}{\sum_{n=0}^{N-1} h[n]\, W_N^{nk}} \\ E_i[k] &= \frac{W_N^{ik}}{\sum_{n=0}^{N-1} h[n]\, W_N^{nk}} \end{aligned}$$

$$\frac{\partial \epsilon^2}{\partial h_i} = \frac{2}{N} \sum_{k=0}^{N-1} \log \frac{H[k] X[k]}{D[k]} \cdot E_i[k]$$

The optimum is achieved by setting

$$\frac{\partial \epsilon^2}{\partial h_i} = 0 \; \forall i = 0 \ldots N-1$$

$$\begin{aligned} \frac{2}{N} \sum_{k=0}^{N-1} \log \left(\frac{H[k] X[k]}{D[k]} \right) \cdot \frac{W_N^{ik}}{\sum_{n=0}^{N-1} h[n] W_N^{nk}} &= 0 \quad \forall i = 0 \ldots N-1 \\ \frac{2}{N} \sum_{k=0}^{N-1} \log \left(\frac{H[k] X[k]}{D[k]} \right) \cdot \frac{W_N^{ik}}{H[k]} &= 0 \quad \forall i = 0 \ldots N-1 \end{aligned}$$

Then, it holds that $\forall i = 0, \ldots, N-1$ the optimal filter satisfies

$$\begin{aligned} \frac{2}{N} \sum_{k=0}^{N-1} \left(\log(H[k]) - \log \left(\frac{D[k]}{X[k]} \right) \right) \cdot \frac{W_N^{ik}}{H[k]} &= 0 \\ \sum_{k=0}^{N-1} \frac{\log(H[k])}{H[k]} \cdot W_N^{ik} &= \sum_{k=0}^{N-1} \frac{\log \left(\frac{D[k]}{X[k]} \right)}{H[k]} W_N^{ik} \end{aligned}$$

We now identify this as an equation in a "double-transform" domain, i.e.

$$\begin{aligned} \mathcal{H}[i] &\stackrel{\text{def}}{=} \mathbf{FFT} \left\{ \frac{\log H[k]}{H[k]} \right\} [i] \\ \mathcal{T}[i] &\stackrel{\text{def}}{=} \mathbf{FFT} \left\{ \frac{\log \left(\frac{D[k]}{X[k]} \right)}{H[k]} \right\} [i] \end{aligned}$$

so that

$$\mathcal{H}[i] = \mathcal{T}[i]$$

From the uniqueness property of the FFT, if $\mathcal{H} \leftrightarrow \frac{\log H}{H}$ and $\mathcal{T} \leftrightarrow \frac{\log \left(\frac{D}{X} \right)}{H}$ and $\mathcal{H} = \mathcal{T}$, then

$$\frac{\log H}{H} = \frac{\log \left(\frac{D}{X} \right)}{H} \tag{4.5}$$

Table 4.1: Cepstrum-error minimizing filter

$H[k]$	$D[k]=0$	$D[k]\neq 0$
$X[k]=0$	1	∞
$X[k]\neq 0$	0	$\frac{D[k]}{X[k]}$

4.7.1 Interpretation of the Result

If we enforce $H[k] \neq 0$, $X[k] \neq 0$ and $D[k] \neq 0$ then the optimum is attained at

$$H[k] = \frac{D[k]}{X[k]} \tag{4.6}$$

Therefore, in this case, the optimal FIR filter for cepstrum-error minimization is the empirical input-to-output transfer function which can be estimated by the methods described so far (especially adaptive filters).

If $X[k] \equiv 0$ for some k, and $D[k] \neq 0$, the filter is unstable, because we would have to set $H[k] = \infty$ for that k, to satisfy equation 4.5. This makes sense, because it means that for that particular frequency bin, the system output has nonzero energy while the input has zero energy. In other words, the system has infinite gain at that frequency, or put differently, a pole on the unit circle, located at that frequency, and therefore, is unstable.

If both $X[k]$ and $D[k]$ are zero for the same k, then we can set $H[k] = 1$. That way, the system has unit gain at that frequency.

Finally, if $D[k] = 0$ but $X[k] \neq 0$, then we will set $H[k] = 0$. The results are summarized in table 4.1.

In summation we can choose a small damping constant $\varepsilon > 0$ so that we set

$$H[k] = \frac{D[k] + \varepsilon}{X[k] + \varepsilon} \tag{4.7}$$

While the result in section 4.7 is very powerful, linking the complicated problem of cepstrum minimization to the well-known problem of transfer function estimation, it relies upon several assumptions. The main assumption is that we estimate the transfer function from the whole data set, i.e. considering the whole length of the signal. While this is almost always impossible to do in practice, it is fortunately unnecessary. In practice, we perform our estimations on a frame by frame basis, updating a given-length impulse response.

Nevertheless, we may attempt to obtain a filter which minimizes the mean square cepstral error, in hopes of gaining some insight as how to extend the result to our ultimate features, MFCCs.

4.7.1.1 Equivalence Between Cepstral- and Frequency-Domain Minimization

As already suggested in 4.7, a filter is optimal in the cepstral domain if and only if it is optimal in the frequency domain. Since calculations are sometimes less cumbersome in one domain or the other, we can resort to the domain where manipulation is easier.

4.7.2 The Least Mean Square Cepstrum (LMSC) Filter

In this section we introduce a new kind of filter which minimizes the cepstral error in the mean sense, on a block by block basis. We start by defining some notation

$$\begin{aligned} N &= \text{block length} \\ L &= \text{block skip} \\ W &= \exp\left(-\frac{2\pi j}{N}\right) \\ x[n] &= \text{input signal of length } N_x \gg N \\ d[n] &= \text{desired signal of length } N_d \gg N \\ h[n] &= h_n \\ h_n &= \text{optimal filter of length } N_h = N \end{aligned}$$

Let

$$\begin{aligned} X_n[p] &\stackrel{\text{def}}{=} \sum_{k=n-N+1}^{n} x[k] W^{kp} \\ D_n[p] &\stackrel{\text{def}}{=} \sum_{k=n-N+1}^{n} d[k] W^{kp} \end{aligned}$$

Our output at time n is given by

$$\begin{aligned} y[n] &= (h * x)[n] \\ &= \sum_{r=-\infty}^{\infty} h[r]\, x[n-r] \\ &= \sum_{r=0}^{N_h} h[r]\, x[n-r] \quad \forall n \geq r \wedge n \leq N_x + r \\ &= \sum_{r=0}^{\min(N_h,n)} h[r]\, x[n-r] \end{aligned}$$

One fundamental observation is that minimizing the norm of the error in the cepstral domain, is the same as minimizing it in the log-frequency domain. This is because of Parseval's theorem and the linearity of the FT and IFT.

$$\begin{aligned} Y_n[p] &\stackrel{\text{def}}{=} \sum_{k=n-N+1}^{n} y[k]\, W^{kp} \\ &= \sum_{k=n-N+1}^{n} \sum_{r=0}^{\min(N_h,k)} h[r]\, x[k-r]\, W^{kp} \end{aligned}$$

$$\begin{aligned} \epsilon_n[p] &= \log D_n[p] - \log Y_n[p] \\ J(n) &= \|\epsilon_n\|_2^2 \\ &= \sum_{p=0}^{N} |\epsilon_n[p]|^2 \end{aligned}$$

As we already saw in section 2.5.3, the LMS filter update equation is given by

$$h_k(n+1) = h_k(n) - \mu \frac{\partial J(n)}{\partial h_k} \tag{4.8}$$

which in this case will be rewritten as

$$\begin{aligned} b &= \left\lfloor \frac{n}{L} \right\rfloor \\ h_k(b) &= h_k(b-1) - \mu \frac{\partial J(n)}{\partial h_k} \end{aligned}$$

where b is the frame number. Now we calculate the value of the gradient as follows:

$$\begin{aligned}\frac{\partial J(n)}{\partial h_k} &= \frac{\partial}{\partial h_k}\sum_{p=0}^{N}\left|\epsilon_n[p]\right|^2 \\ &= \sum_{p=0}^{N}\frac{\partial}{\partial h_k}\left|\epsilon_n[p]\right|^2 \\ &= \sum_{p=0}^{N}\frac{\partial}{\partial h_k}\left|\log\frac{D_n[p]}{Y_n[p]}\right|^2\end{aligned}$$

$$\frac{\partial J(n)}{\partial h_k} = \sum_{p=0}^{N} 2\left|\log\frac{D_n[p]}{Y_n[p]}\right|\frac{\partial}{\partial h_k}\left|\log\frac{D_n[p]}{Y_n[p]}\right|$$

To simplify the notation, define $A_n[p] \stackrel{\text{def}}{=} \frac{D_n[p]}{Y_n[p]}$ and

$$\begin{aligned}D_n[p] &\stackrel{\text{def}}{=} x_1 + jy_1 \\ &\stackrel{\text{def}}{=} z_1 \\ Y_n[p] &\stackrel{\text{def}}{=} x_2 + jy_2 \\ &\stackrel{\text{def}}{=} z_2 \\ J(n) &\stackrel{\text{def}}{=} \sum_{p=0}^{N}\varphi_p(n)\end{aligned}$$

so that

$$\begin{aligned}\frac{\partial\varphi_p(n)}{\partial h_k} &= \frac{\partial}{\partial h_k}\left|\log A_n[p]\right|^2 \\ &= \frac{\partial}{\partial h_k}\left|\log\left|A_n[p]\right|e^{j\phi_A}\right|^2 \\ &= \frac{\partial}{\partial h_k}\left|\log\left|A_n[p]\right| + j\left(\phi_A + 2\pi m\right)\right|^2 \\ &= \frac{\partial}{\partial h_k}\left(\log^2\left|A_n[p]\right| + \left(\phi_A + 2\pi m\right)^2\right) \\ &= 2\log\left|A_n[p]\right|\frac{\partial}{\partial h_k}\left|A_n[p]\right| + 2\left(\phi_A + 2\pi m\right)\frac{\partial}{\partial h_k}\left(\phi_A + 2\pi m\right)\end{aligned}$$

So we now need to calculate the derivatives of the absolute value and phase of $A_n[p]$.

$$\begin{aligned} \phi_A &= \angle A_n[p] \\ \tan\angle\frac{D_n[p]}{Y_n[p]} &= \frac{\Im\frac{D_n[p]}{Y_n[p]}}{\Re\frac{D_n[p]}{Y_n[p]}} \end{aligned}$$

We calculate the real and imaginary parts of the quotient in terms of the real and imaginary parts of $D_n[p]$ and $Y_n[p]$:

$$\begin{aligned} \Re\frac{z_1}{z_2} &= \Re\frac{x_1+jy_1}{x_2+jy_2} \\ &= \Re\frac{(x_1+jy_1)(x_2-jy_2)}{x_2^2+y_2^2} \\ &= \frac{x_1x_2+y_1y_2}{x_2^2+y_2^2} \end{aligned}$$

$$\begin{aligned} \Im\frac{z_1}{z_2} &= \Im\frac{x_1+jy_1}{x_2+jy_2} \\ &= \Im\frac{(x_1+jy_1)(x_2-jy_2)}{x_2^2+y_2^2} \\ &= \frac{x_2y_1-x_1y_2}{x_2^2+y_2^2} \end{aligned}$$

$$\frac{\Im\frac{z_1}{z_2}}{\Re\frac{z_1}{z_2}} = \frac{x_2y_1-x_1y_2}{x_1x_2+y_1y_2}$$

So now we have

$$\begin{aligned} \frac{\varphi_p'(n)}{2} &= \log|A_n[p]|\frac{\partial}{\partial h_k}|A_n[p]| + (\phi_A+2\pi m)\frac{\partial}{\partial h_k}(\phi_A+2\pi m) \\ &= \log|A_n[p]|\frac{\partial}{\partial h_k}|A_n[p]| + (\phi_A+2\pi m)\frac{\partial}{\partial h_k}\left(\arctan\frac{x_2y_1-x_1y_2}{x_1x_2+y_1y_2}+2\pi m\right) \end{aligned}$$

The calculation of the derivative of the absolute value is straightforward and can be written as

$$\begin{aligned}\frac{\partial}{\partial h_k}\left|A_n\left[p\right]\right| &= \frac{\partial}{\partial h_k}\frac{\left|D_n\left[p\right]\right|}{\left|Y_n\left[p\right]\right|} \\ &= -\frac{\left|D_n\left[p\right]\right|}{\left|Y_n\left[p\right]\right|^2}\frac{\partial}{\partial h_k}\left|Y_n\left[p\right]\right| \\ &= -\frac{\left|D_n\left[p\right]\right|}{\left|Y_n\left[p\right]\right|^2}\frac{\partial}{\partial h_k}\sqrt{x_2^2+y_2^2} \\ &= -\frac{\left|D_n\left[p\right]\right|}{\left|Y_n\left[p\right]\right|^2}\frac{1}{2\sqrt{x_2^2+y_2^2}}\frac{\partial}{\partial h_k}\left(x_2^2+y_2^2\right)\end{aligned}$$

so

$$\frac{\partial}{\partial h_k}\left|A_n\left[p\right]\right| = -\frac{\left|z_1\right|}{\left|z_2\right|^3}\left(x_2x_2'+y_2y_2'\right) \tag{4.9}$$

The phase derivative is much more cumbersome to calculate (yet the result is quite simple). We start off by writing

$$\begin{aligned}\frac{\partial}{\partial h_k}\left(\phi_A+2\pi m\right) &= \frac{\partial}{\partial h_k}\left(\arctan\tan\angle\frac{D_n\left[p\right]}{Y_n\left[p\right]}+2\pi m\right) &(4.10)\\ &= \frac{1}{1+\left(\tan\angle\frac{D_n[p]}{Y_n[p]}\right)^2}\underbrace{\frac{\partial}{\partial h_k}\tan\angle\frac{D_n\left[p\right]}{Y_n\left[p\right]}}_{\text{term}}+2\pi\frac{\partial m}{\partial h_k} &(4.11)\end{aligned}$$

the denominator of which becomes

$$\begin{aligned}1+\left(\tan\angle\frac{D_n\left[p\right]}{Y_n\left[p\right]}\right)^2 &= 1+\left(\frac{x_2y_1-x_1y_2}{x_1x_2+y_1y_2}\right)^2 \\ &= 1+\frac{x_2^2y_1^2+x_1^2y_2^2-2x_1x_2y_1y_2}{x_1^2x_2^2+y_1^2y_2^2+2x_1x_2y_1y_2} \\ &= \frac{x_1^2x_2^2+y_1^2y_2^2+2x_1x_2y_1y_2+x_2^2y_1^2+x_1^2y_2^2-2x_1x_2y_1y_2}{x_1^2x_2^2+y_1^2y_2^2+2x_1x_2y_1y_2} \\ &= \frac{\left(x_1^2+y_1^2\right)\left(x_2^2+y_2^2\right)}{x_1^2x_2^2+y_1^2y_2^2+2x_1x_2y_1y_2}\end{aligned}$$

In what follows we will assume that the value of h_k does not affect our decision of the phase turn value m so that $\frac{\partial m}{\partial h_k}$. Now, using that

$$\begin{aligned}\frac{\partial}{\partial h_k}D_n &= 0\\ \frac{\partial}{\partial h_k}x_1 &= 0\\ \frac{\partial}{\partial h_k}y_1 &= 0\end{aligned}$$

we get that the term in braces in equation 4.11 becomes

$$\begin{aligned}\frac{\partial}{\partial h_k}\left(\frac{x_2y_1 - x_1y_2}{x_1x_2 + y_1y_2}\right) &= \frac{(x_1x_2 + y_1y_2)\frac{\partial}{\partial h_k}(x_2y_1 - x_1y_2) - (x_2y_1 - x_1y_2)\frac{\partial}{\partial h_k}(x_1x_2 + y_1y_2)}{(x_1x_2 + y_1y_2)^2}\\ &= \frac{(x_1x_2 + y_1y_2)\left(y_1\frac{\partial}{\partial h_k}x_2 - x_1\frac{\partial}{\partial h_k}y_2\right)}{(x_1x_2 + y_1y_2)^2}\\ &\quad -\frac{(x_2y_1 - x_1y_2)\left(x_1\frac{\partial}{\partial h_k}x_2 + y_1\frac{\partial}{\partial h_k}y_2\right)}{(x_1x_2 + y_1y_2)^2}\\ &= \frac{x_1x_2y_1x_2' - x_1^2x_2y_2' + y_1^2y_2x_2' - x_1y_1y_2y_2'}{(x_1x_2 + y_1y_2)^2}\\ &\quad -\frac{x_1x_2y_1x_2' + x_2y_1^2y_2' - x_1^2y_2x_2' - x_1y_1y_2y_2'}{(x_1x_2 + y_1y_2)^2}\\ &= \frac{-x_1^2x_2y_2' + y_1^2y_2x_2' + x_1^2y_2x_2' - x_2y_1^2y_2'}{(x_1x_2 + y_1y_2)^2}\\ &= \frac{-(x_1^2 + y_1^2)x_2y_2' + (y_1^2 + x_1^2)y_2x_2'}{(x_1x_2 + y_1y_2)^2}\\ &= \frac{|z_1|^2(y_2x_2' - x_2y_2')}{(x_1x_2 + y_1y_2)^2}\end{aligned}$$

so the phase derivative becomes

$$\frac{\partial}{\partial h_k}\phi_A = \frac{x_1^2x_2^2 + y_1^2y_2^2 + 2x_1x_2y_1y_2}{|z_1|^2|z_2|^2}\frac{|z_1|^2(y_2x_2' - x_2y_2')}{(x_1x_2 + y_1y_2)^2}$$

or

$$\frac{\partial}{\partial h_k}\phi_A = \frac{y_2x_2' - x_2y_2'}{|z_2|^2} \tag{4.12}$$

Now we need to calculate the derivatives of $Y_n[p]$ with respect to h_k. This is a crucial step since it is a representation of the variation in the output as a result of varying the filter tap coefficients:

$$\begin{aligned}
\frac{\partial}{\partial h_k} x_2 &= \frac{\partial}{\partial h_k} \Re\left\{Y_n\left[p\right]\right\} \\
&= \Re\left\{\frac{\partial}{\partial h_k} Y_n\left[p\right]\right\} \\
&= \Re\left\{\frac{\partial}{\partial h_k} \sum_{u=n-N+1}^{n} \sum_{r=0}^{\min(N_h,k)} h_r x\left[u-r\right] W^{pu}\right\} \\
&= \Re\left\{\sum_{u=n-N+1}^{n} x\left[u-k\right] W^{pu}\right\} \\
&= \Re\left\{\sum_{u=n-N+1}^{n} x\left[u-k\right] W^{p(u-k)} W^{pk}\right\} \\
&= \Re\left\{W^{pk} \sum_{u=n-N+1}^{n} x\left[u-k\right] W^{p(u-k)}\right\} \\
&= \Re\left\{W^{pk} X_{n-k}\left[p\right]\right\}
\end{aligned}$$

Similarly, we get

$$\frac{\partial}{\partial h_k} y_2 = \Im\left\{W^{pk} X_{n-k}\left[p\right]\right\}$$

The phase derivative is then

$$\frac{\partial \phi_A}{\partial h_k} = \frac{\Im\left\{Y_n\left[p\right]\right\} \Re\left\{W^{pk} X_{n-k}\left[p\right]\right\} - \Re\left\{Y_n\left[p\right]\right\} \Im\left\{W^{pk} X_{n-k}\left[p\right]\right\}}{\left|Y_n\left[p\right]\right|^2} \tag{4.13}$$

and the absolute value derivative is

$$\begin{aligned}
\frac{\partial \left|A_n\right|}{\partial h_k} &= -\frac{\left|D_n\left[p\right]\right|}{\left|Y_n\left[p\right]\right|^3} \left(\Re\left\{Y_n\left[p\right]\right\} \Re\left\{W^{pk} X_{n-k}\left[p\right]\right\}\right. \\
&\qquad \left. + \Im\left\{Y_n\left[p\right]\right\} \Im\left\{W^{pk} X_{n-k}\left[p\right]\right\}\right)
\end{aligned} \tag{4.14}$$

Finally, we write the gradient as

$$\begin{aligned}
\frac{1}{2} \frac{\partial \varphi_p\left(n\right)}{\partial h_k} &= \log\left|A_n\left[p\right]\right| \frac{\partial}{\partial h_k} \left|A_n\left[p\right]\right| + \left(\phi_A + 2\pi m\right) \frac{\partial}{\partial h_k} \left(\phi_A + 2\pi m\right) \\
&= \frac{1}{\left|z_2\right|^2} \left\{-\left(\log \frac{\left|z_1\right|^2}{\left|z_2\right|^2}\right) \frac{\left|z_1\right|}{\left|z_2\right|} \left(x_2 x_2' + y_2 y_2'\right) + \left(\angle \frac{z_1}{z_2} + 2\pi m\right) \left(y_2 x_2' - x_2 y_2'\right)\right\}
\end{aligned}$$

Then

$$\Upsilon_1 = \overbrace{\left(\log\frac{|z_1|^2}{|z_2|^2}\right)}^{\Delta E_{\text{dB}}} \overbrace{\frac{|z_1|}{|z_2|}}^{\exp\left(\frac{\Delta E}{2}\text{dB}\right)} (x_2x_2' + y_2y_2') \tag{4.15}$$

$$\Upsilon_2 = \left(\underbrace{\angle z_1 - \angle z_2}_{\Delta\Phi} + 2\pi m\right)(x_2y_2' - y_2x_2') \tag{4.16}$$

$$\frac{1}{2}\frac{\partial J(x)}{\partial h_k} = -\sum_{p=0}^{N}\frac{1}{|z_2|^2}\{\Upsilon_1 + \Upsilon_2\} \tag{4.17}$$

where dependence upon p and n of x_1, x_2, y_1, y_2, z_1, z_2,m and Υ_1 and Υ_2 has been avoided for notation's sake.

4.7.2.1 Interpretation

An important interpretation of this result is given by the terms with brackets in the core terms Υ_1 and Υ_2, ΔE_{dB} and $\Delta\Phi$ respectively. The first term is directly the difference in energy (measured in dB) between the desired output and the output produced by using $h(n)$ and it itself weighted by $\exp\left(\frac{\Delta E}{2}\text{dB}\right)$ (which is nothing but the ratio of energies ϱ_E, measured as a linear gain). The second is directly the phase difference between the desired and produced outputs.

4.7.2.2 Summary

We have arrived at an equation which provides a block-by-block adaptive algorithm that minimizes the cepstral error. This is yet another **novel** development to the best of the author's knowledge. The update equation is

$$h_k(n) = h_k(n-N) + \Delta h_k(n)$$

$$\Delta h_k(n) = \mu\sum_{p=0}^{N}\frac{1}{|z_2|^2}\left\{\Delta\Phi(x_2y_2' - y_2x_2') + \varrho_E\cdot\Delta E_{\text{dB}}(x_2x_2' + y_2y_2')\right\} \tag{4.18}$$

This equation may be more compactly written in matrix notation as

$$\begin{aligned} h_k(n) &= h_k(n-N) + \mu \sum_{p=0}^{N} \frac{1}{|z_2|^2} \left\{ x_2 \left(\Delta\Phi y_2' + \varrho_E \Delta E_{\text{dB}} x_2' \right) \right. \\ &\qquad \left. + y_2 \left(-\Delta\Phi x_2' + \varrho_E \Delta E_{\text{dB}} y_2' \right) \right\} \\ &= h_k(n-N) + \mu \sum_{p=0}^{N} \frac{1}{|z_2|^2} \begin{pmatrix} x_2 & y_2 \end{pmatrix} \begin{pmatrix} \varrho_E \Delta E_{\text{dB}} & \Delta\Phi \\ -\Delta\Phi & \varrho_E \Delta E_{\text{dB}} \end{pmatrix} \begin{pmatrix} x_2' \\ y_2' \end{pmatrix} \\ &= h_k(n-N) + \mu \sum_{p=0}^{N} \frac{1}{|z_2|^2} \begin{pmatrix} x_2 & y_2 \end{pmatrix} Z \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_2' \\ y_2' \end{pmatrix} \end{aligned}$$

$$h_k(n) = h_k(n-N) + \mu \sum_{p=0}^{N} \begin{pmatrix} x_2 & y_2 \end{pmatrix} \frac{Z}{|z_2|^2} \cdot \mathcal{R}_\theta \begin{pmatrix} x_2' \\ y_2' \end{pmatrix} \tag{4.19}$$

where $Z = \sqrt{\left(\varrho_E \Delta E_{\text{dB}}\right)^2 + \left(\Delta\Phi\right)^2}$ and $\theta = \arctan \frac{\Delta\Phi}{\varrho_E \Delta E_{\text{dB}}}$ and $\mathcal{R}_\theta$ is a rotation matrix with angle θ. The interpretation of this last expression is that the matrix $\frac{Z}{|z_2|^2}\mathcal{R}_\theta$ represents a rotohomotethy, which is applied to the variation of Y_n with respect to the filter tap coefficients, and that rotated vector is multiplied using the $\mathbb{R}^2$ internal product (figure 4.1).

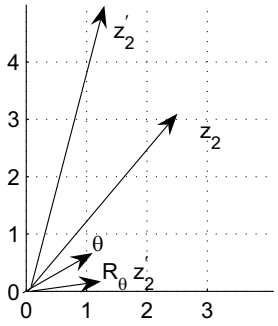


Figure 4.1: Geometrical interpretation of equation 4.19

So, the procedure for each frame of the input and desired output is as follows:

1. Calculate the Fourier Transform of the desired output ending at time n, $D_n[p] \quad \forall p = 0, \ldots, N-1$

2. Calculate the output using the previous filter coefficients ($h_k(n-N)$), $y[k] \ \forall k = n-N+1, \ldots, N$ and its corresponding Fourier Transform $Y_n[p]$. $y(k) = \sum_m x[m] h_{k-m}$

3. For each $k = 0, \ldots, N_h$, calculate the N-point FFT of x up to sample $n-k$, $X_{n-k}[p]$
4. Calculate the change in the filter coefficients according to 4.18.

To initialize the algorithm, we can use the following definitions, which correspond to the equivalence between cepstrum minimization and frequency minimization for a single realization:

1. $H_k(0) = \frac{D_N[k]+\delta}{X_N[k]+\delta}$ where $\delta \ll 1$ is a very small positive number.
2. Calculate $h_p(0) = \mathbf{IFFT}\{H_0[k]\}[p]$

The overall procedure is summarized in algorithm 6. The main advantage of this algorithm as compared to plain RLS, is that it is explicitly minimizing the cepstral error, instead of the energy. Note that the energy is closely related to one of the cepstral coefficients (the 0^{th}coefficient). It is similar in that the results should be the same, but we are explicitly using a kind of explicitness on the choice of cepstrum as a feature. This is depicted by the presence of the $\log Y_n[p]$ divided by the actual output $Y_n[p]$.

While this approach will work, it suffers from the same problems as LMS, namely slow convergence and dependency upon the input signal gain (this is an unnormalized algorithm). Moreover, the decision of an appropriate value for m in the calculation of Υ_2 is a priori arbitrary[2]; one criterion might be to minimize phase jumps over time, another might be to minimize them within a frame but between frequency bins. All in all, this is an extremely complicated algorithm to arrive at, and very computational intensive, while only considering cepstrum error minimization. The upside to this algorithm is that it is a block-by-block algorithm as opposed to having to know the whole signal beforehand. Nevertheless, by this thorough algebraic manipulations we arrived at an **elegant** and **sensible** update equation which takes into consideration the energy differences between the desired signal and the produced signal at each iteration, as well as their phase differences; so we have a direct link between the magnitude

An alternative approach which will work for **any** kind of feature choice is to use search algorithms. These are algorithms borrowed from the area of Operation Research and Optimization. We will now briefly expose a method called the cyclic coordinate search, or spider-web search.

[2]and for that matter, the assumption that $\frac{\partial m}{\partial h_k}$ is arbitrary too.

Algorithm 6 Least-mean cepstrum search

Input:

- N algorithm block size
- δ a small, positive regularization parameter
- $x\left[n\right]$ input signal of length $N_x \gg N$
- $d\left[n\right]$ desired output of length $N_d \gg N$

Output:

- $h_k,\ k = 1, \ldots, N$ filter tap coefficients

Definitions

- $\mathbf{STFT}_N \left\{g\left(m\right)\right\}\left[p\right] = \sum_{n=m-N+1}^{m} g\left(n\right) W^{np}$
- $D_k\left(m\right) = \mathbf{STFT}_N \left\{d\left(m\right)\right\}\left[k\right]$
- $X_k\left(m\right) = \mathbf{STFT}_N \left\{x\left(m\right)\right\}\left[k\right]$
- $Y_k\left(m\right) = \mathbf{STFT}_N \left\{\left(x * h\left(n\right)\right)\left(m\right)\right\}\left[k\right]$
- $x_1 = \Re D_k\left(m\right)$, $y_1 = \Im D_k\left(m\right)$
- $x_2 = \Re Y_k\left(m\right)$, $y_1 = \Im Y_k\left(m\right)$
- $z_1 = x_1 + jy_1$, $z_2 = x_2 + jy_2$
- $x_2' = \Re\left\{W^{pk} X_{n-k}\left[p\right]\right\}$, $y_2' = \Im\left\{W^{pk} X_{n-k}\left[p\right]\right\}$

1. $\Upsilon_1 = \left(\log \frac{|z_1|^2}{|z_2|^2}\right) \frac{|z_1|}{|z_2|} \left(x_2 x_2' + y_2 y_2'\right)$
2. $\Upsilon_2 = \left(\angle z_1 - \angle z_2 + 2\pi m\right)\left(x_2 y_2' - y_2 x_2'\right)$

Initialization:

1. $H_k\left(0\right) = \frac{D_k(N)+\delta}{X_k(N)+\delta}$
2. $h_k\left(0\right) = \mathbf{IFFT}\left\{H_k\left(0\right)\right\}$

Recursion

$$h_k\left(n\right) \quad = \quad h_k\left(n-N\right) + \mu \sum_{p=0}^{N} \frac{1}{|z_2|^2} \left\{\Upsilon_1 + \Upsilon_2\right\}$$

4.8 General Feature Error Minimization

In this section we attempt to provide a review of a very general optimization method called cyclic coordinate search, applied to the problem of finding a filter's coefficients such that the feature error energy is minimized. The only constraint is that the objective function (and therefore the features) are continuous $\mathcal{C}^0$ functions of their input (this will almost always be the case). Contrast this with LMS, or steepest descent methods in which we use the gradient of the objective function, therefore requiring the features to be at least $\mathcal{C}^1$.

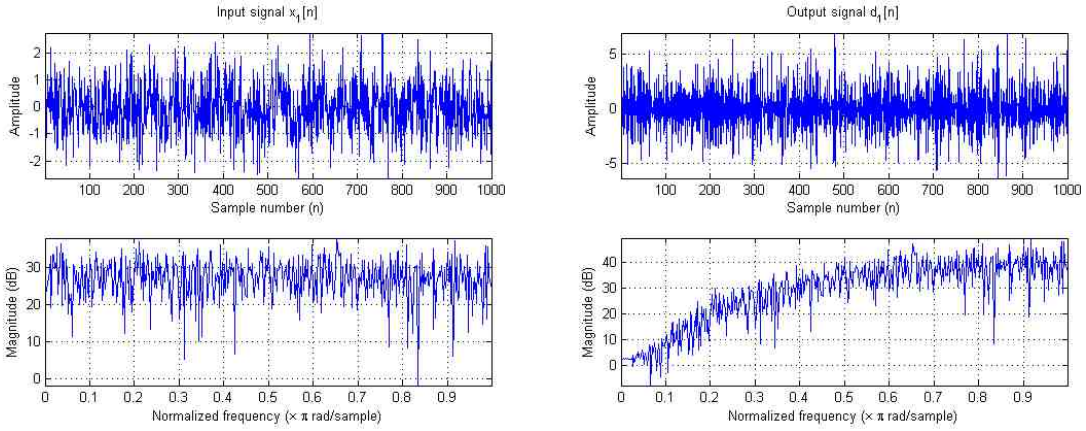
4.8.1 Exhaustive Search

To illustrate our reasoning, let us consider an input signal x_1 drawn from a Gaussian white noise source with zero mean and unit variance (figure 4.2(a)). Suppose this signal goes through a linear known filter H whose z-transform is $H(z) = 1 - 2z^{-1} + z^{-2}$, to render an output d_1 (figure 4.2(b)). Finally, let us deal with MFCCs in his example, so that our objective is find an h such that

$$h = \arg\min \|\mathcal{M}\{d_1\} - \mathcal{M}\{h * x_1\}\|$$

where $\mathcal{M}$ is an operator rendering an MFCC vector for each frame.

As already stated, we readily know that the optimum will be attained at $h = H$; in this case, the minimum is trivial since the value of the norm is zero by definition. But let us concern ourselves to evaluating what happens when we change each value of the impulse response by a small number δ in each direction (i.e. we allow positive and negative variations).



(a) Input signal

(b) Output signal

Figure 4.2: Input and output signal

A first approach to estimate an impulse response is to choose a grid discretization step δ, and then exhaustively traverse this grid in tap coefficient space. Table 4.2 summarizes the MFCC error value around the optimum $H(z)$, by changing the three tap coefficients by $-\delta$, 0 or $+\delta$. Obviously, the optimal is achieved by setting $\delta_1 = \delta_2 = \delta_3 = 0$, but the variations in errors are very wide. For example, $\delta_1 = +\delta$, $\delta_2 = -\delta$ and $\delta_3 = 0$ gives an error of 1.2, while the same configuration with $\delta_3 = -\delta$ gives an error of 11.4, an almost tenfold increase. This illustrates how creased and wrinkled the error performance surface is, full of saddle points, and therefore how non-uniform the gradient tensor is. Since conventional derivative-based search methods try to fundamentally invert the gradient tensor (as in the Newton-Raphson method), we will not be able to confidently use those in our search.

Table 4.2: MFCC error variation with $\delta = 0.05$

(a) $\delta_1 = -\delta$

$\delta_2 \backslash \delta_3$	$-\delta$	0	$+\delta$
$-\delta$	19.9	16.5	11.4
0	16.4	11.2	**2.4**
$+\delta$	11.1	**1.2**	10.1

(b) $\delta_1 = 0$

$\delta_2 \backslash \delta_3$	$-\delta$	0	$+\delta$
$-\delta$	16.5	11.3	**1.3**
0	11.2	**0**	10.5
$+\delta$	**1.1**	10.8	16.1

(c) $\delta_1 = +\delta$

$\delta_2 \backslash \delta_3$	$-\delta$	0	$+\delta$
$-\delta$	11.4	**1.2**	10.4
0	**2.2**	10.1	16.4
$+\delta$	9.8	16.3	20.5

Note that for a filter of order M, if we want to do an exhaustive search we need to test for $\delta_i = -\delta, 0, +\delta$ for each $i = 1 \ldots M$, so that this needs M^3 evaluations of the MFCC matrix for **each** iteration step. For a T-second long audio file, we have that

$$\begin{aligned} N &= T\text{s} \times 8000 \frac{\text{sample}}{s} \\ &= 8T \times 10^3 \text{sample} \end{aligned}$$

Calculating the MFCC matrix entails calculating the FFT for each block of data. If we use a standard block size of 200 samples (25 ms) with 50% overlap (12.5 ms), we have $\frac{8T \times 10^3}{100} = 80T$ data blocks. For each data block we compute a 512 point FFT (256 points in positive frequencies), which uses $256 \log_2 512 = 2304$ floating point operations. After that, we also need to apply the mel-scale warping; this is done by multiplying with a constant matrix, which requires $40 \times 256 \times 80T = 819200T$[3]. After that, we calculate the DCT of the log for each matrix column, which takes approximately the same number of operations as the DFT.

[3]Multiplying an $M \times N$ by a $N \times P$ matrix requires $M \times N \times P$ products and additions. The Hz2Mel matrix consists of 40 mel frequency bins and 256 columns.

All in all, one evaluation of the MFCC matrix needs $80T \times 2304 + 2 \times 819200T = 1822720 \times T$ Flops $= 1.82T$ MFlops. An Intel® Pentium® M 725 (Centrino®) processor running at 1.6 GHz, can perform at about 2.5 MFlops, so each one-second evaluation would take around $0.7s$. This is the main cause why exhaustive algorithms cannot be used: we have a function to minimize which is prohibitively expensive to calculate. At the same time, there are no easy ways to calculate gradients to use gradient descents. Additionally, we want to use small grid values δ to be able to have a good discretization and therefore an accurate estimate of our impulse response, but this will increase the number of evaluations the algorithm has to perform since at each step it can advance only δ units[4]. Hence, we find that using an exhaustive search is only useful for proof-of-concept and not feasible with the current state of the art of home computing. Nevertheless, when personal computers will start achieving even greater computational power, or conversely if and when a compact, tractable gradient for the MFCC calculation is derived, the algorithm will leverage that power and render the desired solution in an acceptable time frame.

4.8.2 Line Search

Searching for optima in a given direction is the backbone of many search algorithms. The problem may be stated as follows: given a point x_k and a direction vector d_k find λ_k such that $f(x_k + \lambda d_k)$ is minimized (or at least decreases with respect to $f(x_k)$. This is a one-dimensional search problem in the variable λ (note that x_k and d_k may be vectors of any dimension). The minimization can furthermore be constrained to real λ, positive λ or some other restriction.

In general when we want to minimize a function $\theta(\lambda)$, we proceed to calculate its derivative and solve for the derivative to be zero. There are a number of problems with this approach. First, the function θ may be defined implicitly in terms of another function of several variables: $\theta(\lambda) = f(x + \lambda d)$. Secondly, more often than not θ fails to be differentiable (this is even more complicated in the case θ needs to be a complex function of a complex variable, since differentiability is a much stronger concept[5] than in the real line). If f is differentiable, then we need to solve $\theta'(\lambda) = d^T \nabla f(x + \lambda d) = 0$, which is usually nonlinear in λ. To complicate matters even further, even if we are able to solve for zero derivative, we might end up with points that are not minima, such as maxima and saddle points. To get rid of those we need to calculate the value of the second derivative of $\theta(\lambda)$, which entails calculating the Hessian matrix of f. Therefore, minimization in general by setting the derivative to zero is not a good (or at least tractable) method. Instead, we

[4] This can be somewhat alleviated by using a large δ at first and then subsequently switching to smaller δ.

[5] the concept we refer to is that every differentiable complex function is holomorphic.

prefer using numerical methods whenever possible (such as when a closed form solution is no better than a numerical one).

4.8.2.1 Convexity

A function f that satisfies $f\left(\lambda x_1 + (1-\lambda)\, x_2\right) \leq \max\left\{f\left(x_1\right); f\left(x_2\right)\right\} \quad \forall 0 \leq \lambda \leq 1$ is called quasiconvex; similarly, the function is strictly quasiconvex if the inequality is strict. A set is called convex if given two points x_1 and x_2, the line segment described by $l = \lambda x_1 + (1-\lambda)\, x_2$ is entirely contained in the set.

Strict quasiconvex functions play an important role in optimization because when we find a local minimum or maximum on a convex set, we are guaranteed to have found a **global** optimum. In general it is fairly difficult to prove that a function is strictly quasiconvex, so an approach would be to divide the function domain in small cells in which the function behaves quasiconvexly, find the minima in each cell and then find the global minima among all cells. Moreover if one uses methods that assume quasiconvexity and the function is not quasiconvex, then the methods will converge to local minima.

4.8.2.2 Fibonacci Method

Suppose we try to minimize the function $\theta\left(\lambda\right)$ for $\lambda \in [a_1, b_1]$. Given these two points we generate two new points λ_k and μ_k defined as

$$\begin{aligned} \lambda_k &= a_k + \frac{F_{n-k-1}}{F_{n-k+1}}\left(b_k - a_k\right) \\ \mu_k &= a_k + \frac{F_{n-k}}{F_{n-k+1}}\left(b_k - a_k\right) \end{aligned}$$

where F_t is the t^{th} Fibonacci number, n is the number of expected function evaluations and $k = 1 \ldots n-1$. Then it can be proven that at each iteration the new interval of uncertainty (i.e. the interval within which the optimum is achieved) is reduced from $[a_k, b_k]$ to $[\lambda_k, b_k]$ if $\theta\left(\lambda_k\right) > \theta\left(\mu_k\right)$ and to $[a_k, \mu_k]$ if $\theta\left(\lambda_k\right) \leq \theta\left(\mu_k\right)$. In any case, we get

$$b_{k+1} - a_{k+1} = \frac{F_{n-k}}{F_{n-k+1}}\left(b_k - a_k\right)$$

The trick with this method is that it requires two evaluations to start with and then just one evaluation per iteration, since it can be proven that if $\theta\left(\lambda_k\right) > \theta\left(\mu_k\right)$ then $\lambda_{k+1} = \mu_k$ and similarly if $\theta\left(\lambda_k\right) \leq \theta\left(\mu_k\right)$ then $\mu_{k+1} = \lambda_k$. The number of iterations n must be chosen so that $\frac{b_1 - a_1}{F_n}$ reflects the required accuracy.

Algorithm 7 Cyclic coordinate search

Initialization Choose $\varepsilon > 0$, a tolerance constant used to terminate the algorithm
Let x_1 be an initial point and $y_1 = x_1$
Recursion

1. Let λ_j be an optimal solution to $\min(y_j + \lambda d_j)$ and let $y_{j+1} = y_j + \lambda d_j$
2. If $j < n$ let $j = j + 1$ and go to step 1. Otherwise, $j = n$ and go to step 3
3. Let $x_{n+1} = y_{n+1}$.
4. If $\|x_{n+1} - x_n\| < \varepsilon$ then stop. Otherwise let $y_1 = x_{k+1}$ and $j = 1$ and go to step 1.

4.8.3 Cyclic Coordinate Search

The cyclic coordinate search uses the coordinate axis as search directions, i.e. searches along the directions $d_1 \ldots d_N$ where d_j is a vector of zeros with a one in the j^{th} coordinate, so the search is conducted by going in the direction of d_1, finding the minimum, then going in the direction of d_2, and so on. In a variant known as the Aitken Double-sweep method, after we search along d_N, we go back to d_{N-1}, d_{N-2}, up until d_1. The algorithm is exemplified in figure 4.3 and stated in algorithm 7.

If the objective function is non-differentiable, the method may stall at a non-optimum point, as depicted in figure 4.4, because of the sharp-edged valley in the search space. The difficulty can be overcome by searching along the direction $x_2 - x_1$. A search along the direction $x_{k+1} - x_k$ is usually introduced every p iterations even if the function is differentiable, since this modification speeds up convergence. This modification is called an acceleration step.

It can be proven that a cycle through all the coordinates is equivalent to one iteration of steepest descent, which goes to show how inefficient searching a non-differentiable function can be. In-depth information on searching and nonlinear programming as well as a number of more complicated search methods can be found in [122].

4.9 Least Feature-Error Norm Filter

In this section we describe an estimation procedure which we will call the Least Feature-error Norm filter. The idea of this filter is to minimize the error energy in the given feature space. For that purpose, we use a cyclic coordinate search with acceleration steps inter-

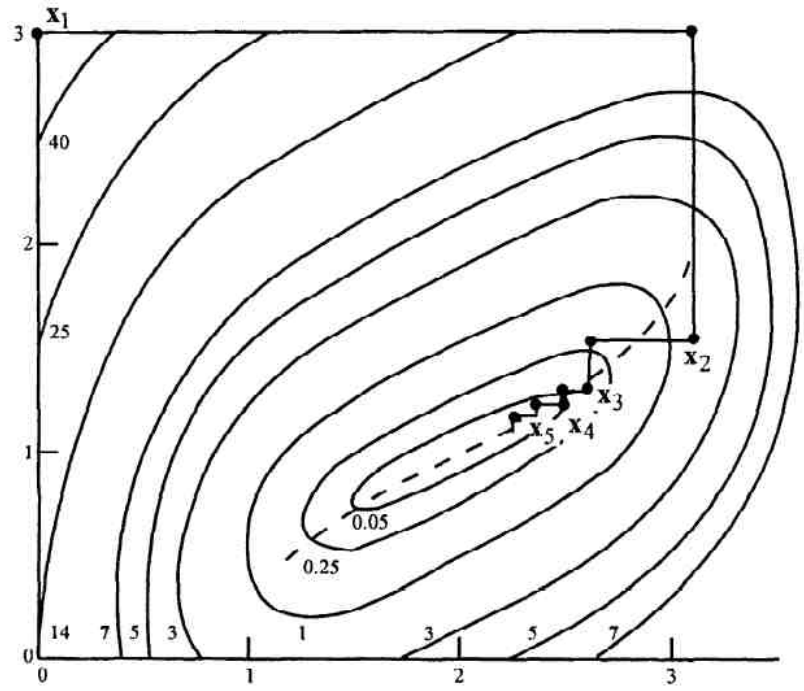
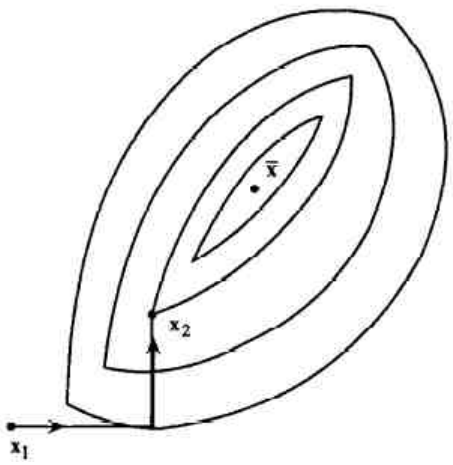
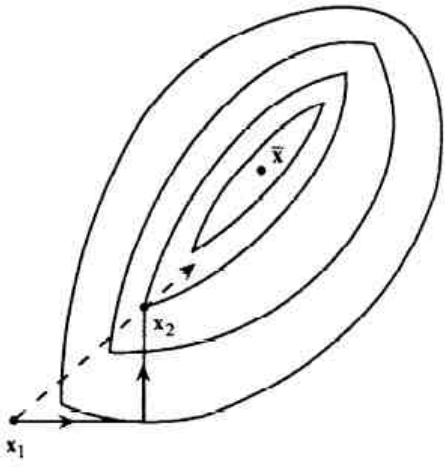


Figure 4.3: Cyclic Coordinate search example (taken from [122])

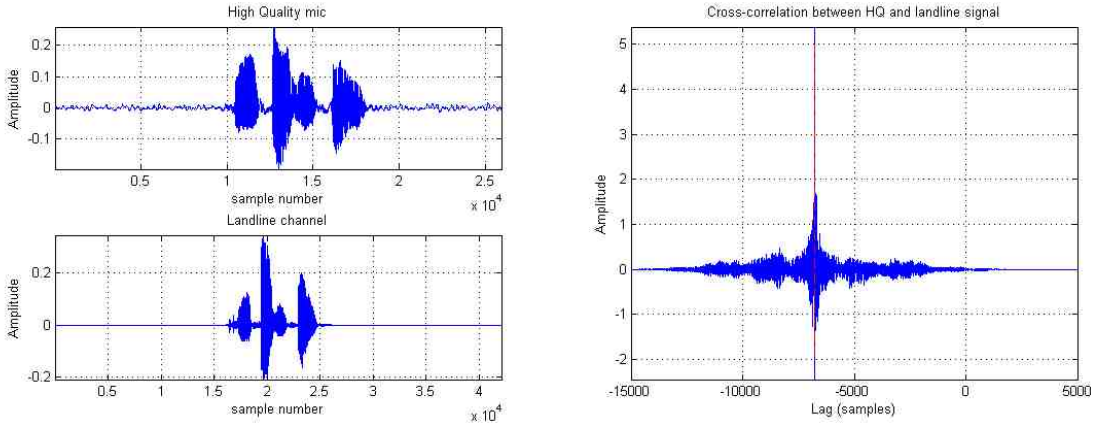


(a) x_2 is a stall point. Despite going along any of the coordinate axes, we will not find a better point



(b) Acceleration step speeds up convergence

Figure 4.4: Stalling and acceleration step in Cyclic Coordinate Search



(a) Waveforms (b) Cross-correlation

Figure 4.5: HQ and landline channel signals and their cross-correlation

leaved whenever a whole cycle does not produce new points. We ran a standard RLS filter to estimate the input-output transfer function in parallel to compare results.

Figure 4.5 shows the HQ mic and landline signals corresponding to a single test utterance, as well as the cross-correlation and the point at which the maximum absolute value of the cross-correlation occurs.

The comparative magnitude responses of the 13 tap RLS and the 13 tap LFEN estimates are shown in figure 4.6. Both filters were design to estimate the landline signal delayed by 6 samples from the high quality signal[6]. We see that while the DC and high frequency gains are relatively equal, the middle section of the frequencies present a difference of up to 25.1 dB at $\omega = 0.129\pi$.

On the other hand, by looking at the pole-zero map in figure 4.7,we see that the zeros[7] (at $\omega = 0$ and $\omega = \pi$) in both estimates are extremely close, while the mid-frequencies zeros are not too distant from one another but rather they exhibit a certain "shifting" pattern, i.e. the complex conjugate zeroes pairs pattern of the LFEN can be obtained by a homotethy. The reason behind the differences in middle frequencies is of course, the mel scaling that occurs when calculating MFCCs, as well as the different weighting of the frequency components (MFCCs have a logarithmic weighting to them).

For the LFEN, the MFCC error norm[8] per unit sample was $\vartheta^{\text{LFEN}}_{\text{MFCC}} = 0.0719$, while for the RLS filter was $\vartheta^{\text{RLS}}_{\text{MFCC}} = 0.0771$. In that respect we see that while the RLS filter

[6]The delay is necessary as already discussed, to consider anticausal components.

[7]The RLS zeros have been reflected back into the unit circle (to make the response minimum phase). This allows to compare the behavior of magnitude responses without concerning ourselves with how the phase is affected.

[8]Frobenius norm (2-norm)

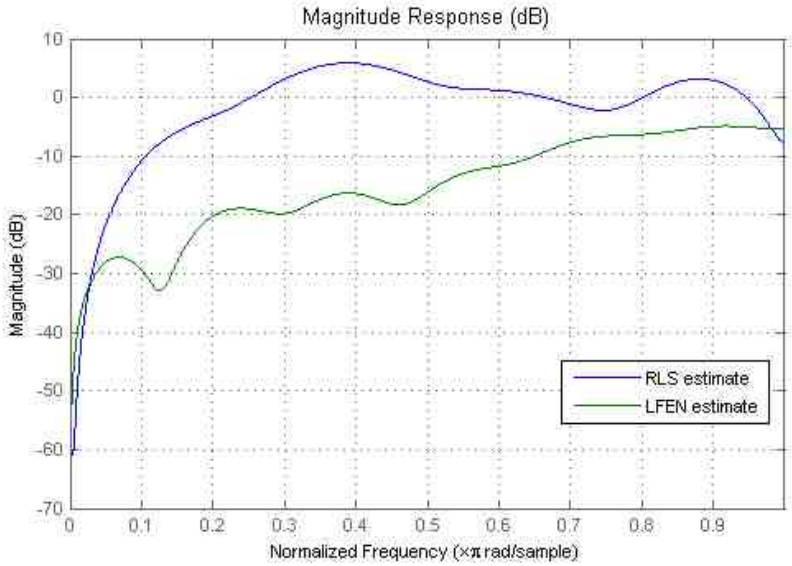
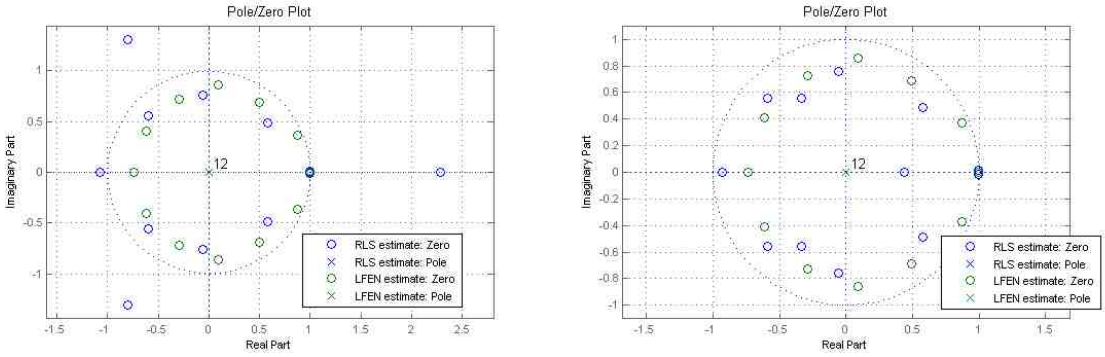


Figure 4.6: Magnitude responses of RLS and LFEN estimates



(a) Original zero-pole map (b) Minimum-phase zero-pole map

Figure 4.7: Zero pole maps of RLS and LFEN estimates

minimizes the error norm in the time domain, it does not do that in the MFCC space. On the other hand, if we consider time domain error energy, we get that the error norms per unit sample were $\vartheta_t^{\text{LFEN}} = 9.9 \times 10^{-5}$ and $\vartheta_t^{\text{RLS}} = 8.7 \times 10^{-5}$, so the RLS filter outperforms the LFEN filter as to time domain energy minimization, as expected.

In summation, the LFEN outperforms the RLS in the feature space, while the converse is true in the time domain. As for computational purposes, even with the stringent RLS inverse correlation matrix computation, is much more efficient than the LFEN search. To begin with, the RLS filter converges very quickly, while LFEN has to go through many iterations to converge to values which is ultimately comparable to that of the RLS. All in all, while a LFEN filter is better for our purposes, it may not be a feasible option to use when the audio files are long, are sampled at higher frequencies or when many estimates have to be averaged (as is our case). Therefore, since estimating the LFEN filters will take a very long time and we arrive at errors in the same order of magnitude as the RLS filters, we can proceed by using the latter.

Chapter 5

Development

5.1 Relation to the IBM Project

This thesis was conducted within the framework of the IBM-UM ASR Joint Project. This Project seeks to eliminate the need for channel-specific data collection to be used in ASR engine retraining, by obtaining a compact and complete representation of the processes a signal undergoes along the channel. As already stated, the motivation for this is threefold:

1. The availability of collected clean speech
2. The high costs and necessary times resulting from additional data collection processes.
3. The need to collect data for each specific channel

Therefore, if we would have to retrain the system with new data, we would incur in excessive costs of both monetary and temporal nature. To remedy this, some of the approaches presented in the past chapters were implemented to try to simulate channel data from clean data. This simulation has to have certain characteristics

On one hand, the simulation has to preserve the format of the input data. This means, that just as the system is already set up to receive its input data from the channel in PCM format (i.e. time domain), then our simulation must output time-domain data. Thus we are limited to use the class of analysis techniques which are invertible.

On the other hand, our simulation must be able to simulate the channel characteristics; in that respect, the simulation must be optimal among all other possible models with the same *ansatz* or form. This optimality condition is given by minimizing a measure of the

"distance" between the actual channel output and the simulated data. We can put these two statements together in a mathematical form and so that the optimal simulation $\mathcal{S}$ satisfies the optimization problem

$$\mathcal{S} = \arg\min_{S} \left\| \mathcal{M}\left\{S\left\{x\left[n\right]\right\}\right\} - \mathcal{M}\left\{d\left[n-\Delta\right]\right\} \right\|_{\Psi} \tag{5.1}$$

where

- $x\left[n\right]$ clean speech
- $d\left[n\right]$ channel output
- $y\left[n\right] = S\left\{x\left[n\right]\right\}$ simulated data
- Δ a constant delay
- $\mathcal{M}$ a transformation which maps time domain into another domain, to be defined
- $\left\|\cdot\right\|_{\Psi}$ a type of norm. The degree of freedom introduced by μ in principle is not fundamental but we will see its usefulness

The simulation function S is defined as acting on a real-valued time-domain signal, and outputting another real-valued time-domain signal. Therefore, we may write

$$S \ : \ \mathbb{R}^{\mathbb{Z}} \rightarrow \mathbb{R}^{\mathbb{Z}}$$

or assuming causality

$$S \ : \ \mathbb{R}^{\mathbb{N}} \rightarrow \mathbb{R}^{\mathbb{N}}$$

The domain-mapping functional $\mathcal{M}$ takes a time-domain signal and outputs a dual domain signal. The definition of the dual domain is a fundamental part of the optimization process. We want to choose the dual domain so that, as far as the recognizer knows, the channel output and the optimal simulated data are very close. In other words, we must choose the functional $\mathcal{M}$ to match the features that the ASR front-end extracts.

The norm $\left\|\cdot\right\|_{\mu}$ introduces a degree of freedom which allows to induce a metric over the dual domain. Basically, if we identify the dual domain with the space $\mathbb{C}^{N^{\mathbb{N}}}$, i.e. sequences of N-dimensional complex-valued vectors, then the metric can weigh each component of the N-dimensional vector differently. Although this weighting could be taken care of by a

more accurate choice of the domain-mapping functional $\mathcal{M}$, we prefer separating the two since that way we maintain a more direct relation to the recognition front-end.

In particular, we will be interested in the class of two-norms arising from the following definition

$$\|v\|_{\Psi} = \sum_{k \in \mathbb{N}} v\,[k]^{H}\, \Psi v\,[k]$$

where the matrix Ψ is positive semi-definite. We follow with an intuitive explanation of our statement. The dual domain will usually consist of a time domain sequence, where each sequence element is a vector of features. In general we might say that the dual domain representation will be a feature evolution along frames. In the limiting case, we might have that the frames skipping length is of one sample, which would yield a feature signal with the same time resolution as the originating time-domain signal. Since usually this would be overkill, the lengths and therefore the time resolution of the feature signal and the original time-domain signal will be different.

Note that another degree of freedom might be included by letting the norm be time-varying, that is, assign a different weight to the k^{th} feature at different instants in time. To formalize this, we might write either

$$\|v\|_{\Psi} = \sum_{k \in \mathbb{N}} v\,[k]^{H}\, \Psi\,[k]\, v\,[k]$$

or also

$$\|v\|_{\Psi} = \sum_{k \in \mathbb{N}} \rho\,[k]\, v\,[k]^{H}\, \Psi\,[k]\, v\,[k]$$

where

- $\Psi\,[k]$ is the time-varying weighting matrix;
- $\rho\,[k]$ is a time-varying factor; note that this second approach uses a constant feature-weighting matrix, so the time-weighting is uniform over all features for a given time.

Note that the first approach is more general than the second, since one might factor out a proportionality factor $\rho\,[k]$ from the matrix $\Psi\,[k]$ or equivalently define $\Psi\,[k] = \rho\,[k] \cdot \Psi_0$.

Despite the generality of these time-varying norms, because of the kinds of features we will be using and the stationarity of the channels in question, it doesn't make much sense to assign different weights at different instants in time. However, it is important to note that we might want to consider them when trying to simulate non-stationary channels.

5.2 Collection of the Speech Database

The first step in designing a successful simulation system is obtaining a large set of utterances processed by the channel to be modeled, and at the same time, clean versions of those utterances. That way, we process the obtained data starting from a set of initial model coefficients, and arriving at another set of model coefficients that are optimal for the just seen data set. This process is known as the training or adaptation phase, because we iteratively[1] change our model so that the adapted model is more likely to have generated the channel data.

5.3 Setup

In this section we describe the connection setup and architecture of the data collection process.

5.3.1 Description of the process

The collection process involves two classes of users:

1. Operators or proctors
2. Speakers

For this project, more than 60 speakers were recorded by 6 operators. Operators were also recorded, so as to get as much data as possible. Each speaker uttered 200 phrases from a set of 50 scripts. The scripts contained phrases from various typical IVR use-cases: yes/no responses, access to e-mail over the phone, access to accounts, etc.

Figure 5.1 depicts the connection setup for the speech database collection task. The recording use case is described in algorithm 8.

5.3.2 Differences Between the Local and Remote Audio File Repositories

Files in the local laptop (HQ mic and VoIP headset) are saved as raw PCM files at 44.1 kHz 16 bits. On the other hand, files in the server side are saved as NeXT/Sun audio files in

[1] This iteration is done on two levels: first, inside a given utterance, on a frame-by-frame basis or a sample-by-sample basis. Then, the iteration goes across utterances.

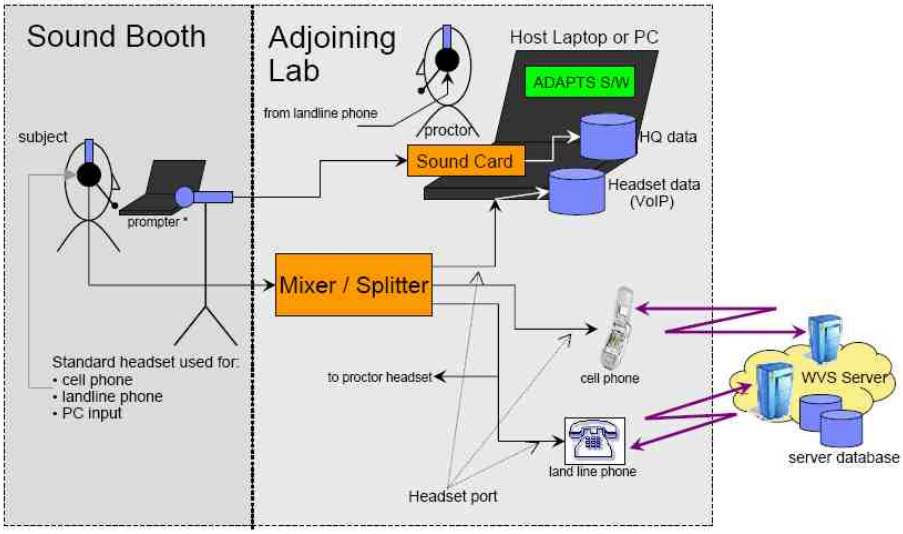


Figure 5.1: Collection Connection Setup

the AU format at 8 kHz 16 bits. While on the local laptop each file represents an utterance (phrase), each file on the server side represents a session. A session is usually composed of about 25 utterances. Using more than one phrase per session allows a speedier recording (since going on to the next session involves some waiting for the server to save the file), while allowing the operator to disregard mistakes made by the speaker in between correct phrases. On the other hand this introduces the need to segment the server files into phrases. Thus, while the beginnings and endings of the files on the laptop were triggered by the press of a key, on the server side there is no *ad hoc* synchronization to pinpoint those events.

5.3.3 Testing Procedure

Since we have abundant data, we can set aside a big number of utterances to be used to test the validity of our model. This implies that the test data will not be used to estimate the model parameters.

5.4 Difficulties in the Collection Process and Expected Ramifications

Because of the complicated nature of the recording setup (multiple microphones, remote telephony server, data network connection), we had to face several difficulties in the recording process.

Algorithm 8 Recording use case

1. Operator dials into the WebSphere Voice Server (WVS) using the landline phone
2. Initialize the system for data collection session by entering a speaker ID, and a session ID.
3. Operator dials into the WVS Server using the cellular phone
   (a) Initialize the system for data collection by entering the same speaker ID and session ID
4. WVS system prompts the caller to speak the first phrase
   (a) The proctor hears the prompt and initiates the appropriate utterance prompting using the ADAPTS control program
      i. The ADAPTS control program displays the appropriate utterance text on the prompter
      ii. The subject speaks the phrase
      iii. The ADAPTS control program records/stores the audio from the high quality input and headset input on the local laptop
   (b) Server monitors the audio from the landline and the cellular channels
      i. Upon detection of end of speech for both channels, the server saves the audio in the appropriate session directory/files
5. The server prompts for the next utterance in the series
6. When the operator hangs up, the WVS copies the recorded audio files to a local UM server through an HTTP connection.

5.4.1 Server Crashes

The first difficulty we had to face had to do with server reliability. Often the server would not be able to bounce its recorded audio files to our server, either because the files were too long, the connection would time out, or some transient network error. This would cause the entire session to be lost . This was especially true at the beginning of the collection process, where some configuration parameters weren't correctly tuned for optimal performance (especially HTTP timeouts regarding file transfers).

5.4.2 Recording Conditions

Another difficulty arises in the presence of what appears to be a low amplitude humming noise in the high-quality microphone channel. This noise is not present in all files, actually it's present in only some. The source of this noise could not be successfully traced, since a

consistent reproduction pattern could not be designed to trigger the noisy behavior. Some possible sources may be noise induced by the cellphone signal leaking into the HQ mic's preamplifier box, or some ground mismatch, but no source candidate has been deemed the ultimate culprit. Nevertheless, the presence of the noise is scattered throughout the filebase.

Also, especially at the beginning of the process, we had trouble adjusting the volume gain for the four channels. Speakers start with a certain volume and then usually tend to lower that volume by the end of their recording session. That made some channels exhibit very low signal levels, resulting in very low SNRs.

5.4.3 Channel Variabilities

A third difficulty arises because of the nature of switched virtual circuits. In landline and cellular phones alike, a virtual signal circuit is established at the beginning of the call. Because of telephone network use and availability, subsequent phone calls may use slightly different circuits. These circuits will have different physical lengths, attenuation characteristics, etc., which determine the ultimate transfer function of the spoken audio to the telephone recording on the server. In summation, different calls to the server will result in slightly different transfer functions.

5.4.4 Averaging

The good news about all these variabilities, is that if many realizations of the process are conducted (as is the case), when the time comes to get a final answer as to what the channel is, one can assume that the variabilities will vanish. Namely, consider the i^{th}-call's transfer function (or for that matter the transfer function from HQ to telephone of the i^{th} phrase):

$$T_i(\omega) = T_{\text{avg}}(\omega) + \delta_i(\omega)$$

If we want a model that minimizes the deviation from a certain template transfer function (however that template T_{avg} is calculated), we need to choose $T_{\text{avg}}(\omega)$ so that a norm of $\delta_i(\omega)$ is minimized, viz.

$$T_{\text{avg}}(\omega) = \arg\min \|\delta_i(\omega)\|^2 \tag{5.2}$$

$$\begin{aligned} T_{\text{avg}}(\omega) &= \arg\min \left\| T_i(\omega) - T_{\text{avg}}(\omega) \right\|^2 \\ &= \arg\min \sum_{i=0}^{N-1} \left| T_i(\omega) - T_{\text{avg}}(\omega) \right|^2 \\ &= \text{sol} \frac{\partial}{\partial T_{\text{avg}}(\omega)} \left\{ \sum_{i=0}^{N-1} \left| T_i(\omega) - T_{\text{avg}}(\omega) \right|^2 \right\} \end{aligned}$$

What we are saying is that $T_{\text{avg}}(\omega)$ is a number (for each ω) so that the distance from T_i is minimized. Using the result proven in section 4.5.2 expressed in equation 4.2, we obtain that

$$T_{\text{avg}}(\omega) = \frac{1}{N} \sum_{i=0}^{N-1} T_i(\omega)$$

satisfies equation 5.2.

This in turn implies that

$$\frac{1}{N} \sum_{i=0}^{N-1} \delta_i(\omega) = 0$$

The last condition can be augmented so that in the limit

$$\begin{aligned} \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} \delta_i(\omega) &= 0 \\ \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} \left| \delta_i(\omega) \right|^2 &= 0 \end{aligned}$$

This is a condition known as nullification of the ensemble first and second moments, and is ubiquitously found in mean-square optimization.

5.5 Subsolution Recombination

In light of our discoveries, we now present a high-level overview of how to obtain a transfer function estimate, given the TF estimates for each phrase in algorithm 9.

Algorithm 9 Transfer function estimate recombination

1. Estimate TF for each phrase and speaker $T_{p,s}$
2. Average TF over all phrases for a given speaker, $T_s(\omega) = \frac{1}{200}\sum_{p=1}^{200} T_{p,s}(\omega)$
3. Look for outliers, i.e. TFs $T_{p_o,s}$ which deviate from the average T_s more than desired:

$$p_o = \arg\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\frac{T_{p_o}(\omega)}{T_s(\omega)} - 1\right|^2 d\omega > \varepsilon\right\}$$

   we might use other metrics to make our decision, such as

$$p_o = \arg\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\log\left|\frac{T_{p_o}(\omega)}{T_s(\omega)}\right|\right| d\omega > \varepsilon\right\}$$

   which may be combined in what is known as the Itakura-Saito (IS) distance

$$p_o = \arg\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\frac{T_{p_o}(\omega)}{T_s(\omega)} - \log\frac{T_{p_o}(\omega)}{T_s(\omega)} - 1\right|^2 d\omega > \varepsilon\right\}$$

   which may be restated as

$$p_o = \arg\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|e^{v(\omega)} - v(\omega) - 1\right|^2 d\omega > \varepsilon\right\}$$

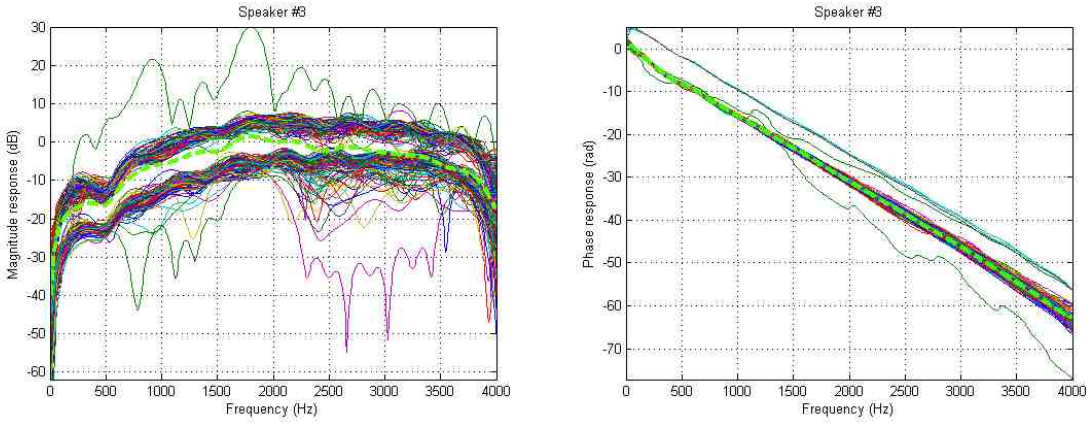
   where

$$v(\omega) = \frac{T_{p_o}(\omega)}{T_s(\omega)}$$

4. Discard those outliers from the calculation of the average, recalculate average and go to step 3 until deviation is less than desired (ε).
5. Repeat the same procedure on the speakers

The threshold parameter ε has to be small enough so that outliers are successfully detected, but large enough so as not to create false positives, since this would greatly reduce the number of actual data segments to use, which in turn is counterproductive with our objective to make $N \to \infty$ so as to get rid of variabilities. In other words, we want variabilities, but not so much as to disturb the whole estimation, i.e. we allow "small" disturbances, where how "small" is mandated by the value of ε.

In addition, there are some characteristics that one can look at to note discrepancies and variabilities. In particular, the phase response (or equivalently the group delay) of a



(a) Magnitude Response (b) Phase Response

Figure 5.2: Frequency response for a recorded speaker

TF tells us a lot about how different sinusoids get re-aligned at the output. For different circuit lengths, the electrical signal carrying the sound goes through different delays. So for example in figure 5.2, we see two clear magnitude response modes (and also several outliers), and at the same time, we see a very dominant phase response and a secondary one, less dominant and less delayed.

Because of the linearity of the Fourier Transform, if we have the transfer functions T_i and their impulse responses t_i so that $t_i \leftrightarrow T_i$, we readily have that $t_{\text{avg}} \leftrightarrow T_{\text{avg}}$ where t_{avg} is the arithmetic mean of the filters' impulse responses, so that we can average the impulse responses instead of going to the frequency domain. This will be very useful since in our case we will be estimating impulse response values and not frequency responses (though the two are just a transform away, doing this computation for each phrase would be very expensive).

5.6 Preprocessing

Segmentation refers to the process of dividing a signal containing multiple utterances into several signals (i.e. files) each one with a single utterance. This is a fundamental step in training the ASR system, since the system is expecting a single utterance at a time and not a constant stream of utterances. The segmentation process can be seen as marking the beginning and ending of each utterance, and in our case is motivated by the accounts described in section 5.3.2.

5.6.1 Synchronization

In order to obtain a meaningful model, the channel and clean utterances must be synchronized. This can be achieved because the channels we are interested in working with are first of all stationary, and secondly and more importantly, the speed at which time advances is constant. Consider the channel defined by

$$S\left\{x\left[n\right]\right\} = x\left[2n\right]$$

This channel is linear, but time-variant, indeed

$$S\left\{x\left[n-\Delta\right]\right\} = x\left[2n-2\Delta\right]$$

Therefore, we can never synchronize $x\left[n\right]$ and $S\left\{x\left[n\right]\right\}$. Thus, time-invariance allows for synchronization of the channel and clean data.

Since the generated files are not synchronized, we need to do some preprocessing to time-align the channel and clean signals.

The best approach to do this is to calculate the cross-correlation of both signals, and find the lag at which the absolute value of the cross-correlation is maximized. Depending on the sign of the lag, we need to delay either the clean or channel data.

Let x be the original clean data and d the channel output. Then, we have that

$$d = \delta_{\Delta} * h * x + v$$

The cross-correlation is thus

$$\begin{aligned} C\left[m\right] &= \sum_{n=-\infty}^{\infty} x\left[n\right] d\left[m+n\right] \\ &= \sum_{n=-\infty}^{\infty} x\left[n\right] \left(\left(\delta_{\Delta} * h * x\right)\left[m+n\right] + v\left[m+n\right]\right) \\ &= \sum_{n=-\infty}^{\infty} x\left[n\right] \left(\left(h * x\right)\left[m+n+\Delta\right]\right) + \sum_{n=-\infty}^{\infty} x\left[n\right] v\left[m+n\right] \end{aligned}$$

If v is white noise, uncorrelated with x, we get

$$\begin{aligned} C[m] &= \sum_{n=-\infty}^{\infty} x[n]\left((h*x)[m+n+\Delta]\right) \\ &= \sum_{n=-\infty}^{\infty} x[n]\left(\sum_{j=-\infty}^{\infty} h[j]\,x[m+n+\Delta-j]\right) \\ &= \sum_{n=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} h[j]\;x[n]\,x[m+n+\Delta-j] \end{aligned}$$

Disregarding for now the effects of the filter h (i.e. inter-symbolic interference (ISI) is minimal), this last expression becomes

$$C[m] = \sum_{n=-\infty}^{\infty} x[n]\,x[m+n+\Delta]$$

which is the signal auto-correlation, which is maximized when $n = m+n+\Delta$. This is attained by setting $m = -\Delta$

The problem with using the cross-correlation approach is the computational burden incurred in calculating the cross-correlation. Remember that for N-samples real signals, the cross-correlation is defined as

$$C[k] = \sum_{n=0}^{N-1} x[n]\,d[n+k]$$

So for each lag value $k \in [-N, N]$, we must calculate $x[n]\,d[n+k]$ for $N-1 \geq n+k \geq 0$, i.e.

$$\begin{aligned} n &\geq \max\{0, -k\} \\ n &\leq \min\{N-1, N-1-k\} \end{aligned}$$

For positive k, we get

$$\begin{aligned} n &\geq 0 \\ n &\leq N-1-k \end{aligned}$$

$$C[k] = \sum_{n=0}^{N-1-k} x[n]\,d[n+k]$$

For example, for $N = 4$

$$\begin{aligned}
C[0] &= x[0]d[0] + x[1]d[1] + x[2]d[2] + x[3]d[3] \\
C[1] &= x[0]d[1] + x[1]d[2] + x[2]d[3] \\
C[2] &= x[0]d[2] + x[1]d[3] \\
C[3] &= x[0]d[3] \\
C[-1] &= x[1]d[0] + x[2]d[1] + x[3]d[2] \\
C[-2] &= x[2]d[0] + x[3]d[1] \\
C[-3] &= x[3]d[0]
\end{aligned}$$

In summation, the cross-correlation goes from $-(N-1)$ to $N-1$. For each lag k there are $N - |k|$ products and $N - 1 - |k|$ additions. Therefore, the computation of all of the cross-correlation values has $\sum_{k=-N+1}^{N-1} N - |k|$ products and $\sum_{k=-N+1}^{N-1} N - 1 - |k|$ additions. This corresponds to

$$\begin{aligned}
\sum_{k=-N+1}^{N-1} N - |k| &= N + 2\sum_{k=1}^{N-1} N - k \\
&= N + 2N(N - 1 - 1 + 1) - 2\sum_{k=1}^{N-1} k \\
&= 2N^2 - N - 2\frac{(N^2 - N)}{2} \\
&= 2N^2 - N - N^2 + N \\
&= N^2
\end{aligned}$$

Thus, calculating the cross-correlation of two long signals is very computational intensive. The usual practice is then to restrict the search for a maximum in the absolute cross-correlation to a couple of seconds.

Chapter 6

Results

Two channels were considered to test the validity of our assumptions: a landline analog telephone line, and a stationary cellular phone. For each of these channels, impulse responses were estimated from clean and channel data, using utterances from the speech database described in section 5.2.

6.1 Landline Estimation

This section describes the methodology and results involved in the estimation of the parameters necessary to build a simulation of an analog landline telephone channel. In the present section some classical techniques in estimation theory, digital signal processing and adaptive filtering theory have been used as well as some new proposed ones.

6.1.1 Methodology

A Recursive Least Squares (RLS) estimation was performed independently for each of the 200 utterances on four different speakers (S1, S2, S3 and S5). This gives a total of 800 training sequences. The RLS algorithm outputs the coefficients of a 40 taps FIR filter. Asymptotically and quasistationarily, this filter is optimal in the mean-square sense in the time domain (i.e. it minimizes the energy of the time domain error signal between the simulated channel and the actual channel).

The input to the estimator is the HQ mic audio, while the reference signal (i.e. desired output) of the estimator are the landline audio files, delayed by 20 samples. This delay allows the filter to be non-minimum phase. The filter order as well as the delay was determined satisfactory after thorough experimentation and fine-tuning.

When the recorded files exhibited the undesirable noise that we encountered at times while recording (which ultimately derived from the HQ mic), the estimations derived from those noisy recordings constitute the "outliers" in the frequency response of those signals,

(i.e. magnitude responses which differ too much from the average). However, as was already mentioned, there were very few noisy recordings.

6.1.2 Ensemble Combination

The impulse response (or conversely, the filter tap coefficients) stemming from each of the 200 utterances within each speaker were averaged[1], rendering h_1, h_2, h_3 and h_5. These in turn, were averaged to render h, the average impulse response over speakers. Several reasons motivate the averaging of the impulse responses:

- Because different sessions represent different connections to the server, and therefore a different signal path, the ideal case would be to estimate the impulse response for each session within a speaker, and then average all the sessions. This takes care of variations in the signal path. This is the intra-session average. This was not calculated.
- Since RLS has a very fast convergence time, it may not be beneficial to start the estimation of one utterance using the estimation for the previous utterance in the same session. Thus, by averaging all utterances within a speaker, we get an inter-session ensemble average. This was calculated for each speaker and is the mentioned h_1, h_2, h_3 and h_5.
- Averaging is a consistent way of combining impulse responses estimated from different realizations through the same channel. This is referred to as the ensemble average, and corresponds to h.

In order for the estimation to make sense, some preprocessing had to be done

1. The naming convention for the files saved locally (HQ mic, VoIP mic) and the server-side recordings (landline, cellphone) is different. For the former we have `PHRASE32.PCM`, and for the latter we have `0005_9_32_1.AU`. Therefore, some pairing up had to be performed.
2. A MATLAB script reads the HQ mic file (saved as 44.1 kHz raw PCM data) and the landline file (saved as 8 kHz AU).
3. Resampling takes place to have both signals at the same sampling rate.

[1] In this section, average refers to the procedure described in 5.4.4 and 5.5.

Algorithm 10 Synchronization algorithm

1. `[xc, lags] = xcorr(hq_data, channel_data)`
2. `[max_corr, max_corr_idx] = max(abs(xc))`
3. Calculate the value of the synchronization lag as `sync = lags(max_corr_idx)`
4. Depending on the sign of `sync`, we have to add silence to either `hq_data` or `channel_data` to get the two files to be time aligned

4. It is vital that the signals are synchronized uniformly. In general, the files have some silence before and after, and this amount of silence is different in the HQ mic file and the landline file. In order for the estimation to make sense, the delay before the signal in both files (i.e. within each utterance, with reference to the HQ mic) must be identical. The optimal way to synchronize the signals is described in the next section.

6.1.3 Synchronization

In order to synchronize both signals, the cross-correlation of the signals is taken up to two seconds. Then, the maximum of the absolute value of the cross-correlation is the delay that one must add (or subtract, depending on the sign) from one signal or the other, so that both are aligned properly. The absolute value is necessary to ignore 180° phase changes.

The second step is truncating the longer signal so that it has the same length as the shorter one.

6.1.4 Results

In the next figures, some characteristic curves are plotted, including the magnitude and phase response for each utterance, as well as the average within a speaker (in dotted bright green line). Also, the impulse response, magnitude and phase responses and group delay is plotted for each of the four speakers as well as their average.

6.1.4.1 Individual speakers

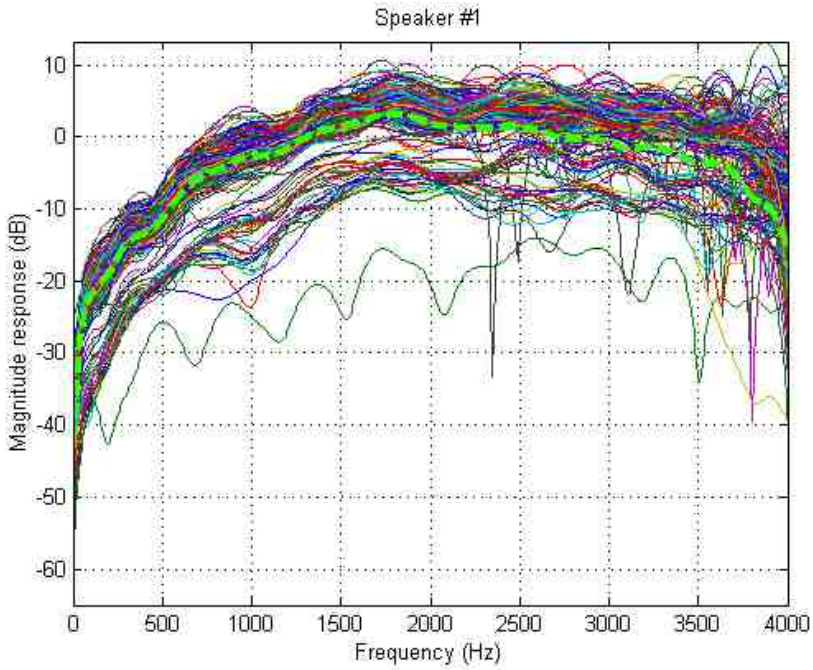


Figure 6.1: Magnitude Response for Speaker #1

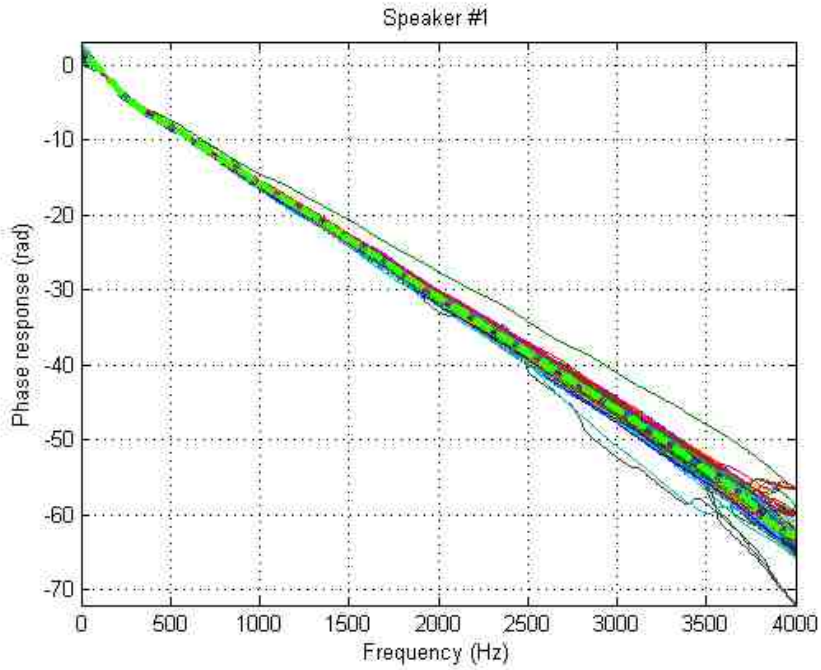


Figure 6.2: Phase Response for Speaker #1

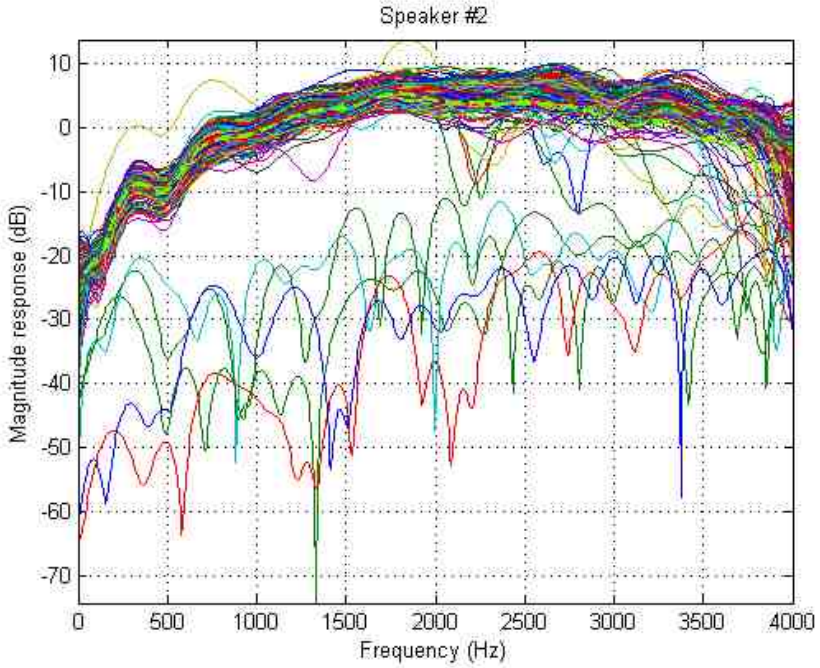


Figure 6.3: Magnitude Response for Speaker #2

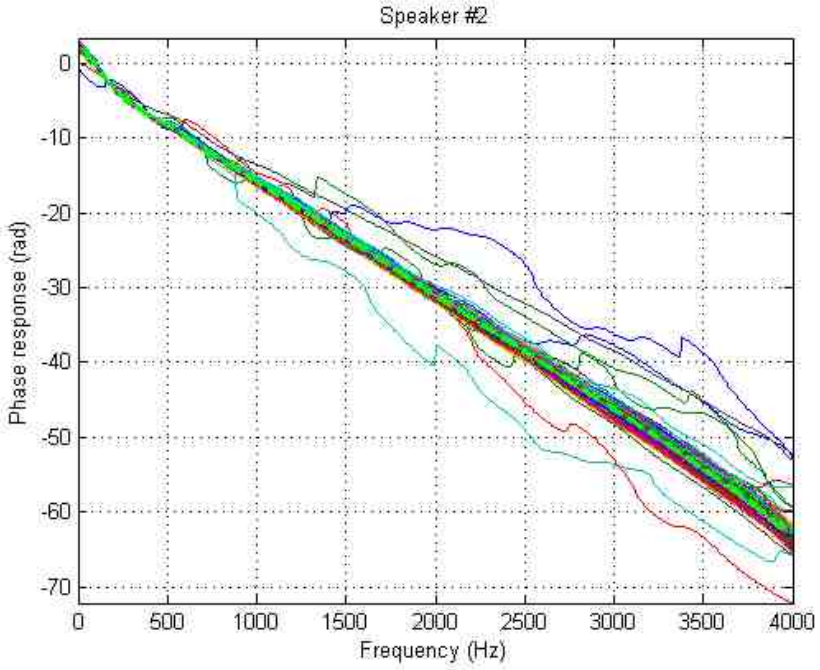


Figure 6.4: Phase Response for Speaker #2

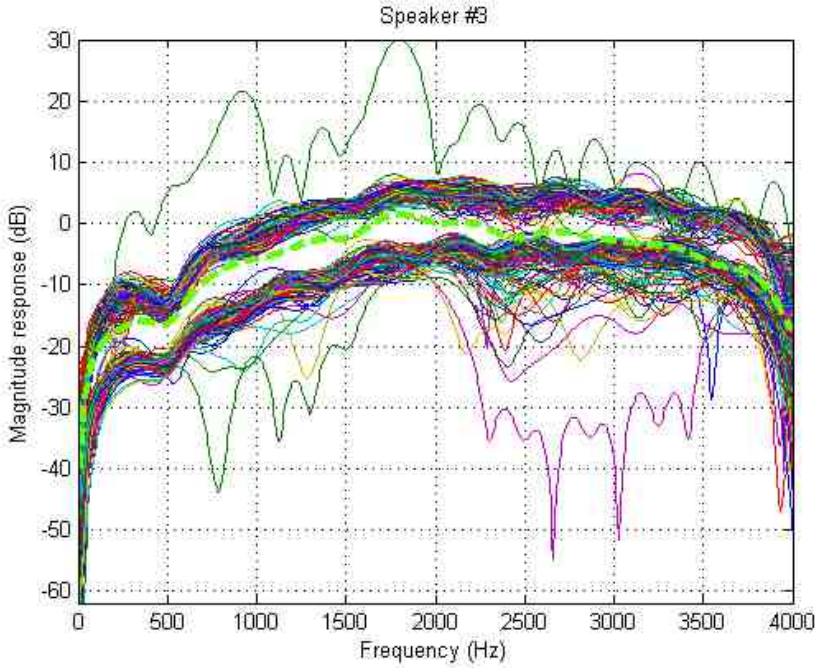


Figure 6.5: Magnitude Response for Speaker #3

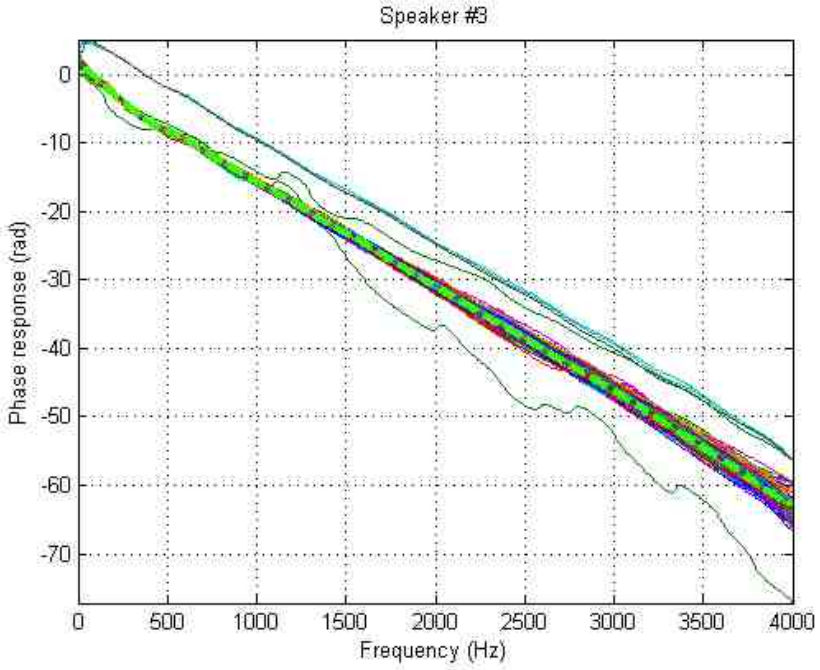


Figure 6.6: Phase Response for Speaker #3

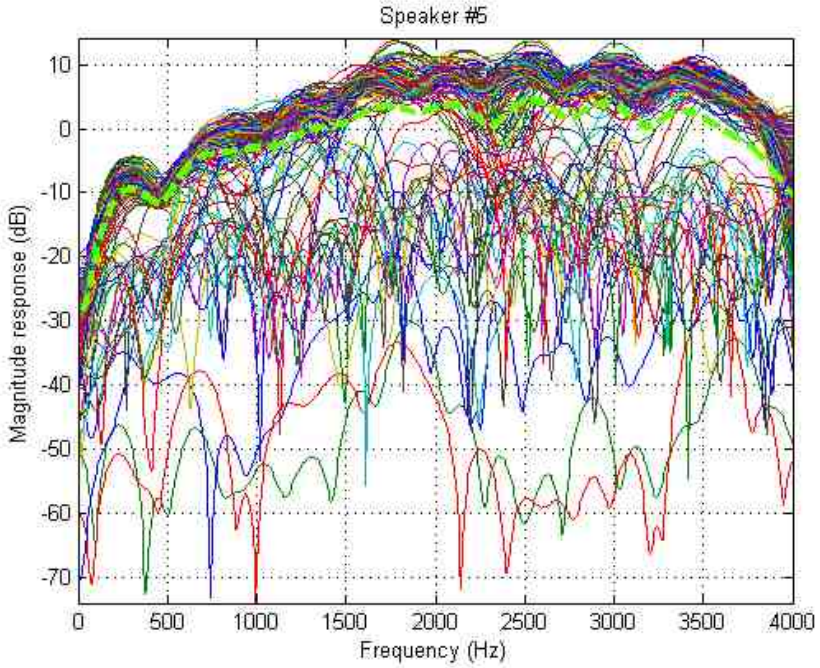


Figure 6.7: Magnitude Response for Speaker #5

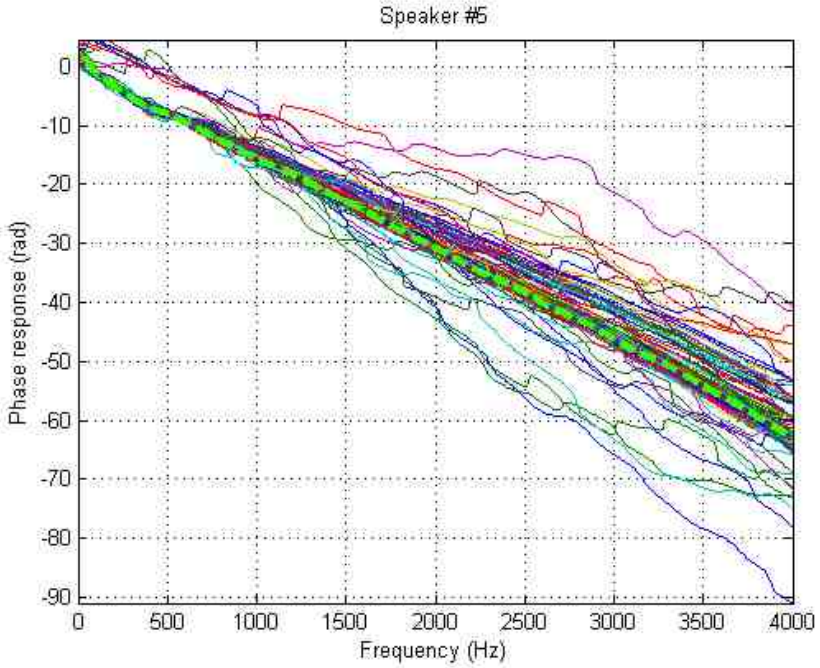


Figure 6.8: Phase Response for Speaker #5

6.1.4.2 Ensemble average

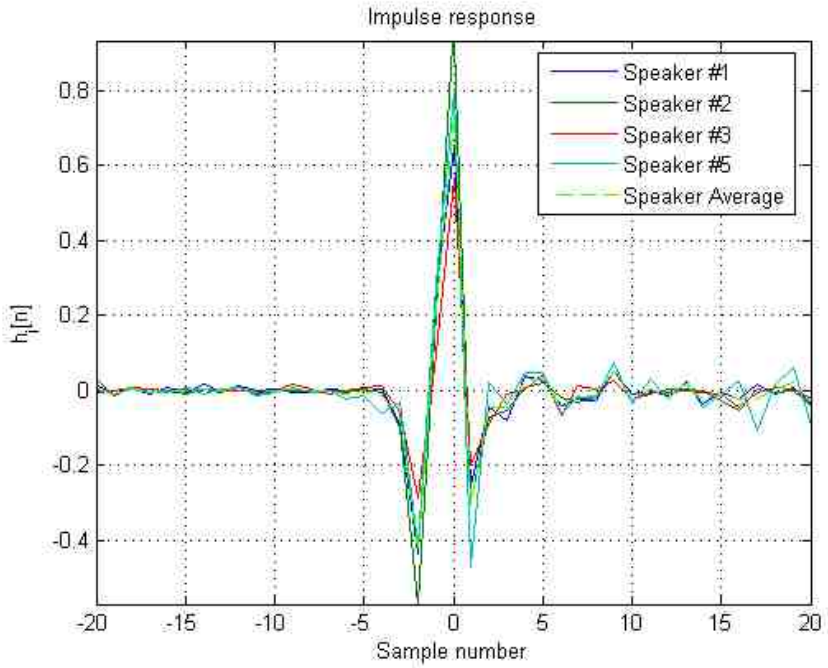


Figure 6.9: Impulse Responses

The value of the impulse response for samples $n = -20$ to $n = 20$ is given in table 6.1, read from left to right, top to bottom.

Table 6.1: Landline Impulse response values (in units of 10^{-2})

Sample number						
from	to	$h\,[n]$				
-20	16	1.1305	−0.8686	0.7354	−0.3506	−0.0024
-15	-11	−0.0254	0.2912	−0.1104	0.7590	−0.6458
-10	-6	0.0396	0.4766	−0.0205	−0.0704	−0.7148
-5	-1	−0.1998	−1.2438	−6.8629	−42.6139	21.5280
0	4	73.6985	−30.4845	−4.5183	−4.4659	2.4641
5	9	3.3840	−4.6331	−1.5656	−1.4442	5.1737
10	14	−2.1328	0.3459	−0.9724	1.2186	−2.0582
15	19	−1.1025	−2.3496	−2.2935	0.2805	1.9251
20		−4.7194				

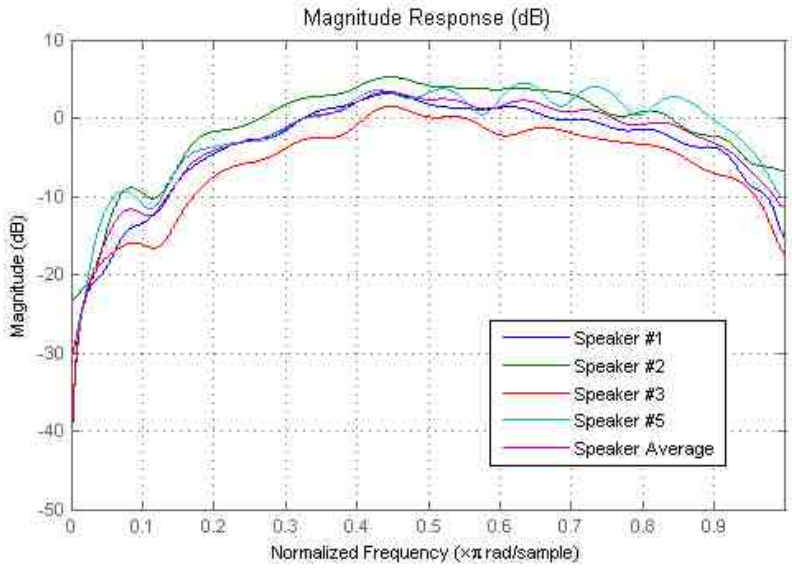


Figure 6.10: Magnitude Responses

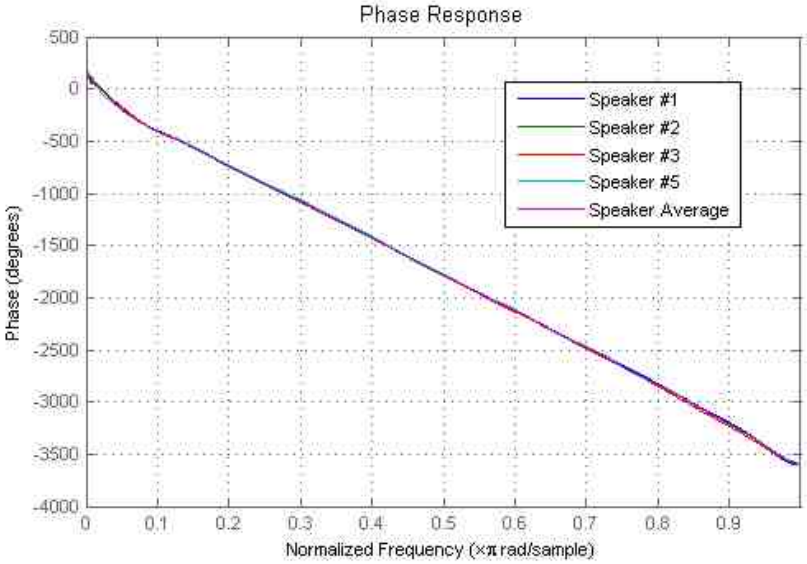


Figure 6.11: Phase Responses

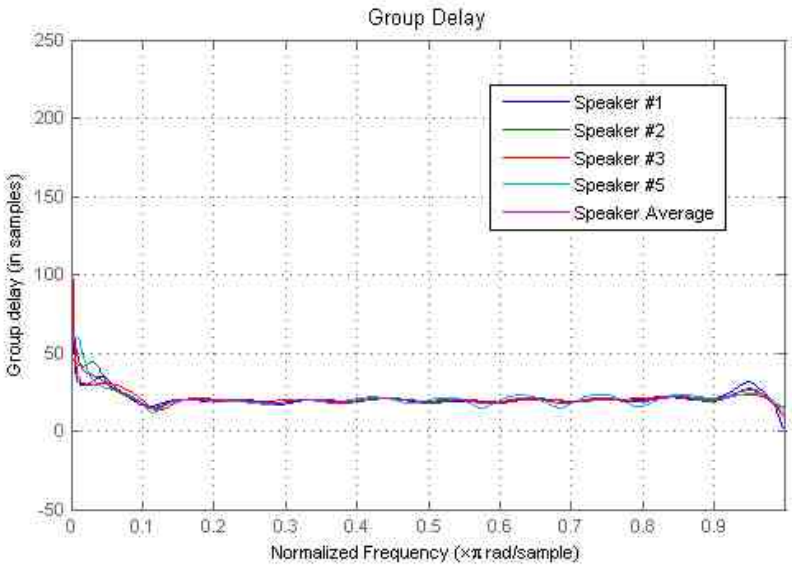


Figure 6.12: Group Delay

Since the filter is an asymmetric FIR filter the realization is quite straightforward. A block diagram is shown in figure 6.13.

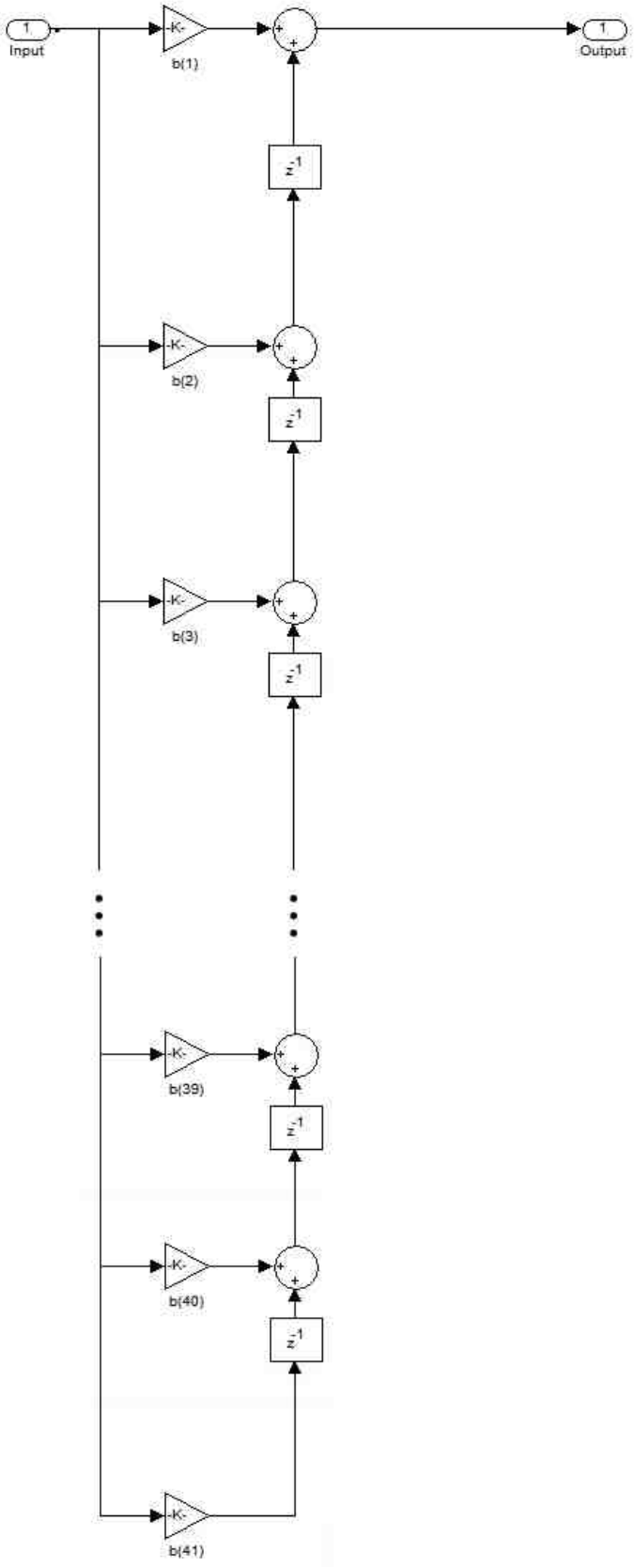


Figure 6.13: Filter realization

6.1.4.3 Subjective Evaluation

In order to evaluate how good the landline telephone channel estimation is, we need to compare it in some meaningful way to the reference landline signal. For this purpose, an experiment was devised in which five subjects independently tried to identify three different versions of sixteen utterances presented to them in random order. The versions in question were the high-quality mic recording (i.e. clean or original speech), the landline telephone signal (the channel to simulate) and the high-quality recording passed through the filter which we estimated to be close in its frequency response to the landline channel.

For each of these versions, the subjects had to assign the played version to what they thought it actually was. As mentioned before, the three versions of each utterance were presented in random order. Thus, one guarantees that the responses will not be biased. The results of the 3×16 evaluations are arranged in what is known as a confusion matrix. Each entry M_{ij} of the matrix is a number which represents the number of times the subject indicated he though the signal was j, when the signal was i. Since we want our estimation to be indistinguishable from the actual landline channel, and very distinguishable from the HQ recording, the ideal confusion matrix will be

$$I = \begin{bmatrix} 16 & 0 & 0 \\ 0 & 8 & 8 \\ 0 & 8 & 8 \end{bmatrix}$$

The first row/column corresponds to the HQ recording, the second to the real landline channel and the third to the simulation. The row indicates the actual source and the column what was indicated. This way, one recognizes the signal as broadband if and only if it is broadband, and with equal probability one confuses the landline and simulation.

The matrices resulting from the five subjects (L_1, L_2, L_3, L_4 and L_5) are shown below

$$L_1 = \begin{bmatrix} 16 & 0 & 0 \\ 0 & 16 & 0 \\ 0 & 0 & 16 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 15 & 1 & 0 \\ 0 & 4 & 12 \\ 0 & 13 & 3 \end{bmatrix}$$

$$L_3 = \begin{bmatrix} 15 & 1 & 0 \\ 0 & 10 & 6 \\ 1 & 5 & 10 \end{bmatrix}$$

$$L_4 = \begin{bmatrix} 16 & 0 & 0 \\ 0 & 9 & 7 \\ 0 & 7 & 9 \end{bmatrix}$$

$$L_5 = \begin{bmatrix} 15 & 0 & 1 \\ 1 & 0 & 15 \\ 0 & 16 & 0 \end{bmatrix}$$

The next step after the collection of the opinions from the test subjects, is coming up with a function that will compare the simulation. A way to do this is using matrix norms. Several options are possible, of which some are noteworthy:

1. Measure $m = \frac{\|L_1-I\|+\|L_2-I\|+\|L_3-I\|+\|L_4-I\|+\|L_5-I\|}{5\|I\|}$. This averages the percentage deviation from the ideal confusion matrix. The problem with this approach is that it doesn't contemplate the inherent asymmetry between not distinguishing landline from simulated and separating simulated from HQ. The resulting value was $m_1 = 0.5920$, where $m = 0$ would be ideal.

2. Use a weighted norm. For this purpose, an auxiliary matrix X is used as a way of changing the basis of the vector space in which the measurement is done. A sensible choice might be

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{6.1}$$

and the measure would then be

$$m = \frac{\sum_{i=1}^{5} \left\| X^T L_i X - I \right\|}{5 \left\| I \right\|}$$

This measure has the property of weighting the HQ differently than the Landline and Simulated. But the problem is that it "counts twice" every error between classifying Landline as Simulated and vice versa. This can be seen from the fact that it has two identical rows, this leads to a rank-deficient matrix, and it's traditional consequences from linear algebra (it's column space does not span the entire space, it has an with eigenvalue 0, it has only two independent directions, etc.). The resulting value was $m_2 = 0.0528$ where $m = 0$ is the optimum

3. Similarly, one can weight with a matrix such as

$$X = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

In this case the ideal confusion matrix is

$$I_2 = \begin{bmatrix} 16 & 0 \\ 0 & 16 \end{bmatrix}$$

and the measure

$$m = \frac{\sum_{i=1}^{5} \left\| X^T L_i X - I_2 \right\|}{5 \left\| I_2 \right\|}$$

In this case the value was also $m_3 = 0.0528$. Note that it is important that the notion of "ideal confusion matrix" be invariant under the basis change performed by the weighting matrix X, i.e. $X^T I X = I$. In the $\mathbb{R}^n \to \mathbb{R}^m$ with $n \neq m$ case, it is necessary and sufficient to have $X^T I X = I_2$, i.e. the ideal matrices in two bases are related by the transform matrix X. Another desirable property of the matrix X is that it has unit norm.

Other choices for the weighting matrix can be derived for example from the following template

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha & 1-\alpha \\ 0 & 1-\alpha & \alpha \end{bmatrix}$$

The interpretation of this metric is fairly simple: the lower the number (i.e. the closer it is to 0), the better the simulation performs at emulating the landline channel and separating itself from the high-quality reference signal. Thus, we find that using reasonable weighting, the measure is around 95%, which is more than appropriate for our purposes.

All opinion matrices are of the form

$$\Omega = \begin{bmatrix} a & b & 16-a-b \\ c & d & 16-c-d \\ e & f & 16-e-f \end{bmatrix}$$

with $16 \geq a, b, c, d \geq 0$ and $a+b, c+d, e+f \leq 16$

The maximum individual distance is $\Delta = \max \left\| X^T \Omega X - I \right\|$. By inspection, and using X as defined in equation 6.1, any matrix of the form

$$\Omega_a = \begin{bmatrix} 0 & a & 16-a \\ 16 & 0 & 0 \\ 16 & 0 & 0 \end{bmatrix}$$

The maximum individual matrix distance is $\Delta = 33.9411$[2] for 16 observations. In the worst case, we have that all five listeners heard according to matrix Ω_a, so that the maximum metric m_2 becomes

$$\begin{aligned} m_2^{\max} &= \frac{\sum_{i=1}^{5} \max \left\| X^T L_i X - I \right\|}{5 \left\| I \right\|} \\ &= \frac{5\Delta}{5 \left\| I \right\|} \\ &= \frac{\Delta}{\left\| I \right\|} \end{aligned}$$

By using the largest singular value as the matrix norm we get

$$m_2^{\max} = 2.1213$$

[2]We must remark that by using either the Frobenius norm ($\sqrt{\sum \operatorname{tr}\left(X^H X\right)}$) or the matrix spectral norm (the matrix's largest singular value) we get the same result for Δ, since the resulting matrix is of rank one (there is only one nonzero singular value). In general this is not true, what is true is that the Frobenius norm is greater or equal to the spectral norm.

so that our distance in actual percentage is

$$\begin{aligned} m_2\% &= 100\% \times \frac{m_2}{m_2^{\max}} \\ &= 100\% \times \frac{0.0528}{2.1213} \\ &= 2.489\% \end{aligned}$$

A random confusion matrix would be

$$\rho = \frac{1}{3}\begin{bmatrix} 16 & 16 & 16 \\ 16 & 16 & 16 \\ 16 & 16 & 16 \end{bmatrix}$$

whose distance is

$$\begin{aligned} m_2^{\rho} &= 1 \\ m_2^{\rho}\% &= 47.141\% \end{aligned}$$

These numbers, illustrated in figure 6.14, show that the simulation is very effective from the point of view of human perception. The next step then is to measure the simulation performance objectively (i.e. without resorting to listeners).

6.1.5 Conclusions

The channel estimates derived from all the 800 utterances are consistent: the magnitude responses do not differ in their overall behavior, but rather there are subtle variations at different frequencies. These variations are expected since the estimates are drawn from non-stationary, finite-length data, which inherently has noise, etc., so the conditions can never really be ideal. In addition, the larger the order of the filter, the more ripple the magnitude response is bound to have.

All in all, the magnitude response exhibits a bandpass characteristic, blocking basically all DC energy as well as all energy at high frequency (i.e. half the sampling frequency, or in this case 4 kHz). The frequency response in between the extremes is more or less flat with a ± 3 dB range going from around 0.3π to 0.8π, or about 2000 Hz. Thus, this estimation corresponds with what is expected from a landline phone channel.

In addition, the phase response is almost linear everywhere, and is only distorted at the extremes (since they have a large attenuation). When plotting the group delay curve,

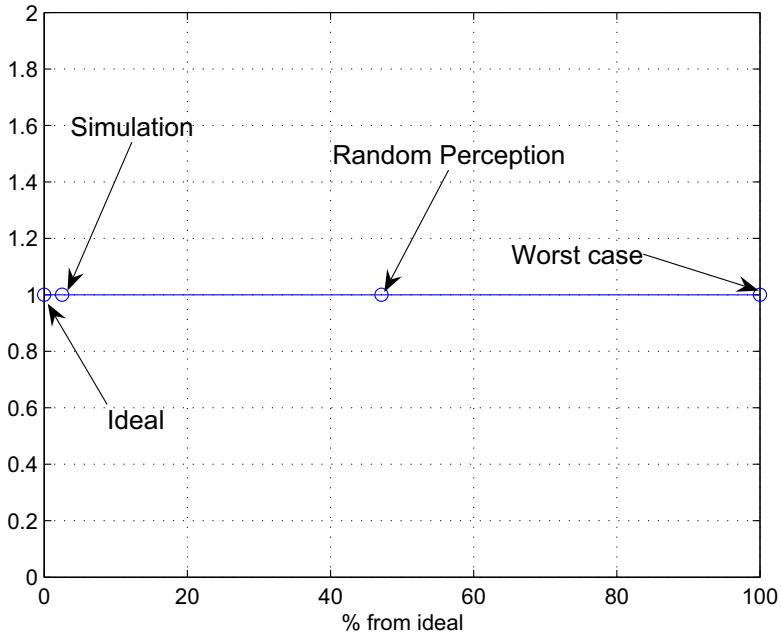


Figure 6.14: Landline simulation Mean opinion score evaluation

one sees that, as expected, the group delay in the middle band is 20 samples. This is the delay introduced by delaying the desired output signal in order to capture the non-minimum phase components of the impulse response.

The impulse response presents some energy before its peak (which corresponds to the zero-th sample). This energy corresponds to anti-causal components (or conversely, maximum phase components when delayed). Thus, we can not avoid delaying the desired output since we would not capture these components.

An experiment was executed to determine the validity of the simulation from the perceptual point of view. Several metrics are proposed to measure the performance of the simulation using the opinions of the test subjects. The results of this evaluation show that the estimation of the impulse response (or conversely the transfer function of the landline channel), represents an excellent simulation of the actual channel.

6.1.6 Future Work

Though the synchronization computational effort is not enormous since we are considering only lags less than two seconds, a better synchronization detection algorithm would be beneficial to speed up the estimation process. For that purposes, fast VAD algorithms

could be used. The only requirement is that the VAD has a resolution of one sample, which is very difficult to achieve since they usually work on a frame-by-frame basis.

Also, since this is a minimization of the error energy in the time domain, we need to compare the performance of the RLS estimation with that obtained with the algorithms introduced in sections 4.7.2 and 4.9 (Least Mean-Square Cepstrum and especially the Least Feature-Error Norm). As for testing purposes, more exhaustive tests may be performed, either by increasing the number of test subjects or the number of utterances to which they are exposed.

6.2 Stationary Cellphone Estimation

This section describes the methodology and results involved in the estimation of the parameters necessary to build a simulation of a stationary cellular telephone channel.

6.2.1 Methodology

The methodology was identical to the one described for the landline telephone channel estimation in section 6.1, i.e. a Recursive Least Squares (RLS) estimation performed independently for each of the 200 utterances on four different speakers (#1, #2, #3 and #5). The same considerations regarding combination and synchronization as well as implementation apply here.

6.2.2 Results

In the next figures, some characteristic curves are plotted, including the magnitude and phase response for each utterance, as well as the average within a speaker (in dotted bright green line). Also, the impulse response, magnitude and phase responses and group delay is plotted for each of the four speakers as well as their average.

6.2.2.1 Individual speakers

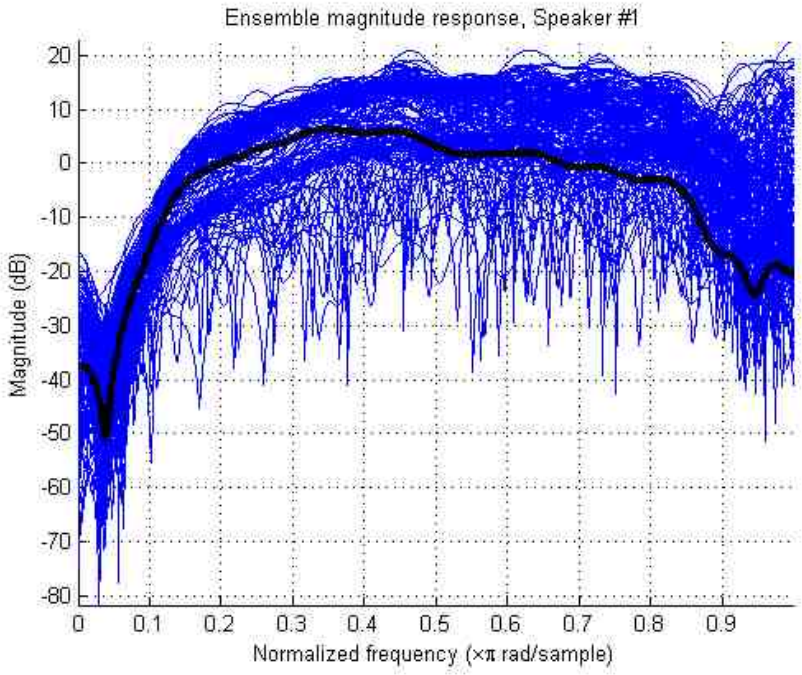


Figure 6.15: Magnitude Response for Speaker #1

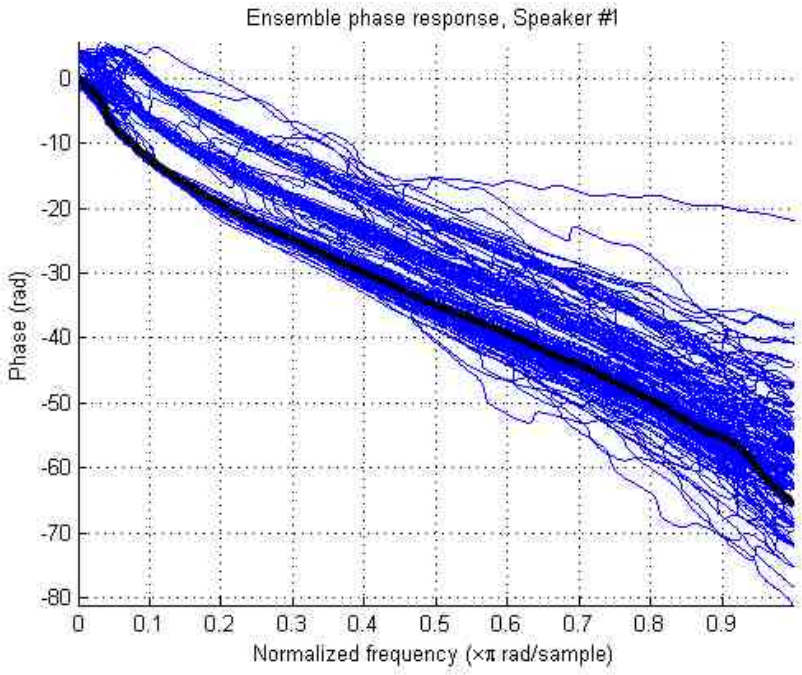


Figure 6.16: Phase Response for Speaker #1

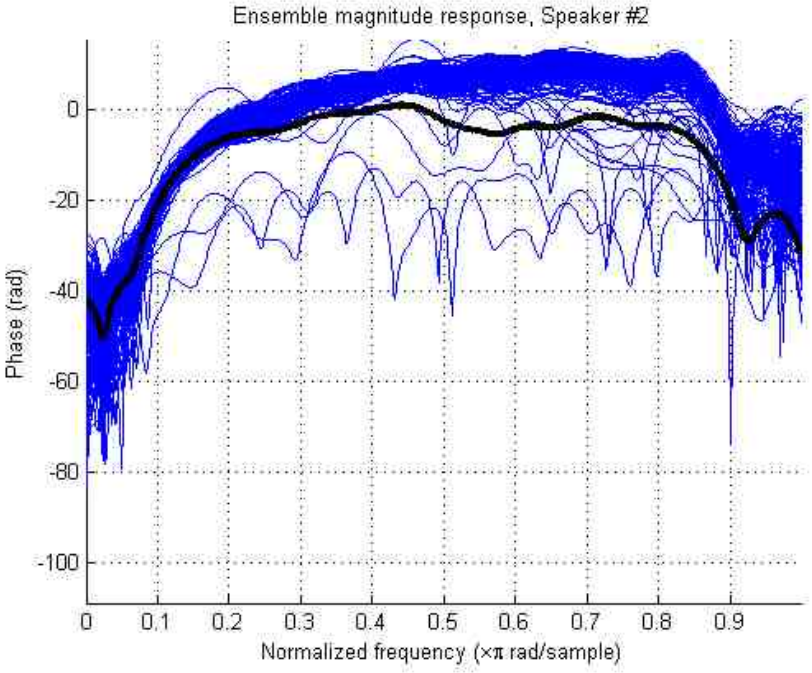


Figure 6.17: Magnitude Response for Speaker #2

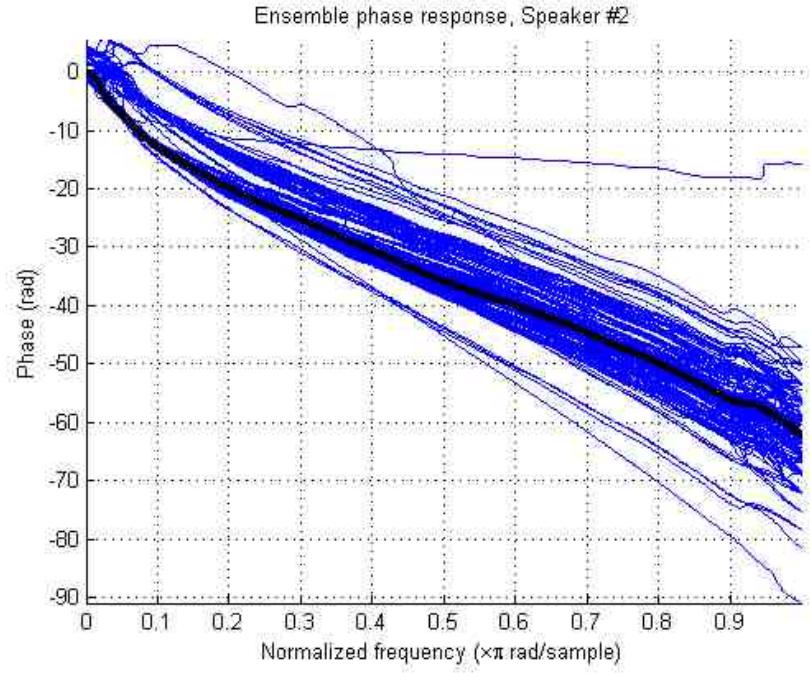


Figure 6.18: Phase Response for Speaker #2

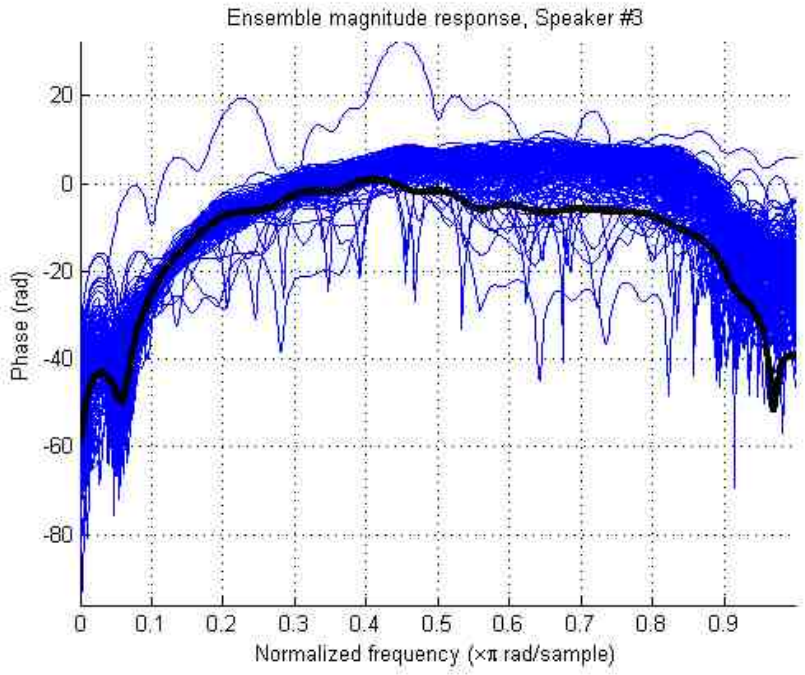


Figure 6.19: Magnitude Response for Speaker #3

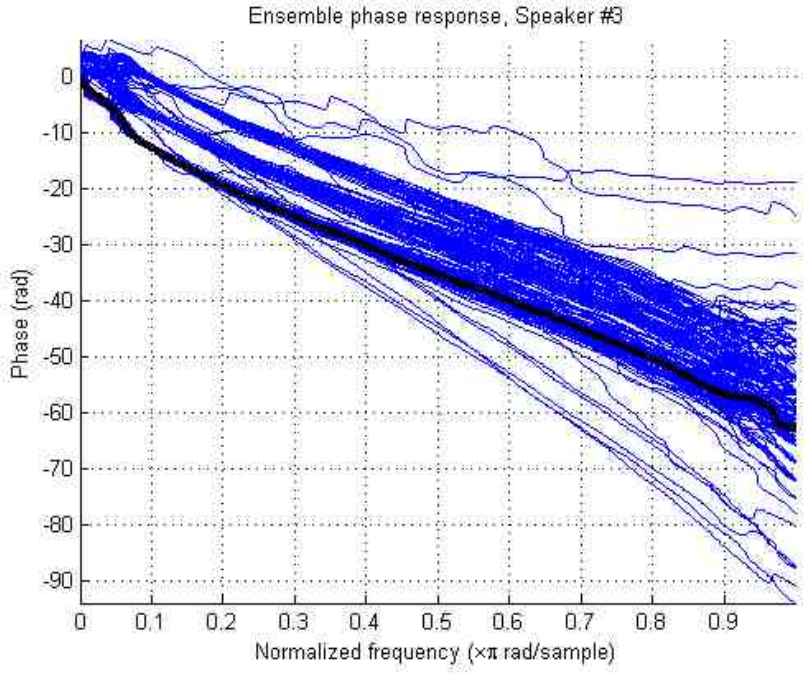


Figure 6.20: Phase Response for Speaker #3

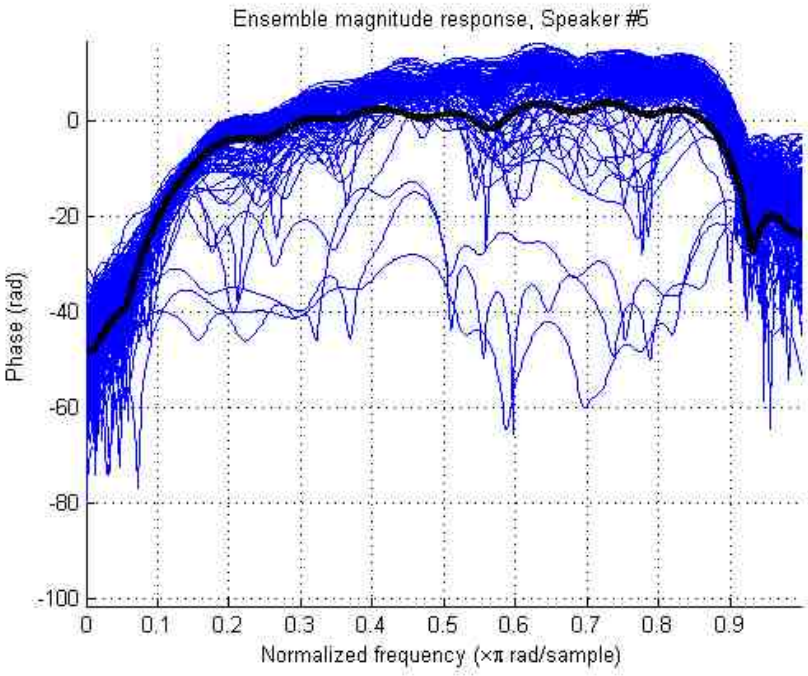


Figure 6.21: Magnitude Response for Speaker #5

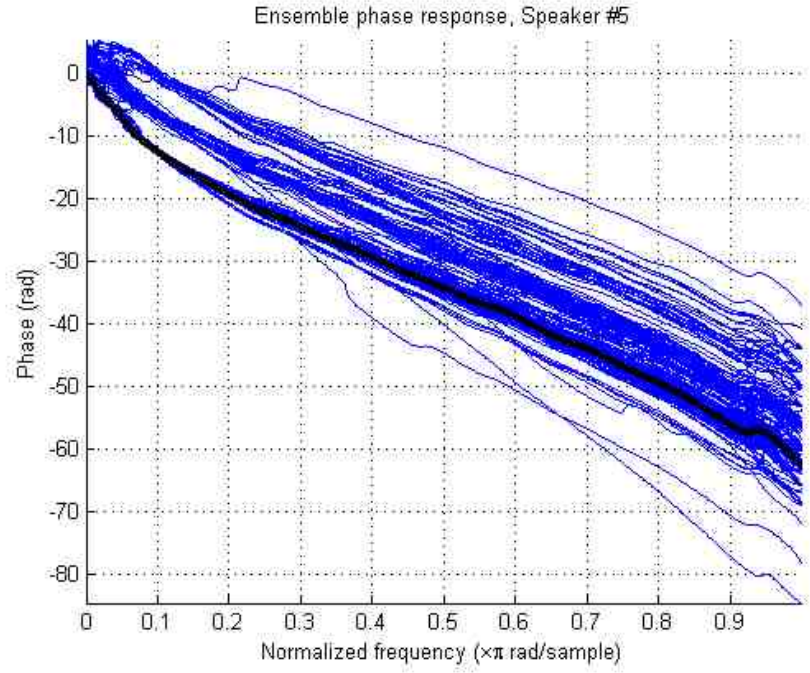


Figure 6.22: Phase Response for Speaker #5

Table 6.2: Cellphone Impulse response values (in units of 10^{-2})

Sample number						
from	to	$h\,[n]$				
-20	16	0.3277	−0.3686	−0.0389	−0.1160	−0.4203
-15	-11	0.0135	0.6720	0.4505	0.3187	−0.2465
-10	-6	−0.6648	−1.7293	−3.1243	−1.9869	29.9319
-5	-1	13.7597	−57.8452	−12.6489	17.4023	7.0242
0	4	24.1227	−6.4032	−2.7558	2.6351	−1.4223
5	9	−1.2178	−7.1823	1.0425	−3.1873	−0.7353
10	14	0.6184	0.9151	1.8110	0.9509	2.5319
15	19	−3.1872	−0.0161	1.2617	−0.3068	−0.7923
20		−0.3140				

6.2.2.2 Ensemble average

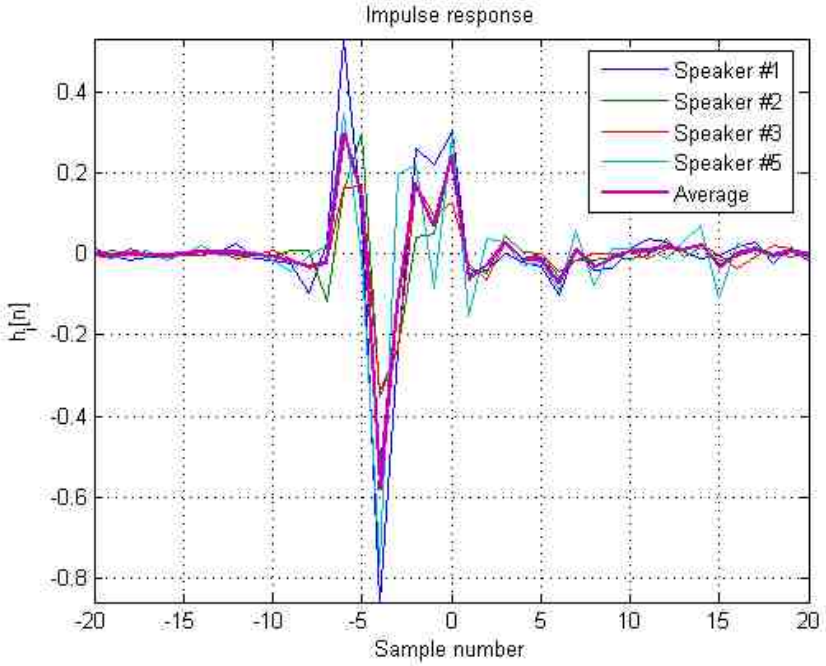


Figure 6.23: Impulse Responses

The value of the impulse response for samples $n = -20$ to $n = 20$ is given in table 6.2, read from left to right, top to bottom.

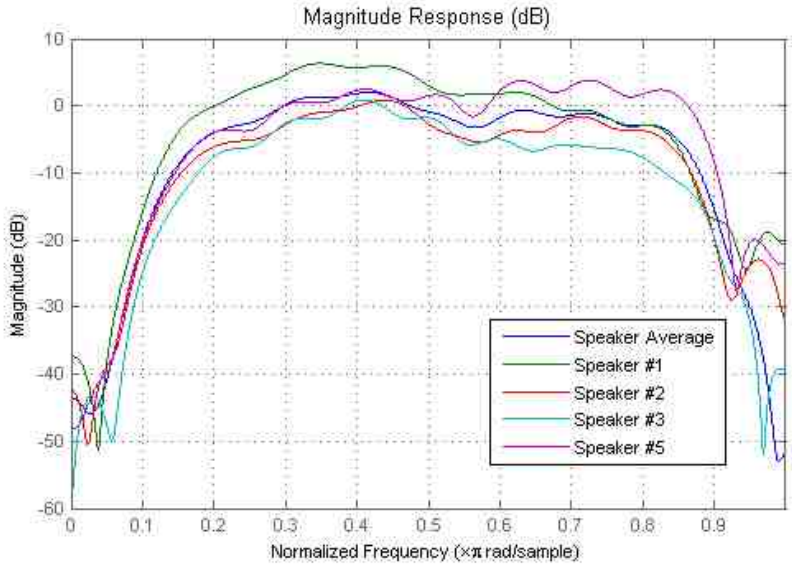


Figure 6.24: Magnitude Responses

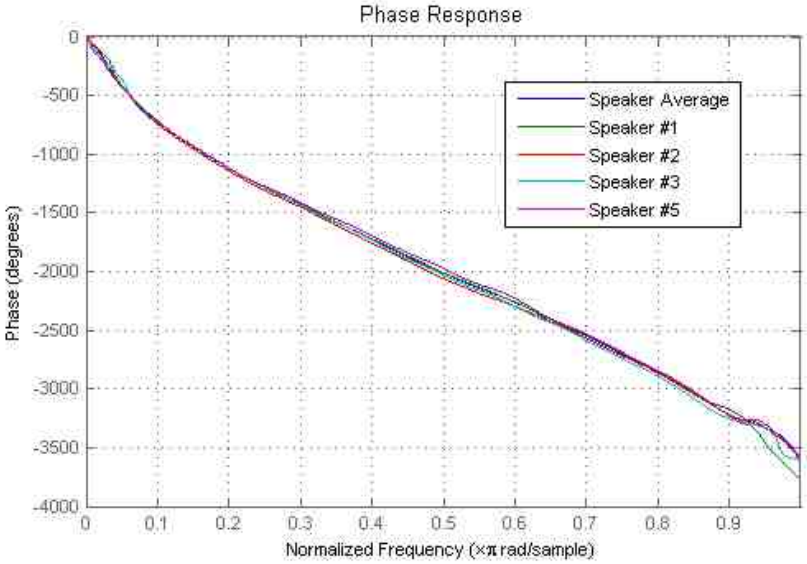


Figure 6.25: Phase Responses

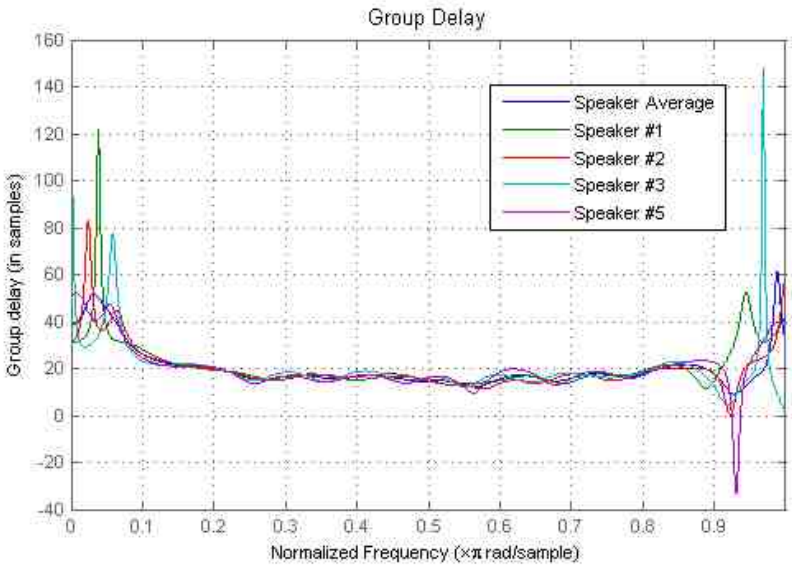


Figure 6.26: Group Delay

6.2.3 Subjective Evaluation

Just as for the landline telephone channel, a perceptual (subjective) test was performed for the cellphone simulation. The details are quite similar to those in section 6.1.4.3, the differences are explained in the following paragraphs.

An experiment was devised in which three subjects independently tried to identify three different versions of eleven utterances presented to them in random order. The versions in question were the high-quality mic recording (i.e. clean or original speech), the cellphone signal (the channel to simulate) and the high-quality recording passed through the filter which we estimated to be close in its frequency response to the stationary cellular channel.

For each of these versions, the subjects had to assign the played version to what they thought it actually was. As mentioned before, the three versions of each utterance were presented in random order. Thus, one guarantees that the responses will not be biased. The results of the 3×11 evaluations are arranged in what is known as a confusion matrix. Each entry M_{ij} of the matrix is a number which represents the number of times the subject indicated he though the signal was j, when the signal was i. Since we want our estimation to be indistinguishable from the actual cellphone channel, and very distinguishable from

the HQ recording, the ideal confusion matrix will be

$$I = \begin{bmatrix} 11 & 0 & 0 \\ 0 & 5.5 & 5.5 \\ 0 & 5.5 & 5.5 \end{bmatrix}$$

The first row/column corresponds to the HQ recording, the second to the real cellphone channel and the third to the simulation. The row indicates the actual source and the column what was indicated. This way, one recognizes the signal as broadband if and only if it is broadband, and with equal probability one confuses the cellphone and simulation.

The matrices resulting from the three subjects (L_1, L_2, L_3) are shown below

$$L_1 = \begin{bmatrix} 11 & 0 & 0 \\ 0 & 10 & 1 \\ 0 & 1 & 10 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 11 & 0 & 0 \\ 0 & 2 & 9 \\ 0 & 9 & 2 \end{bmatrix}$$

$$L_3 = \begin{bmatrix} 10 & 0 & 1 \\ 0 & 7 & 4 \\ 0 & 4 & 7 \end{bmatrix}$$

As with the landline estimation, we proceed to evaluate the matrix norms:

1. Measure $m = \frac{\|L_1 - I\| + \|L_2 - I\| + \|L_3 - I\|}{3\|I\|}$. This averages the percentage deviation from the ideal confusion matrix. The problem with this approach is that it doesn't contemplate the inherent asymmetry between not distinguishing cellphone from simulated and separating simulated from HQ. The resulting value was $m_1 = 0.4139$ for Frobenius norm and $m_1 = 0.5787$ for maximum singular value.

2. Use a weighted norm. For this purpose, an auxiliary matrix X is used as a way of changing the basis of the vector space in which the measurement is done. A sensible choice might be

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

and the measure would then be

$$m = \frac{\sum_{i=1}^{3} \left\| X^T L_i X - I \right\|}{3 \left\| I \right\|}$$

This measure has the property of weighting the HQ differently than the cellphone and simulated. But the problem is that it "counts twice" every error between classifying cellphone as Simulated and vice versa. This can be seen from the fact that it has two identical rows, this leads to a rank-deficient matrix, and it's traditional consequences from linear algebra (it's column space does not span the entire space, it has an eigenvector with eigenvalue 0, it has only two independent directions, etc.). The resulting value was $m_2 = 0.0262$ for Frobenius norm and $m_2 = 0.0371$ for largest singular value.

The worst case scenario for a matrix would be to look like

$$\Omega = \begin{bmatrix} 0 & 11 & 0 \\ 11 & 0 & 0 \\ 11 & 0 & 0 \end{bmatrix}$$

The maximum individual matrix distance is $\Delta = 23.3345$[3] for 11 observations. In the worst case, we have that all three listeners heard according to matrix Ω, so that the maximum metric m_2 becomes

$$\begin{aligned} m_2^{\max} &= \frac{\sum_{i=1}^{3} \max \left\| X^T L_i X - I \right\|}{3 \left\| I \right\|} \\ &= \frac{3\Delta}{3 \left\| I \right\|} \\ &= \frac{\Delta}{\left\| I \right\|} \end{aligned}$$

By using the largest singular value as the matrix norm we get

$$m_2^{\max} = 2.1213$$

[3]We must remark that by using either the Frobenius norm ($\sqrt{\sum \mathrm{tr}\,(X^H X)}$) or the matrix spectral norm (the matrix's largest singular value) we get the same result for Δ, since the resulting matrix is of rank one (there is only one nonzero singular value). In general this is not true, what is true is that the Frobenius norm is greater or equal to the spectral norm.

so that our distance in actual percentage is

$$\begin{aligned} m_2\% &= 100\% \times \frac{m_2}{m_2^{\max}} \\ &= 100\% \times \frac{0.0371}{2.1213} \\ &= 1.749\% \end{aligned}$$

A random confusion matrix would be

$$\rho = \frac{1}{3}\begin{bmatrix} 11 & 11 & 11 \\ 11 & 11 & 11 \\ 11 & 11 & 11 \end{bmatrix}$$

whose distance is

$$\begin{aligned} m_2^{\rho} &= 1 \\ m_2^{\rho}\% &= 47.141\% \end{aligned}$$

Thus our cellphone simulation performs subjectively better than our landline simulation (1.7% vs. 2.5% distance).

6.2.4 Conclusions

Similar conclusions to those regarding the landline telephone channel estimation apply here. The results of this evaluation show that the estimation of the impulse response (or conversely the transfer function of the landline channel), represents an excellent simulation of the actual channel.

6.3 Analog Channels' Objective Evaluation

To perform an objective evaluation, we took several speakers and their the 200 input-output realizations and the estimated average channels h_{LL} and h_{CP} and calculate several distances: figures 6.27 through 6.30 depict the value of each distance between the landline simulation and landline and between the high quality recording and the landline, for comparison. Also in the figure a horizontal line depicting the average distance value for both series is shown.

1. RMS distance

$$\vartheta_{\mathbf{RMS}}(h) = \frac{1}{N}\sum_{i=0}^{N-1}\frac{1}{L_i}\sum_{n=0}^{L_i}\left|(x_i * h)[n] - d_i[n]\right|^2$$

2. Kullback-Liebler distance (Information divergence). For this we take the spectral pdf $Y_i = \mathbf{FFT}\{(x_i * h)[n]\}$ and compare it with that of $D_i = \mathbf{FFT}\{d_i[n]\}$

$$D(p,q) = \sum p_i \log_2 \frac{p_i}{q_i}$$

$$\bar{D}_i[n] = \frac{|D_i[n]|}{\sum_{i=0}^{L_i-1}|D_i[i]| + \delta}$$

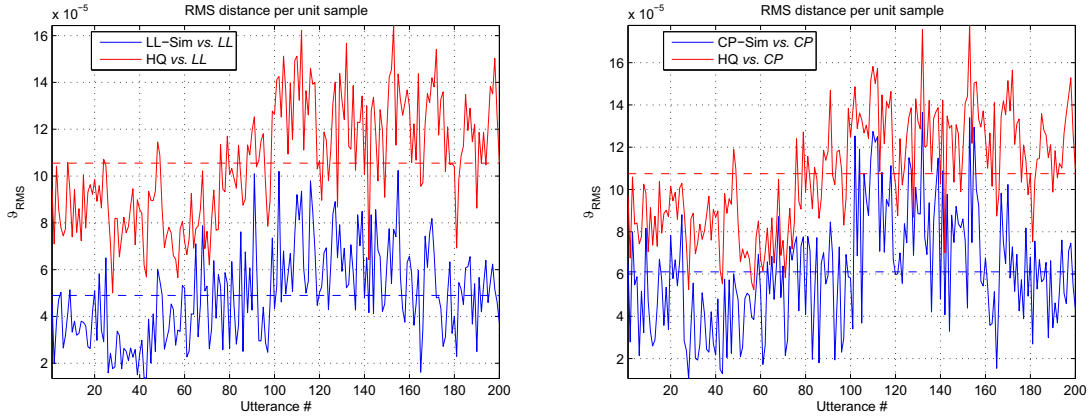
$$\vartheta_{\mathbf{KL}} = \frac{1}{N}\sum D\left(\bar{D}_i, \bar{Y}_i\right)$$

3. Symmetrized Kullback-Leibler distance

$$\begin{aligned} D_S &= \frac{1}{2}\left(D(p,q) + D(q,p)\right) \\ &= \frac{1}{2}\sum p_i \log_2 \frac{p_i}{q_i} + q_i \log_2 \frac{q_i}{p_i} \end{aligned}$$

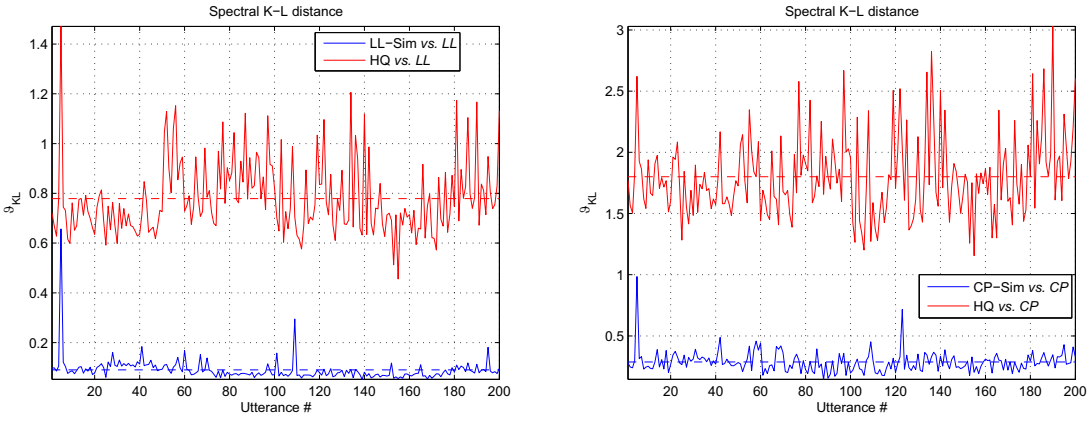
4. Jensen-Shannon divergence

$$\begin{aligned} D_{JS} &= \frac{1}{2}\left(D(p,m) + D(q,m)\right) \\ m &= \frac{p+q}{2} \end{aligned}$$



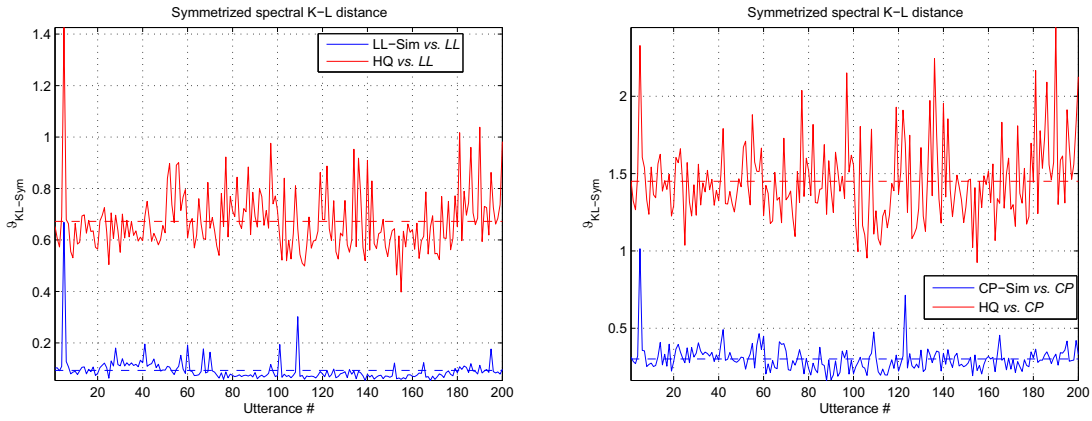
(a) Landline (b) Cellphone

Figure 6.27: RMS distance



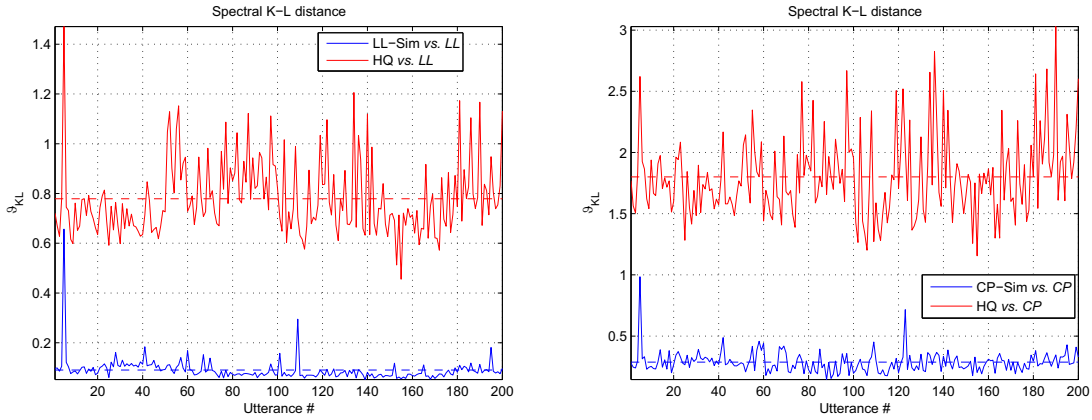
(a) Landline (b) Cellphone

Figure 6.28: Kullback-Leibler distance



(a) Landline (b) Cellphone

Figure 6.29: Symmetrized Kullback-Leibler distance



(a) Landline (b) Cellphone

Figure 6.30: Jensen-Shannon divergence

The first measure deals with the differences in the time domain, namely the time-domain error energy. the subsequent measures deal with the entropy[4] or "randomness" of the signal spectra, more specifically they are different ways to measure joint randomness e.g. between the simulation and the actual channel.

[4]This can also be interpreted as the minimum number of bits we would have to use to represent (i.e. encode) a random variable with a pdf equal to the normalized signal spectrum.

Table 6.3: Average distances among speakers

Distances	RMS[5]	KLD	KLD-Sym	JSD
Sim-LL vs. LL	3.97×10^{-5}	0.20	0.21	0.12
HQ vs. LL	1.09×10^{-4}	0.95	0.82	0.36
Sim-CP vs. CP	9.65×10^{-5}	0.59	0.62	0.51
HQ vs. CP	2.15×10^{-4}	2.76	2.31	2.18

Table 6.4: Average distances' gain ratios among speakers

Gain ratio	RMS[5]	KLD	KLD-Sym	JSD
Sim-LL vs. LL	2.72	4.63	3.90	3.11
Sim-CP vs. CP	2.23	4.65	3.71	4.30

From the graphs we see that both the time-domain distance (i.e. per unit sample RMS prediction error energy) as well as the information theoretical measures (entropy related metrics) are extremely low for the simulation vs. landline and high for the high quality recording vs. the landline. This goes to show the that from an objective point of view, the simulation is a faithful representation of the actual landline channel.

The figures show the values for the 200 utterances of a single speaker to expose the trend that appears throughout speakers. The average distances among six speakers are expressed in table 6.3; several observations are in order. First, we see that the trend presented in the figures for one speaker is preserved among multiple speakers. Secondly, the distances between the simulation and the channel is much lower than between the clean recording and the channel itself. While differences between the landline and cellphone distances are significant, we must examine the gain ratio of the distance HQ-to-channel over the distance simulation-to-channel, which are shown in table 6.4. This ratio now shows that the distances are extremely consistent, i.e. within each metric, the values are very similar for both the landline and cellphone simulations. Because of all of this, both our estimates can be said to perform equally well from an objective point of view.

6.4 Voice Over IP Simulation

In VoIP systems, the situation is such that the transfer function from input to output does not characterize the system. In fact, since the system is nonlinear and time variant, it

[5]per unit sample.

doesn't really makes sense to speak of a transfer function. For that purpose, a stochastic model is adopted. In VoIP communications, packet losses and out-of-order arrivals are the primary channel characteristics, so our model consists of three parameters: the network delay, the network delay jitter (variability of the delay) and the receiver buffer length. These three parameters jointly determine most of the behavior of a VoIP communication. Additional parameters of lesser influence include the actual codec being used, the voice activity thresholding mechanism, etc.

A model of a VoIP system is shown in figure 6.31. For our purposes, we separate the system into independent components: the signal first goes through a LPF to filter out any information which is not strictly necessary for intelligibility purposes. Then, the audio stream is split into data packets and encoded using some standard (PCM, ADPCM, etc). Voice activity detection is performed on each packet so that packets which carry little information aren't even transmitted over the network. If the frame contains data that is significant, then the packet will be transmitted over the network. By going through a physical network topology, a delay will occur; if this delay were the same for every packet, then at the other end we would just end up with a delayed version of the input. However, data networks shuffle packets so that they may arrive with different delays. A measure of the delay variance is called the jitter. The network jitter hinders the ability to successfully reproduce audio on the other end in real time, since the receiver has a finite buffer. In addition to a variable delay, the network may introduce errors in the packet (at a bit level). The packet may include some kind of redundancy to either palliate or fix these errors when possible (such as Hamming codes). The final step entails decoding the packet and constructing a waveform out of the stream.

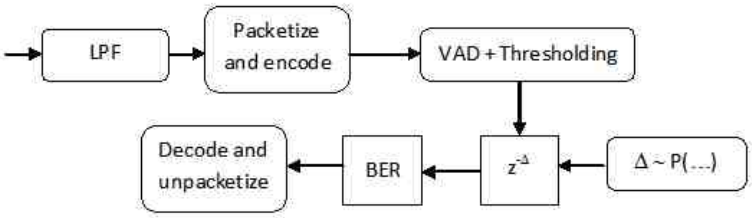


Figure 6.31: A VoIP model

The main difference between the approaches for the landline and cellphone simulations versus the VoIP simulation is that in the former we estimate our model from existing data, i.e. we fit the model parameters so that the model mimics the actual data as close as possible

Algorithm 11 Variable delay simulation: stream building

Input: an array of data packets. J, the network jitter.

Output: an array of lists of packets. Each entry in the array is a time slot and represents the time at which the packets in the corresponding list arrive at the receiving end.

For each packet number p,

1. Calculate the delay that packet is going to go through $\Delta_p = |x| J$ where $x \sim \mathcal{N}(0, 1)$ and J is the network jitter
2. Put the packet p into the list at array entry number $p + \Delta_p$.

(within the allowable models in the model class), while in the latter we make certain "reasonable" assumptions as to the model parameters. These in turn may be measured from the actual data (which at the time of this writing is unavailable), and modify the simulation's parameters accordingly.

6.4.1 Variable Delay

In our simulation the delay for each packet is drawn from a right-tailed normal distribution with variance J (the network jitter). As we already stated, the receiver buffer which is of finite length, will wait to be full before starting to output data. That way the system can cope with packets arriving in disorder at the beginning. Furthermore, if we get a packet that is supposed to be played in the future, we can store it and hold off until it is time to play it. If at any given time we don't have any data in the buffer to play, we must output some kind of silence, noise being the more desirable option because of its perceptual "comfort" advantage over digital silence. Finally, if we get a packet that we should have gotten before, we will just discard it since the communication must be conducted in real time.

For our simulation we split this network delaying into two separate processes: the first one builds up a stream of packets arriving at each time instant, while the second one decides whether there is data in the buffer to output or noise. The pseudocode is illustrated in algorithms 11 and 12.

The variable delay algorithm was run on a number of audio files producing credible results to the ear. One such sample along the error introduced by the algorithm is shown in figure 6.32.

Algorithm 12 Variable delay simulation: receiver throttling

Input: an array of list of packets such as the output from algorithm 11. `bufsize`, the buffer size.

Output: an array of data packets such as the input from algorithm 11. The number of dropped packets.

Initialization: While the buffer is not full, insert data into the buffer. Set `next_packet = 1`.

Once the buffer is full,

1. Insert all the elements at the current time from the input structure into the buffer
2. `min_packet` = the packet with the minimum number in the buffer.
3. If we have already started outputting data
   (a) If `min_packet = next_packet`
      i. Remove and output `min_packet` from the buffer
   (b) Else, if `min_packet != next_packet`
      i. Calculate $\sigma_n^2 = \frac{1}{10}\sigma^2_{\text{packet}}$ (i.e. 20 dB below the packet level)
      ii. Output noise with that variance
   (c) In any case, `next_packet = next_packet + 1`

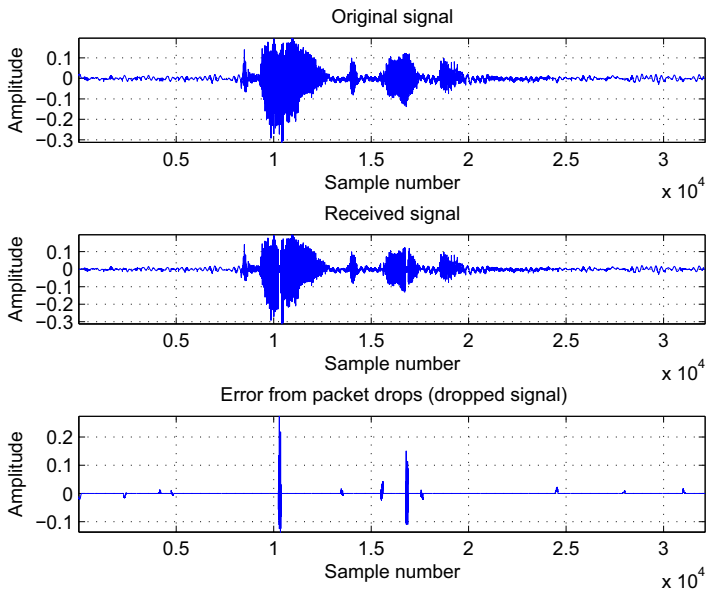


Figure 6.32: Variable delay simulation: signal and introduced error by network jitter, drop rate is 4%.

6.4.2 Packet Error Simulation

The simplest approach to packet losses would be to assume that packet losses are independent of each other and have a probability of P_e. This is known as the Bernoulli model. In a data stream of N packets, the expected number of lost packets will be NP_e. This model despite being simple is not realistic since it fails to capture burst losses.

The Gilbert-Elliot model is a discrete-state discrete-time Markov chain as shown in figure 6.33. This model consists of two states "OK", where the probability of losing a packet is $e_{\text{OK}} \ll 1$ and "LOSS" where it is high ($e_{\text{LOSS}} \rightarrow 1^-$). At the same time, the system dynamics are governed by transition from the OK state to the LOSS state and viceversa. Within a given state, errors are assumed to be independent from each other. Therefore, to fully specify the G-E model we need to give the 2×2 probability transition matrix as well as the packet error rates in each state (so four numbers in total). The mean soujourn time (average time in a state) measured in time slots (duration of a packet) is given by

$$\begin{aligned} T_{\text{OK}} &= \frac{1}{1-p_{\text{OK}}} \\ T_{\text{LOSS}} &= \frac{1}{1-p_{\text{LOSS}}} \end{aligned}$$

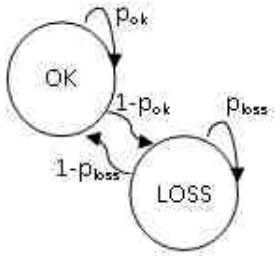


Figure 6.33: The Gilbert-Elliot model

Some standard values are:

- $e_{\text{OK}} = 10^{-5}$, $e_{\text{LOSS}} = 10^{-2}$, $p_{\text{OK}} = 0.9999918$ and $p_{\text{LOSS}} = 0.999184$ ([84])
- $e_{\text{LOSS}} = 0.8$, $e_{\text{OK}} = 0$, $T_{\text{OK}} = 1 \ldots 10s$, $T_{\text{LOSS}} = 50 \ldots 500ms$ @10 Mbps (Ethernet), 512 byte packet size ([85])

$$\begin{aligned} p_{\text{OK}} &= 1 - \frac{1}{T_{\text{OK}}} \\ &= 1 - \frac{1}{5s \times 40} \\ &= 0.9950 \\ p_{\text{LOSS}} &= 1 - \frac{1}{250ms \times 40} \\ &= 0.9 \end{aligned}$$

The packet size makes sense: we have packets every 25 ms (40 packets per second) at 16 bits per sample at 4 kHz BW, so $\frac{1}{8\text{kHz}} = 1$ sample every 0.125ms which is equivalent to $\frac{25\frac{\text{ms}}{\text{packet}}}{0.125\frac{\text{s}}{\text{sample}}} = 200\frac{\text{sample}}{\text{packet}}$. Since each sample is 2 bytes long, we have 400 data bytes per packet. By adding a packet header we introduce some overhead, arriving at the figure of 512 bytes per packet.

The same signal as in 6.32 was processed with the G-E model; the signal as well as the introduced error are shown in figure 6.34.

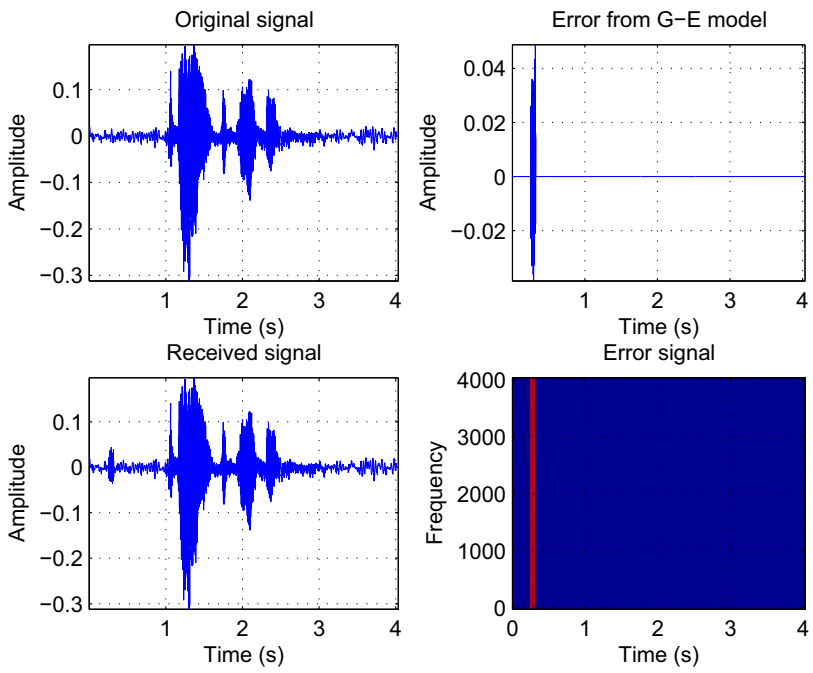


Figure 6.34: Packet error simulation: signal and error introduced by G-E model. Packet error rate is 1.6% (4×10^{-4} BER)

To learn the system parameters (transition and emission probabilities) from a set of training data, one can use the Baum-Welch algorithm ([10]). A treatment of training Markov model parameters for channels and especially the Gilbert-Elliot model is given in [123, 124, 125, 126, 127, 128].

6.4.3 Packet Error Recovery and Concealment

Forward error correction (FEC) is a system of error control for data transmission, where the sender adds redundant data to its messages, which allows the receiver to detect and correct errors (within some bound) without the need to ask the sender for additional data. Often, by using FEC, data retransmission is unnecessary. This nevertheless comes at the cost of higher bandwidth requirements on average, and is therefore applied in situations where retransmissions are relatively costly or impossible.

Packet loss concealment is a family of techniques to mask the effect of packet loss in VoIP communications. There are three types of PLC techniques:

Zero insertion: the lost speech frames are replaced with digital silence (zeros)

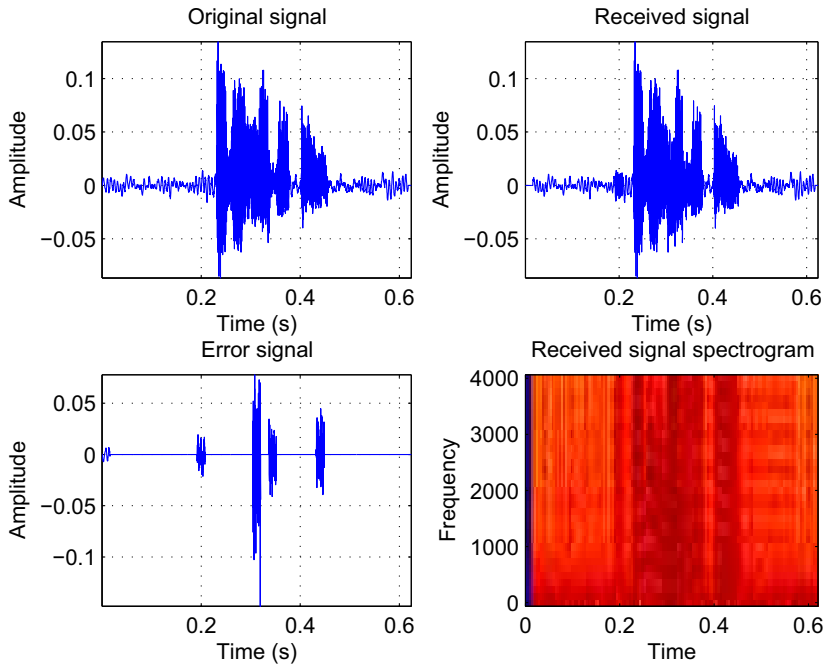


Figure 6.35: Packet concealment using 0 dB SNR white noise to conceal lost packets. Top: original and received signal; bottom: packet concealment-induced noise injected into received signal, spectrogram of the received signal.

Waveform substitution: the missing gap is reconstructed by repeating a portion of already received speech. The simplest form of this would be to repeat the last received frame. Other techniques account for fundamental frequency, gap duration etc. Waveform substitution methods are popular because of their simplicity to understand and implement. An example of such algorithm is proposed in ITU recommendation G.711 Appendix I ([11]).

Model-based methods: an increasing number of algorithms that take advantage of speech models of interpolating and extrapolating speech gaps are being introduced and developed.

An excellent review on packet error recovery and concealment for audio streams is found in [90].

Figure 6.35 shows the effects of packet concealment. Note the smooth strips at the times of noise injection in the spectrogram, indicating the presence of white noise.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

We have reviewed the state of the art in speech recognition tools and have gathered enough knowledge to reduce the problem of ASR retraining to the simulation of channel data. We have described all the design choices that are to be made when designing the simulation (simulation structure, features, norms, estimation algorithms, implementation). Moreover, we have introduced a result which states that minimization of the cepstral energy is equivalent to minimization of the frequency response deviation, and introduced a novel method to iteratively pursue a general feature search as well as revisited methods to perform minimization by using techniques related to the cyclic-coordinate search. Among our results we have included the successful estimation of landline and cellphone channels by using a simple and short RLS FIR filter and the feature-optimal LFEN filter, and then compared their performance. Performance measures for the RLS simulations, both subjective and objective, were exposed in the previous chapter.

We also have arrived at the conclusion that, given

1. today's average PC's computational power,
2. the excessive computational requirements of the LFEN filter relative to the RLS filter, and
3. the comparable performance for RLS and LFEN filters as to RMS and MFCC error norm,

we can use RLS filter estimates without incurring in too great an error; alternatively we might also benefit on pursuing LFEN estimates by using the RLS estimate as a starting

point for the algorithm iterations. In addition, we have delineated the difficulties arising from trying to model a VoIP channel and proposed and implemented a modular architecture for VoIP simulation.

7.2 Future Work

Our results would greatly benefit from an algorithm efficient enough to be comparable in speed of convergence and computational requirements to the RLS algorithm, while at the same time being able to accommodate different feature sets, i.e. have the algorithm be parametric on the features it tries to minimize.

In addition to a more efficient estimation algorithm, we would like to incorporate time-varying behavior to our channel model. As of now, despite our model being an adaptive filter (and therefore adapting to **slow** signal variations), once the estimation is complete we are left with an impulse response $h[n]$ which is fixed. To be able to model time-variance, we must incorporate an impulse response of the form $h[n, m]$, that is, a (slowly) varying impulse response for different time instants. The reason for such a requirement is the ability to capture such phenomena as network jitter and packet reordering/loss, which is ubiquitous in data networks which do not guarantee ordered delivery (such as TCP) but rather are best-effort transport layers (RTP, VoIP, and cellular networks in general), and mobile cellular phone channels which include handover from one base station to another, and since the speaker is moving, the channel will exhibit different attenuations at different times for a given frequency. All of these phenomena are thus time-variant and in the case of mobile cellular phones, even worse, since they depend on the speakers trajectory, and the cellular network state at the time (such as availability), as well as the precise cellular technology, such as the channel access technology (CDMA, TDMA, FDMA) and moreover the exact cellular standard (GSM, GPRS and such).

Finally, we have made the conscientious decision to use a linear model, for otherwise it would have made the algorithms incredibly difficult to mathematically tract. Going forward, however, we would like to incorporate nonlinear effects in our models. Such effects are present in every model, we usually just choose to regard them as small enough and overlook them. Consider a model $\Lambda(\theta)$ depending on the parameter vector θ. Then under reasonable regularity conditions we may write

$$\Lambda(\theta) = \Lambda(\theta_0) + (\theta - \theta_0)^T \nabla\Lambda(\theta_0) + \frac{1}{2}(\theta - \theta_0)^T \mathbb{H}\Lambda(\theta_0)(\theta - \theta_0) + \mathcal{O}\left(|\theta - \theta_0|^3\right)$$

Therefore, when we use a linear model we are saying that

$$\Lambda(\theta) = \Lambda(\theta_0) + (\theta - \theta_0)^T \nabla\Lambda(\theta_0) + \nu_{\theta_0}(\theta - \theta_0)$$

or in some cases we might express the model in a basis such that

$$\Lambda(\theta) = \nabla\Lambda(0) \cdot \theta + \nu(\theta)$$

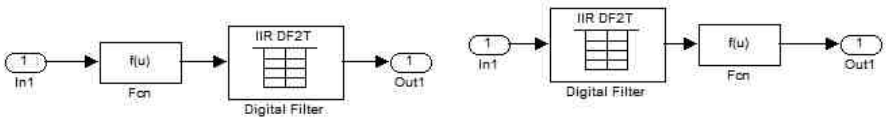
In summation, a linear model is one in which the dependence on the model parameters is of direct proportionality. In addition, we have an extra term $\nu(\theta)$; this term is usually considered the "noise" in the model, since noise is considered anything which is undesirable, or anything that doesn't fit our model.

In any case, nonlinearities in our case arise in the form of several effects:

- Large signal non-linearity; i.e. soft saturation or clipping, which introduces noticeable nonharmonic distortion.
- Small signal non-linearity; we are assuming that the model residuals (terms that depend on the square and higher powers of the parameters) are negligible, even for small parameter values, which isn't always the case.
- Time-dependent non-linearity; i.e. hysteresis in physical materials (such as those arising from ferromagnetic materials) and material degradation over time.

These nonlinearities, even without incorporating time-variance, make the model very complex and almost impossible to estimate. In addition, decomposition of a filter into a non-linearity and a LTI filter can be extremely difficult, since even the simplest polynomial non-linearity is usually not commutative with respect to a convolution operation: consider the filter $h\{x[n]\} = \frac{1}{2}(x[n] + x[n-1])$ and the non-linearity $p(x) = ax + bx^2$. If the signal goes through the non-linearity before going into the LTI filter (see figure 7.1), we get an output

$$y_1[n] = \frac{1}{2}\left(ax[n] + bx[n]^2 + ax[n-1] + bx[n-1]^2\right)$$



(a) Nonlinearity, then LTI (b) LTI, then nonlinearity

Figure 7.1: Non-commutativity of non-linearity and LTI filter

On the other hand, if we reverse the components the output will be

$$\begin{aligned} y_2[n] &= a \cdot \frac{1}{2}(x[n] + x[n-1]) + b\left(\frac{1}{2}(x[n] + x[n-1])\right)^2 \\ &= \frac{1}{2}(ax[n] + ax[n-1]) + \frac{b}{4}\left(x[n]^2 + x[n-1]^2 + 2x[n]\,x[n-1]\right) \\ &= \frac{1}{2}\left(ax[n] + \frac{b}{2}x[n]^2 + ax[n-1] + \frac{b}{2}x[n-1]^2 + bx[n]\,x[n-1]\right) \end{aligned}$$

so that apart from the differences in the coefficients which might be adapted from one model to the other, in y_2 we have cross products, which turn the *ansatz* completely different.

Even in the case of estimating a nonlinear model without any memory (i.e. a filter in which the output depends only on the current input sample), the parameter estimation has its own subtleties. Let us consider the following simple example, of estimating the following model

$$y[n] = a \cdot x[n] + b \cdot x[n]^2 + \nu[n]$$

So the MMSE estimate is derived as follows:

$$\begin{aligned} \nu[n] &= y[n] - ax[n] - bx[n]^2 \\ \nu[n]^2 &= y[n]^2 + a^2x[n]^2 + b^2x[n]^4 \\ &\quad -2ax[n]\,y[n] - 2bx[n]^2\,y[n] + 2abx[n]^3 \end{aligned}$$

To minimize $\sum \nu[n]^2$ we set the derivatives with respect to a, b and c to zero:

$$\begin{aligned} \frac{\partial \nu[n]^2}{\partial a} &= 2ax[n]^2 - 2x[n]\,y[n] + 2bx[n]^3 \\ \frac{\partial \nu[n]^2}{\partial b} &= 2bx[n]^4 - 2x[n]^2\,y[n] + 2ax[n]^3 \end{aligned}$$

so

$$\begin{cases} \sum 2ax\left[n\right]^2 - 2x\left[n\right]y\left[n\right] + 2bx\left[n\right]^3 = 0 \\ \sum 2bx\left[n\right]^4 - 2x\left[n\right]^2 y\left[n\right] + 2ax\left[n\right]^3 = 0 \end{cases}$$

$$\begin{aligned} \sum ax\left[n\right]^2 - x\left[n\right]y\left[n\right] + bx\left[n\right]^3 &= 0 \\ \sum bx\left[n\right]^4 - x\left[n\right]^2 y\left[n\right] + ax\left[n\right]^3 &= 0 \end{aligned}$$

so that we have a 2×2 system of equations which can be solved by well-known methods. However, depending on the data points and the actual model equation, the system of equations might be unsolvable (i.e. the associated matrix is not full rank and hence it is not invertible).

In summation, a comprehensive time-variant and nonlinear model is extremely difficult to estimate, but will be necessary in order to accurately predict and simulate such effects which escape the LTI filter model.

Index

Bibliography

[1] S. Deligne et al., "A Robust High Accuracy Speech Recognition System for Mobile Applications". IEEE Trans. Speech and Audio Processing, 2002.

[2] L. Ljung, "System Identification: Theory for the user", Prentice-Hall, Englewood Cliffs, NJ, 1987.

[3] S.K. Mitra, "Digital Signal Processing", 2^{nd} Edition, McGraw-Hill Science, McGraw-Hill, 2001.

[4] S.J. Orphanidis, "Optimum Signal Processing", 2^{nd} Edition, McGraw-Hill, 2007.

[5] S. Haykin, "Adaptive Filter Theory", 4^{th} Edition, Pearson Education, 2007.

[6] A. Oppenheim and R. Schafer, "Discrete-time Signal Processing", Prentice-Hall Signal Processing Series. Prentice-Hall, Upper Saddle River, NJ, 1999.

[7] A.H. Sayed, "Fundamentals of Adaptive Filtering", Wiley-IEEE Press, 2003.

[8] T.F. Quatieri, "Discrete-Time Speech Signal Processing: Principles and Practice", Prentice-Hall, 2001.

[9] L. Rabiner and R. Schafer, "Digital Processing of Speech Signals", Prentice-Hall, 1978.

[10] Wikipedia `http://www.wikipedia.com`

[11] International Telecommunications Union Recommendation G.711, 1988. Available at `http://www.itu.int/rec/T-REC-G.711/e`.

[12] J. Shlens, "A Tutorial on Principal Component Analysis", Salk Institute for Biological Studies and University of California, San Diego, 2005.

[13] R. Mutihac and M. Van Hulle, "Comparison of Principal Component Analysis and Independent Component Analysis for Blind Source Separation". Romanian Reports in Physics, Publishing House of the Romanian Academy, vol. 56, No. I, pp. 20-32, 2004.

[14] P. Somervuo, “Feature Transformations and Combinations for Improving ASR Performance”, Eurospeech, 2003.

[15] G. Strang, “The Discrete Cosine Transform”, SIAM Review, vol. 41, No. 1, pp. 135-147, 1999.

[16] K. Wright, “Notes on Ogg Vorbis and the MDCT”, `http://www.free-comp-shop.com/vorbis.pdf`, 2003.

[17] J.P. Princen and A.B. Bradley, "Analysis/synthesis filter bank design based on time domain aliasing cancellation" IEEE ASSP, vol. 34, issue 5, pp. 1153-1161, 1986.

[18] P.P. Vaidyanathan, “Multirate Systems and Filterbanks”, Prentice Hall, 1993.

[19] I. Daubechies, “Ten Lectures on Wavelets”, SIAM, 1992.

[20] C. Valens, “A Really Friendly Guide to Wavelets”, `http://perso.orange.ft/polyvalens/clemens/wavelets`, 1999-2004.

[21] B.T. Tan et al., “The Use of Wavelet Transforms in Phoneme Recognition”, Proc. ICLSP, vol. 4, pp. 2431-2434, 1996.

[22] G. Welch and G. Bishpo, “An Introduction to the Kalman Filter”, SIGGRAPH, course 8, 2001. Available at `http://www.cs.unc.edu/~tracker/ref/s2001/kalman`

[23] J.J. Shynk, ”Adaptive IIR Filtering”, IEEE ASSP Magazine, vol. 6, no. 2, pp. 4-21, 1989.

[24] M. Harteneck, J.G. McWirther, “An Effective Approach to Adaptive IIR Filtering”, Proc. ICASSP, 1996.

[25] S.T. Neely, J.B. Allen, “Invertibility of a room impulse response”, Journal of the Acoustical Society of America, vol. 66, 1979.

[26] G. Dumont, “Adaptive Control: Recursive Identification Algorithms”, University of British Columbia, 2007.

[27] Y. Zhou et al., “A New LMS/Newton Algorithm for Robust Adaptive Filtering in Impulsive Noise”, EUSIPCO, 2004.

[28] M. Rabinovitz et al., “An Adaptive Newton-like Training Algorithm For Nonlinear Filters which have Embedded Memory”, IEEE-EURASIP NSIP99, 1999.

[29] A.M. Reza, Spire Lab, UWM, “From Fourier Transform to Wavelet Transform”, white paper, 1999.

[30] E. Zwicker, “Subdivision of the audible frequency range into critical bands", Journal of the Acoustical Society of America, vol. 33, 1961.

[31] B.J. Shannon and K.K. Paliwal, "A Comparative Study of Filter Bank Spacing for Speech Recognition", Microelectronic Engineering Research Conference, 2003.

[32] G. Bojana and K.K. Paliwal, "Robust Parameters for Speech Recognition Based on Subband Spectral Centroid Histograms", Microelectronic Engineering Research Conference, 2003.

[33] V. Tyagi and C. Wellekens, "On desensitizing the Mel-Cepstrum to spurious spectral components for Robust Speech Recognition", Proc. ICASSP, vol. 1, pp. 529-532, 2005.

[34] `http://www.voicefoundation.org`, 2007.

[35] A. Sklar, University of Miami, "ARMA Parameter Estimation", unpublished, 2007.

[36] C. Chen et al., "Speech Feature Smoothing for Robust ASR", Proc. ICASSP 05, vol. 1, pp. 525-528, 2005.

[37] B.T. Lilly, Griffith University, "Robust Speech Recognition in Adverse Environments", PhD Thesis, 2000.

[38] C. Benitez et al., "Robust ASR front-end using spectral-based and discriminant features: experiments on the Aurora tasks", Eurospeech 2001, Aalborg, 2001.

[39] Y.N. Rao, University of Florida, "An Augmented Error Criterion for Linear Adaptive Filtering: Theory, Algorithms and Applications", PhD Thesis, 2004.

[40] J. Lindqvist-Gauffin and J. Sundberg, "Acoustic properties of the nasal tract", Quarterly Progress and Status Report, Dept. for Speech Music and Hearing, KTH University, Sweden, vol. 13, pp. 13-17, 1972.

[41] D. Yuk, Rutgers University, "Robust Speech Recognition Using Neural Networks and Hidden Markov Models - Adaptations using Non-linear Transformations", PhD Thesis, 1999.

[42] S. Salvador and P. Chan, "FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space", KDD Workshop on Mining Temporal and Sequential Data, pp. 70-80, 2004.

[43] H. Nakasone, "Automatic Speaker Recognition In Forensic Environment", IOCE Conference, 2002.

[44] M. Benzeghiba et al., "Impact of variabilities on speech recognition", Proc. SPECOM, 2006.

[45] Z. Bin et al., "An Enhanced RASTA processing for speaker identification", Proc. ISCSLP 2006.

[46] H. Tolba and D. O'Shaughnessy, "Combating Nonlinear Telephone Channel-Noise Using The Multiband AM-FM Model", NSIP 1999, pp. 168-171.

[47] C. Jia and B. Xu, “An Improved Entropy-Based Endpoint Detection Algorithm”, ISCLSP 2002, p. 96.

[48] T. Kemp et al., “Strategies For Automatic Segmentation Of Audio Data”, Proc. ICASSP, vol. 3, pp. 1423-1426, 2000.

[49] S. Doh and R.M. Stern, “Inter-class MLLR For Speaker Adaptation”, Proc. ICASSP, vol. , pp. 1755-1758, 2000.

[50] S. Imai, “Cepstral Analysis Synthesis On The Mel Frequency Scale”, Proc. ICASSP, vol. 8, pp. 93-96, 1983.

[51] C. Tarcisio et al., “Use Of Simulated Data For Robust Telephone Speech Recognition”, Proc. EUROSPEECH, vol. 6, pp. 2825-2828, 1999.

[52] H.P. Combrinck and E.C. Botha, “On The Mel-scaled Cepstrum”, Proc. 7th Annual South African Workshop on Pattern Recognition, 1996.

[53] O. Kwon et al., “Emotion Recognition by Speech Signals”, Proc. 8th EUROSPEECH, pp. 125-128, 2003.

[54] T. Fukada et al., “An Adaptive Algorithm For Mel-Cepstral Analysis Of Speech”, ICASSP-92, vol. 1, pp. 137-140, 1992.

[55] D. Giuliani et al., “Training of HMM with Filtered Speech Material for Hands-Free Recognition”, ICASSP, vol. 1, pp. 449-452, 1999

[56] `http://www.kernel-machines.org`, 2007.

[57] H. Hermansky and N. Malayath, “Spectral Basis Functions from Discriminant Analysis", ICSLP, 1998.

[58] A. Stolcke et al., “MLLR Transforms as Features in Speaker Recognition”, Interspeech, 2005.

[59] M.J. Hunt, “Spectral Signal Processing for ASR”, IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU-99, 1999.

[60] S. Salvador and P. Chan, “FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space”, Proc. KDD Workshop on Mining Temporal and Sequential Data, 2004

[61] V. Stahl et al., “Acoustic Synthesis Of Training Data For Speech Recognition In Living Room Environments”, ICASSP, 2001.

[62] J.L. Shen et al., “Robust Entropy-based Endpoint Detection for Speech Recognition in Noisy Environments”, ICSLP, 1998.

[63] B. Fuglede and F. Topsøe, ”Jensen-Shannon Divergence and Hilbert space embedding”, Proc. ISIT, pp. 31-, 2004.

[64] C.C. Rodríguez, "The Kernel Trick", `http://omega.albany.edu:8008`, 2004 (last visited November 2007).

[65] R.C. Rose and S. Parthasarathy, "Tutorial on ASR for Wireless Mobile Devices", ICSLP, 2002.

[66] R. Hsiao, Hong Kong University of Science and Technology, "Kernel Eigenspace-Based MLLR Adaptation", M.Phil. Thesis, 2004.

[67] B. Schölkopf et al., "Kernel Principal Component Analysis", Advances in kernel methods: support vector learning, pp. 327-352, 1999.

[68] B. Schölkopf et al., "Nonlinear Component Analysis as a Kernel Eigenvalue Problem", Neural Computation, vol. 10, pp. 1299-1319, 1998.

[69] M. Matassoni et al., "Hands-Free Speech Recognition Using A Filtered Clean Corpus And Incremental HMM Adaptation", Proc. ICASSP, 2000.

[70] M. Matassoni et al., "Some Results On The Development Of A Hands-Free Speech Recognizer For Car-Environment", Proc. ASRU Workshop, 1999.

[71] D. Giuliani et al., "Robust Hmm Training And Adaptation In Hands-Free Speech Recognition", Proc. ASRU Workshop, 1999.

[72] P. Renevey, Swiss Federal Institute Of Technology (EFPL), "Speech Recognition In Noisy Conditions Using Missing Feature Approach", PhD Thesis, 2000.

[73] K. Tokuda et al., "Speech Coding Based On Adaptive Mel-Cepstral Analysis", ICASSP 94, vol. 1, pp. 197-200, 1994.

[74] C. Chelba and T.J. Hazen, "Automatic Spoken Document Processing for Retrieval and Browsing", ICASSP 2007 Tutorial, 2007.

[75] I. Viikki et al., "Speaker- And Language-Independent Speech Recognition In Mobile Communication Systems", ICASSP 2001, 2001.

[76] Q. Zhu et al., "Noise Robust Feature Extraction for ASR using the Aurora-2 Database", EUROSPEECH, 2001.

[77] X.Y. Gao, University of Toronto, "Adaptive Linear and Nonlinear Filters", PhD Thesis, 1991.

[78] J. Li, Japan Advanced Institute of Science and Technology, "Noise Reduction Based on Microphone Array and Post-filtering for Robust Hands-free Speech Recognition in Adverse Environments", PhD Thesis, 2006.

[79] M.H. Johnsen et al., "Experiments on Cepstral Mean Subtraction (CMS) and RASTA filtering Applied to SAMPA Phoneme Recognition", COST 249, 1995.

[80] L. Neumeyer et al., “Training Issues and Channel Equalization Techniques for the Construction of Telephone Acoustic Models Using a High-Quality Speech Corpus”, IEEE Transactions on Speech and Audio Processing, vol. 2, issue 4, pp. 590-597, Oct 1994.

[81] N. Kumar, Johns Hopkins University, “Investigation of Silicon Auditory Models and Generalization of Linear Discriminant Analysis for Improved Speech Recognition”, PhD Thesis, 1997.

[82] S. Kang, “Dissonant frequency filtering technique for improving perceptual quality of noisy speech and husky voice”, ACM Signal Processing, vol. 84, No. 2, pp. 431-433, 2004.

[83] –, “Spoken Database Collection and Channel Modeling for Automatic Speech Recognition Retraining”, Project Proposal, A joint IBM-UM Project, 2006.

[84] H.S. Wang and N. Moayeri, “Finite State Markov Channel - a Useful Model for Radio Communication Channels”, IEEE Trans. on Vehicular Technology, vol. 44(1), pp. 163-171, 1995.

[85] P. Bhagwat et al., “Using Channel State Dependent Packet Scheduling to Improve TCP Throughput Over Wireless LANs”, Wireless Networks, vol. 3(1), pp. 91-102, 1997.

[86] J.P. Ebert and A. Willig, “A Gilbert-Elliot Bit Error Model and the Efficient Use in Packet Level Simulation”, TKN Technical Reports Series of Technical University Berlin, 1999.

[87] J.H. Mock, University of New Hampshire, “A Voice Over IP Solution to the Problem of Mobile Radio Interoperability”, MSc Thesis, 2003.

[88] A.C. Costa et al., “Channel-Aware Schedulers For VoIP and MPEG4 for Wireless Communications”, Proc. MOMUC, 2003.

[89] C.C. Hess, University of Illinois at Urbana-Champaign, “Media Streaming Protocol: An Adaptive Protocol for the Delivery of Audio and Video Over the Internet”, MSc Thesis, 1998.

[90] C. Perkins et al., “A survey of packet loss recovery techniques for streaming audio”, IEEE Network, issue 5, vol. 12, pp. 40-48, 1998.

[91] J. Chen et al., “New Insights Into the Noise Reduction Wiener Filter”, IEEE Transactions on Audio, Speech and Language Processing, vol. 14, No. 4, July 2006.

[92] P.J. Moreno, Carnegie Mellon University, “Speech Recognition in Noisy Environments”, PhD Thesis, 1996.

[93] C. Mokbel, “Comparison of Several Preprocessing Techniques For Robust Speech Recognition over both PSN and GSM Networks”, EURASIP-Eusipco, 1996.

[94] D. Gelbart, "Reducing the Effect of Room Acoustics on Human-Computer Interaction", AVIOS Conference, 2002.

[95] D. Gelbart and N. Morgan, "Evaluating Long-term Spectral Subtraction for Reverberant ASR", Proc. ASRU Workshop, pp. 103-160, 2001.

[96] T. Eisele et al., "A Comparative Study of Linear Feature Transformation Techniques for Automatic Speech Recognition", Proc. ICSLP '96, pp. 252-255, 1996.

[97] S.B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", IEEE Trans. ASSP, vol. 28, pp. 357-366, 1980.

[98] A. Acero, "Combating Background Noise: Past, Present and Future", Microsoft Research, 2005.

[99] C.P. Chen et al., "Speech Feature Smoothing For Robust ASR", Proc. ICASSP '05, vol. 1, pp. 525-528, 2005.

[100] T.J. Hazen and V.W. Zue, "Recent Improvements in an Approach to Segment-based Automatic Language Identification", Proc. ICLSP '94, pp. 1883-1886, 1994.

[101] Y. Selén, Uppsala University, "Model Selection", PhD Thesis, 2004.

[102] G. Saon and M. Padmanabhan, "Minimum Bayes Error Feature Selection for Continuous Speech Recognition", NIPS, pp. 800-806, 2000.

[103] U.H. Yapanel and J.H.L. Hansen, "A New Perspective on Feature Extraction for Robust In-Vehicle Speech Recognition", Eurospeech 2003, pp. 1281-1284, 2003.

[104] Y. Gao et al., "Recent Advances in Speech Recognition System for IBM DARPA Communicator", Eurospeech 2001, pp. 503-506, 2001.

[105] D. Matrouf, "Model Compensation for Noises in Training and Test Data", Proc. ICASSP, pp. 831-834, 1997.

[106] W. Macherey, "Investigations on Error Minimizing Training Criteria for Discriminative Training in Automatic Speech Recognition", Interspeech, pp. 2133-2136, 2005.

[107] R. O. Duda et al., "Pattern Classification", 2^{nd} Edition, John Wiley & Sons, 2000.

[108] B. Milner, "Distributed Speech Recognition Research at the University of East Anglia", Keynote Presentation at COST-278, 2002.

[109] M. Perakakis, Technical University of Crete, "Distributed Speech Recognition for wired/wireless data networks (Internet/GPRS) using 2 Kbps coding", Diploma Thesis, 2000

[110] E.G. Learned-Miller and J. W. Fisher III, "ICA Using Spacings Estimates of Entropy", Journal of Machine Learning Research 4, pp. 1271-1295, 2003

[111] S. Weiss et al., “Adaptive Equalization in Oversampled Subbands”, IEE Electronic Letters, vol. 34, No. 15, pp. 1452-1453, 1998.

[112] Y. Gong et al., “An Adaptive Linear Equalizer With Optimum Filter Length and Decision Delay”, IMA 2006.

[113] M. Westphal, “The Use Of Cepstral Means In Conversational Speech Recognition (1997)”, Proc. Eurospeech, pp. 1143-1146, 1997.

[114] I. Santamaría et al., “Robust Matched Filtering in the Feature Space”, Proc. EUSIPCO, 2005.

[115] J. Sýkora et al., “Design of Mathematical Model of Telephone Channel and Its Using For Speaker Recognition”, SMADA, 200x.

[116] X. Lei et al., “Robust Feature Space Adaptation for Telephony Speech Recognition”, Interspeech-ICSLP, 2006.

[117] L. Müller and J.V. Psutka, “Selection of an optimum speech parametrization for continuous speech recognition system using a telephone channel”, Proc. SCI 2001, vol. VI, pp. 542-545, 2001.

[118] M. Šimandl and J. Sýkora, “ Structure Selection and Parameter Estimation of Telephone Channel Model”, Proc. of Nostradamus, 2001.

[119] D.E. Knuth, “The Art of Computer Programming”, vol. 2: Seminumerical algorithms, pp. 190-192. 2nd Ed., 1980.

[120] The New Web Portal for Advanced Voice Quality Testing in Telecommunications, `http://www.pesq.org`

[121] ITU-T Recommendation P.862, Perceptual Evaluation of Speech Quality (PESQ), 2002. Available at `http://www.itu.int/rec/T-REC-P.862/en`

[122] M.S. Bazaraa et al., “Nonlinear Programming: Theory and Algorithms”, 3rd Edition, Wiley Interscience, John Wiley & Sons, 2006.

[123] I.A. Akbar, Virginia Polytechnic Institute and State University, “Markov Modeling of Third Generation Wireless Channels”, MSc Thesis, 2003

[124] F.J. Silveira, Universidade Federal do Rio de Janeiro, “Previsão de Estadísticas de Perdas de Pacotes Usando Modelos de Markov Ocultos”, MSc Thesis, 2006.

[125] J.H. Kang, University of Michigan, “Iterative Estimation And Decoding For Channels With Memory”, PhD Thesis, 1999.

[126] J. Aráuz et al., “Discrete Rayleigh Fading Channel Modeling”, Wireless Communications and Mobile Computing, vol. 4, issue 4, pp. 413-425, 2004.

[127] J. Chebil et al., “Combined Source Channel Decoding For Image Transmission Over Fading Channels”, APCC 2003, vol. 1, pp. 297-301, 2003.

[128] F. Silveira and E. de Souza e Silva, “A Method For Predicting Packet Losses With Applications To Continuous Media Streaming”, Proc. SBRC, 2006.

Vita

Alexander Gabriel Sklar Perl was born in Tel Aviv, Israel on January 19, 1981. His parents are Julio H. Sklar and Miriam Perl. He received his elementary and secondary education at the Escuela Integral Hebreo Uruguaya and the Instituto Ariel Hebreo Uruguayo in Montevideo, Uruguay. In March 2000 he entered the Universidad de la República in Uruguay from which he graduated with a System's analyst degree in Computer Science in 2004, and the Electrical Engineer and the Engineer in Computer Science degrees simultaneously in 2005. His senior year project was on Low-cost Biped Robot Design and Construction, where a novel state-machine based approach was proposed for implementing robot locomotion. He worked from 2004 to 2006 at the Information Technology Department at the School of Engineering of the Universidad de la República as a Systems Administrator.

In August 2006 he was admitted to the Graduate School of the University of Miami where he was granted a master's degree in Electrical Engineering in December 2007. As of October 2007 he works for Microsoft Corporation in Redmond, WA.

Permanent Address: 3300 NE 191st St. Apt. 1117, Aventura, Florida, 33180.