Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Speech Signal Processing

Prof. Dr.-Ing. Timo Gerkmann

Compilation date: April 3, 2018

Imprint:

Authors:	Prof. Dr.-Ing. Timo Gerkmann
Publisher:	Universität Hamburg
Edition:	Second edition (2017)
Layout:	Daryl Kelvasa
Copyright:	© 2017 Timo Gerkmann. Any unauthorized reprint or use of this material is prohibited. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system without express written permission from the author/publisher.

Learning objectives

- Speech production: How is speech produced by humans?
- Speech perception: How do we perceive speech signals?
- Speech synthesis: How can we produce speech synthetically?
- Speech analysis: What are the most important parameters of speech and how can we represent them?
- Speech coding: How can we code speech efficiently?
- Speech enhancement: How can we improve noisy speech?
- Speech recognition: How can we automatically rocognize speech by computers?

This Script is based on a transcription of the lecture *Speech Signal Processing* held by Prof. Dr.-Ing. Timo Gerkmann. Particular thanks goes to Daryl Kelvasa for his great job in putting my words, slides, and blackboard drawings into text, figures, and equations.

Timo Gerkmann, Hamburg 17.01.2018

Contents

1 — Introduction

Learning objectives

- Introduction
- Speech Production
- Source Filter Model
- Hearing

1.1 Introduction

Humans have evolved with the unique ability to use their lungs and vocal tracts to produce sounds that convey information. Being able to interpret these sounds allows us to pass down and impart knowledge and information about the world in which we live. Because speech is such an essential human trait, the advances made in speech signal processing have led to an exponential growth of our communication devices and have changed the world in which we live.

Already in 1939, Homer Dudley employed the source filter model of speech production in the development of his *Voder*. This machine produced artificial speech by employing an excitation signal that mimicked the signal produced by the lungs and the vocal chords. Band pass filters modeling the resonances of the vocal tract were used to filter the excitation signal in order to produce comprehensible speech. In present day, variations of this model are implemented in our cell phones and other speech transmission technologies. Digital signal processing plays a critical role in the efficient analysis, coding, transmission, enhancement and automatic recognition of speech.

Another benefit of signal processing is for people suffering from hearing loss. The input signal of a hearing aid must be digitally processed so that the output signal is comfortable for the user and fits the current listening situation. The usage of beamformers and different audio scenes allow for better speech comprehension in noisy environments while still being able enjoy music in a concert. Perhaps one of the most amazing speech processing technologies is the cochlear implant in which audio signals are converted into electrical impulses that directly stimulate the auditory nerve. The success of the cochlear implant in allowing the deaf to hear their first sounds is yet another remarkable achievement of speech signal processing.

The purpose of this class is to study some of the underlying processes involved in speech production and perception. This a priori knowledge will then be used to develop algorithms that allow us to digitally manipulate speech in order to perform certain tasks. These tasks are the foundation for the technologies that allow us to help the hearing impaired hear again and to communicate seamlessly with people across the world.

1.2 Speech Production

Speech production begins with the lungs which provide the airflow and therefore the energy required to produce speech. As this energy flows through the larynx, the vocal chords can either vibrate and periodically modulate the energy to produce a *voiced speech sound* or remain stationary to produce an *unvoiced speech sound*. This excitation signal then passes through the vocal tract in which different positions of the tongue, lips, jaw, etc., each correspond to unique resonances. The interaction of the excitation signal with the vocal tract produces the different sounds that are understood as speech. This was already noted by Isaac Newton in 1665 who wrote: "*The filling of a very deepe flaggon with a constant streame of beere or water sounds the vowells in this order w, u, ω, o, a, e, i , y*". The different levels of the liquid result in resonances that change our perception of the sounds created when filling the glass.

Figure 1.1, depicts the different anatomy involved in speech production. Airflow passes through the *trachea* which connects the *pharynx* and *larynx* to the lungs. The larynx contains the *glottis* and *vocal folds* that control the volume of speech and pitch of voiced speech by means of rapid vibrations. Once the modulated air leaves the pharynx, the mobility of the tongue and lips allow for fast changes in the geometry (and thus the resonances) of the vocal tract that produce the rapidly varying spectrotemporal properties that constitute different vocal sounds. The main two sections of the upper vocal tract are the oral and nasal cavities that are separated by the *velum*. An open velum allows the airflow to resonate in both oral and nasal cavities to produce speech that is commonly described as "nasal'.

Many unique speech sounds exist for the thousands of diverse languages spoken in the world. The most obvious distinction that can be made between all vocal sounds is to categorize them by their excitation. Vibrations of the vocal chords in voiced speech production create longer periodic vowel sounds. Short bursts of air corresponding to a noisy excitation are still filtered by the vocal tract to produce unvoiced speech sounds such as the fricative [sh]. *Plosives* are examples of unvoiced speech sounds in which there is a complete constriction of the vocal tract and then a sudden opening such as [k], [p], and [t]. Although the distinction between unvoiced and voiced speech is convenient for classification and modeling, many speech sounds contain characteristics of both excitations. These mixed excitation signals combine bursts of both types of excitations to produce sounds such as [v] or [z].

Figure 1.2 depicts time domain representations of the different speech sounds. The periodic excitation that produces a voiced speech sound is easily identified by a periodic structure in the temporal plot. The distance between two major peaks in the time-domain plot is called the *fundamental period*. This value represents the short amount of time during which the vocal chords open, close, and re-open. The airflow produced by unvoiced speech signals is more turbulent as it lacks this periodic modulation of the glottis. The result is an excitation with a more diverse spectral content which can be identified by the large amount of zero crossings in

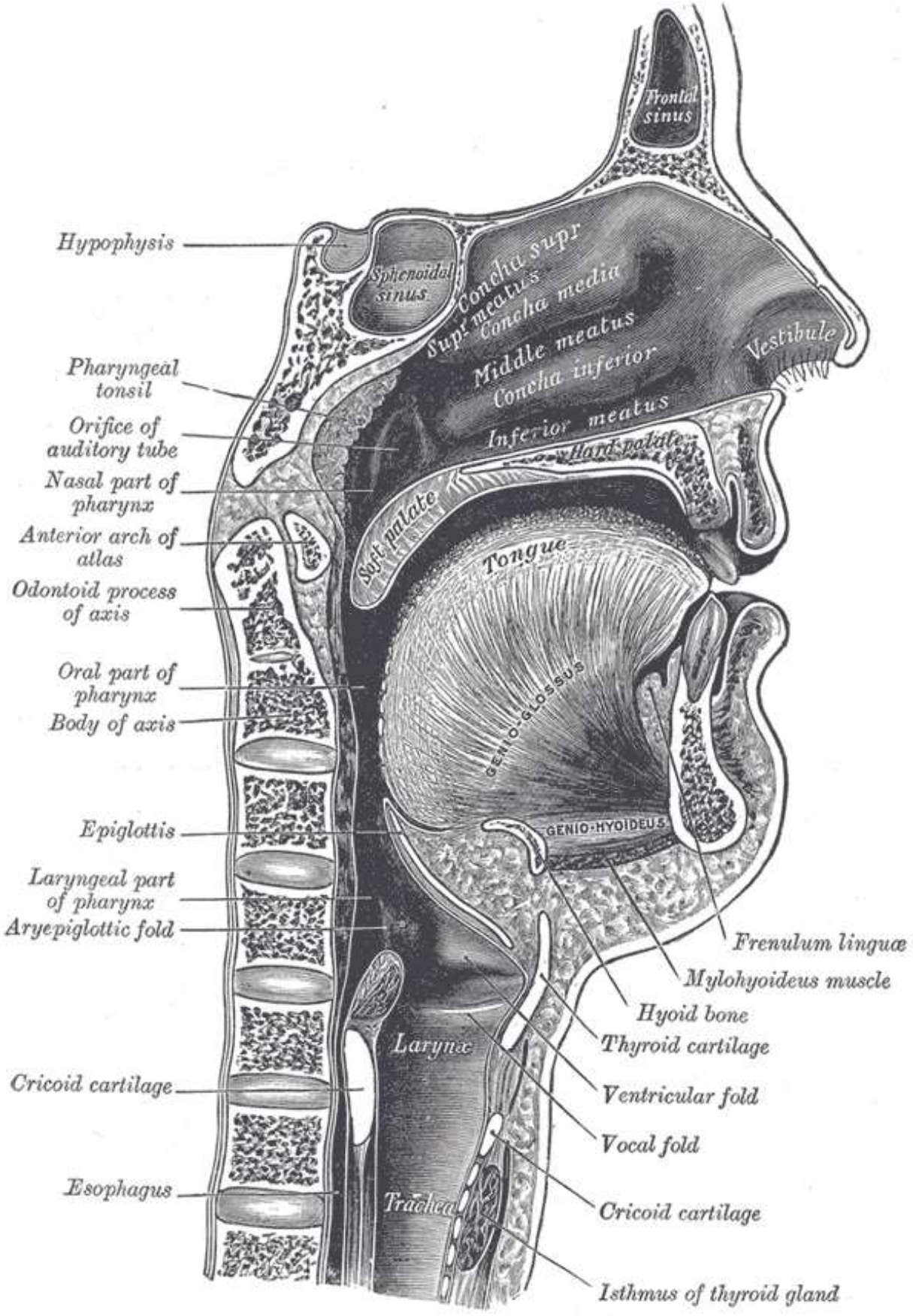
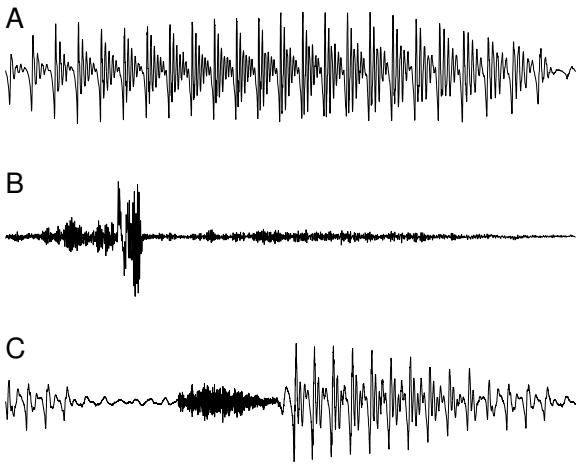


Figure 1.1: Anatomy of the vocal tract. [10]



A) voiced B) unvoiced C) transition vowel-plosive-vowel

Figure 1.2: Time domain signals of voiced and unvoiced speech.

the time domain plot.

Speech sounds convey meaning, however the complex nature of speech does not allow for a simple one-to-one mapping of the speech sound to an intended meaning, thus making it necessary to develop a system to classify these sounds in a linguistic sense. A *phone* is defined as the smallest speech segment with distinct physical or perceptual properties. These distinctions can be obvious as in the difference between the [k] in ”cat” and the [p] in ”pat”, or not so obvious as in the [p] in ”pin” and the [p*] in ”spin”. The former pair may sound similar, however they contain spectrotemporal differences that make them distinguishable. Despite these differences, both words are still comprehensible after interchanging the two sounds, thus constituting a *phoneme*. Notice that this is not the case for the former pair of phones. Phonemes are the smallest segments of speech that can change the meaning of a word. Different realizations of a phoneme are called *allophones* meaning that one allophone is one of the many possible phones that can constitute a phoneme. The words ”cat”, ”kit”, ”school”, ”skill” all contain the phoneme [k] but are pronounced differently due to co-articulation effects from the different vowel transitions. These different phones would therefore form a set of allophones for the phoneme [k].

The large number of allophones that belong to a given phoneme can be explained by *co-articulation*. The position of the vocal tract cannot be changed instantaneously, and the result is a smooth transition of the tongue from one position to the next. The [n] in the word ”hen” is produced by placing the tongue behind the front teeth to create what is known as an *aveolar* sound. However, the [n] in the word ”tenth” is followed by the dental sound [th] that results in it being pronounced more dentally. This difference in articulation distinguishes the two as different phones, however they are both allophones of the same phoneme [n].

Different parts of the vocal tract are used to generate the different phonemes that constitute the thousands of languages spoken across the world. Natural human languages have between 10 and 80 phonemes that can be characterized by the manner in which they are articulated: voiced/unvoiced and place of articulation. The place of articulation is defined by the position of

Figure 1.3: Left: Schematic drawing representing the place of articulation corresponding to phoneme production. Right: Tongue positions to characterize the production of the cardinal vowels. [9]

the tongue when the speech sound is produced. These different postions can be seen in Figure 1.3. These methods of distinguishing between phoneme production allow for a phonetic alphabet that are language independent. Figure 1.4 shows this phonetic alphabet for consonants distinguished by place and type of articulation. A language independent description of the vowel phonemes can also be created by characterizing vowel sounds by the relative position of the tongue when they are generated. The *primary cardinal vowels* are a set of phonemes created by a two-dimensional mapping (high/low and front/back) of tongue position in the oral cavity to the corresponding vowel sound. This mapping is used to characterize the respective phonemes shown in Figure 1.4. One axis depicts the positioning of the tongue from back to front, and a different axis depicts the opening of the mouth. Primary cardinal vowels can also be distinguished from the *secondary cardinal vowels* which are less common and more difficult to pronounce. The main difference between the primary and secondary cardinal vowels is the shape of the lips, which can be either open or round.

Prosody is another important characteristic of speech that encompasses the rhythm, stress, and intonation of an utterance. Although this course will place most of the focus upon the spectral content of speech, the temporal characteristics of speech that are described by prosody still carry important information. Differences in prosody can for instance constitute the difference between a question and a statement by simply raising the fundamental frequency at the end of the sentence.

”I’m going the *bank*”. (Spoken normally)	$\Longrightarrow$	Declarative statement of intent.
”I’m going the *bank*”? (Pitch shifted high)	$\Longrightarrow$	A question is being asked.

Emphasis on specific words can also greatly change the meaning of a sentence.

”Put the *green* ball on the table”.	$\Longrightarrow$	There are multiple balls of different color.
”Put the green *ball* on the table”.	$\Longrightarrow$	There are multiple objects that are all green.

THE INTERNATIONAL PHONETIC ALPHABET (revised to 2005)

CONSONANTS (PULMONIC) © 2005 IPA

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k g	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

VOWELS

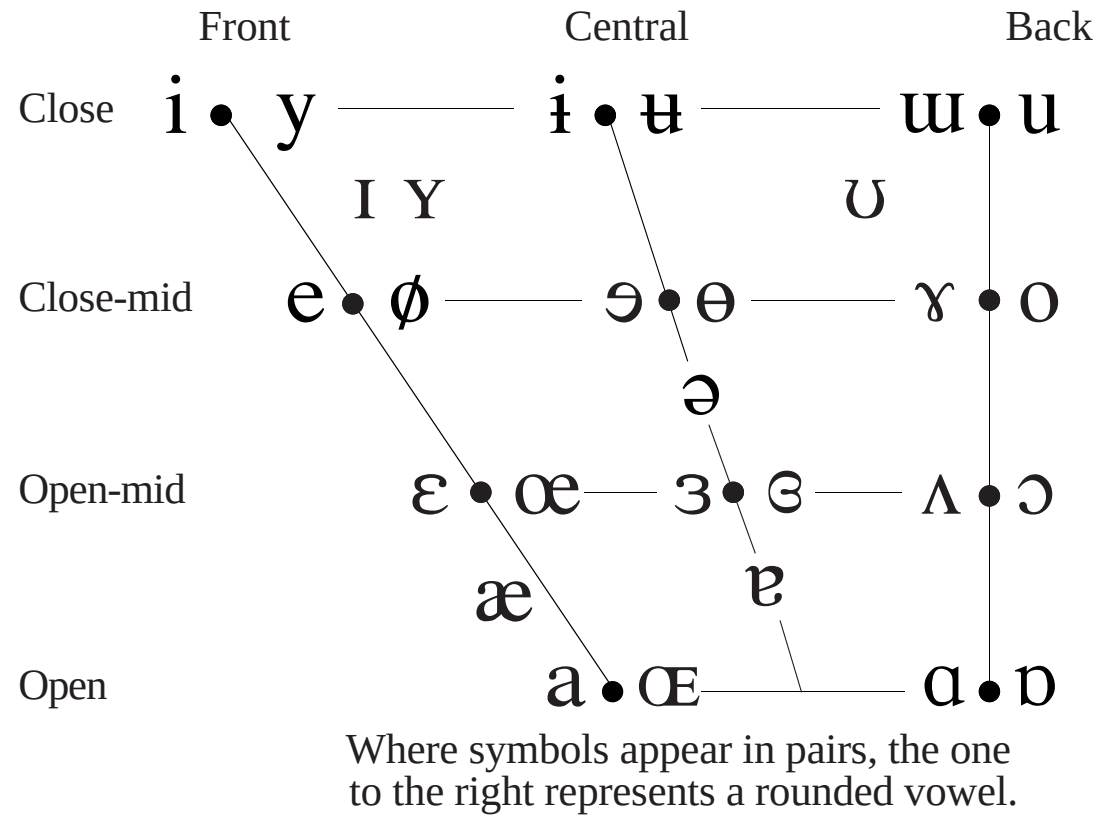


Figure 1.4: Top: Chart of international phonemes and their corresponding places of articulation. Bottom: Chart of tongue positions and the corresponding cardinal vowels. [11]

Information about the emotional state of the speaker is also carried in prosodic cues. Yelling generally instills a sense of urgency in the listener. Perhaps there is some emergency that the speaker is warning about. This reaction in completely different from that produced when the same utterance is whispered.

”I’M BEING FIRED”. (yelled)	$\Longrightarrow$	The speaker is expressing outrage.
”I’m being fired”. (whispered)	$\Longrightarrow$	This is private information for just the listener.

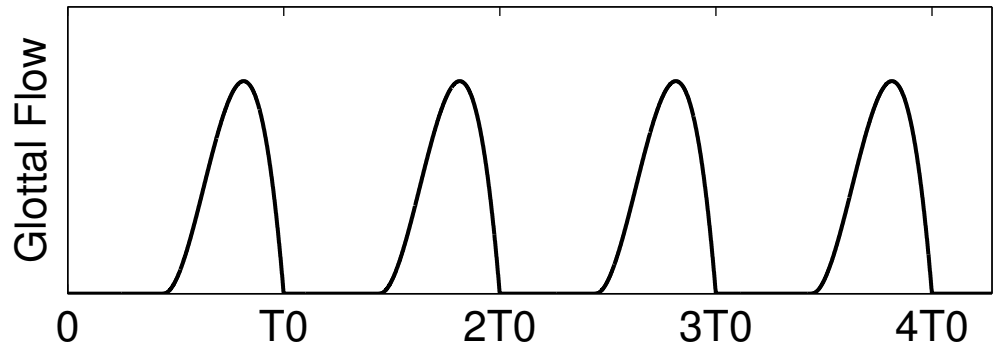


Figure 1.5: Glottal flow as a function of time.

1.3 Source Filter Model

Section 1.2 introduced the underlying anatomical mechanisms behind speech production. The lungs produce energy, in the form of airflow passing through the larynx. This airflow is either modulated by the oscillating vocal chords to produce voiced speech or allowed to flow into the vocal tract to produce unvoiced speech. The current positioning of the jaw, lips, tongue, etc., spectrally shapes this energy to produce the final speech sound. This process can be represented formally by the *source-filter model* so that is can be analyzed and computationally reproduced. In this model, the post-larynx airflow is the source, and the resonance frequencies of the vocal tract constitute a mathematical filter. A very critical assumption of the model is that these two processes remain independent of each other.

Due to the fundamental differences in voiced and unvoiced speech, the ”source” part of the model requires two separate methods for modeling the excitation signal. Voiced speech is produced from a modulated excitation signal that results from the periodic opening and closing of the vocal chords. This periodicity is shown in Figure 1.5 as the glottal flow behind the larynx as a function of time. The process begins with a closed glottis and as energy is produced by the lungs, the glottis is forced to open due to an increased pressure that builds up at the base. The opening of the glottis allows this pressurized air to escape and Bernoulli’s principle states that the increased airflow results in a decrease in pressure. Because the vocal chords are under constant tension, this decrease in pressure allows them to snap shut and begin the process again. The result is a voiced excitation that is modulated at a fundamental period corresponding to this periodicity. To model this mathematically, a time-domain pulse train (or Dirac comb) can be used to model the opening and closing of the glottis by using a peak to peak distance corresponding to the fundamental period, T_0.

The turbulent airflow of unvoiced sounds lacks this periodicity, so it is better described by a random process. Figure 1.6 shows the time course of a white Gaussian signal, its histogram, and spectrum. The spectrum of the white Gaussian noise is flat meaning that it is made up of an equal

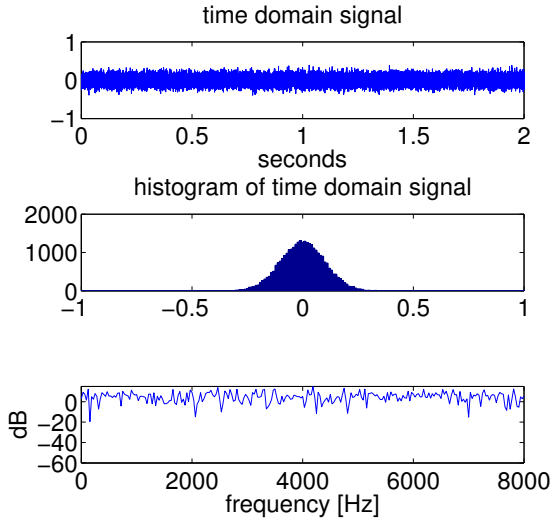


Figure 1.6: (Top) Time, (Middle) statistic, (Bottom) and frequency domain representations of white Gaussian noise.

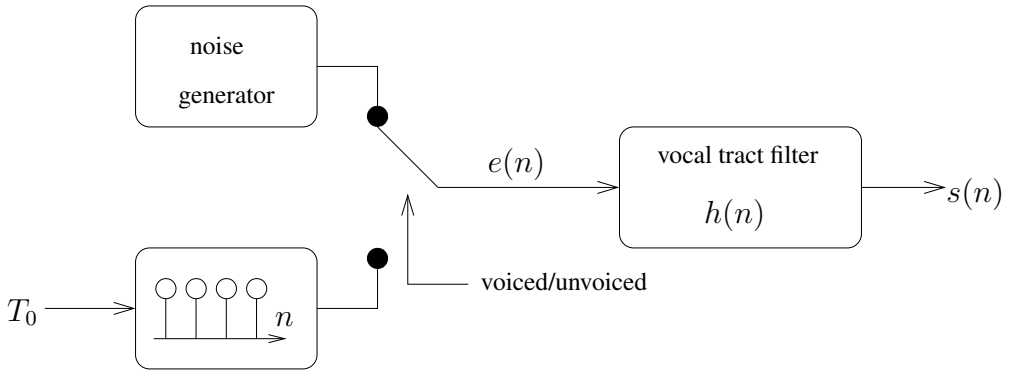


Figure 1.7: Simplified model of speech production.

distribution of all frequencies. This is analogous to "white" light in optics, which also consists of an equal distribution of frequencies corresponding to all the colors of the visible spectrum. To model an unvoiced speech signal, a noise generator can be used to randomly choose numbers from a Gaussian distribution. The two forms of excitation can now be described by the simple model in Figure 1.7. It can be seen that it is necessary to have a switch that chooses between the two forms of excitation. This implies that the model requires some form of detector that can determine the type of excitation that is present in the current speech signal. Of course, this model is limited in that it cannot produce the mixed excitation signals that were introduced in Section 1.2. One can then imagine more complicated models in which a mixed excitation is produced by a weighted summation of the two excitation signals.

The vocal tract can now be modeled as a filter through which the chosen excitation signal passes. This is done by greatly simplifying the complexity of the vocal tract to the tube model in Figure 1.8. One section corresponds to the nasal cavity in which the output is the nose, whereas the bottom represents the oral cavity in which energy escapes through the mouth. The velum regulates the passage of air between the oral and nasal cavities, and this can be modeled by a simple switch. Much like a woodwind instrument, the shape and length of the tube segments corre-

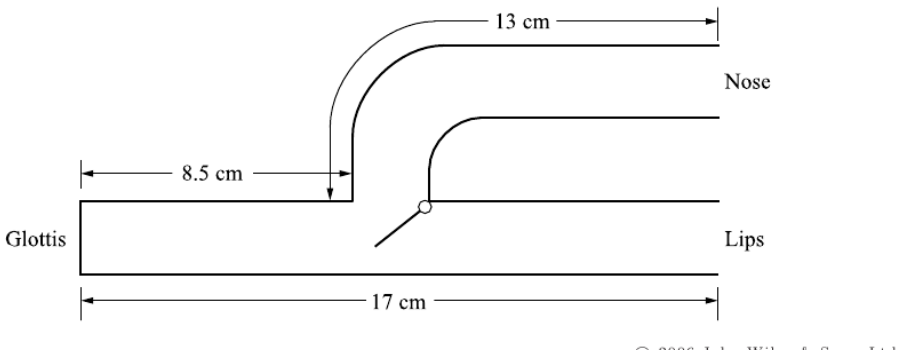


Figure 1.8: Simplified tube model of the vocal tract.[7]

spond to resonance frequencies that concentrate the energy of the excitation signal into certain frequency ranges called *formants*. These resonance frequencies can be mathematically described as a filter in what is referred to as the *vocal tract transfer function*. By changing this vocal tract transfer function over time, synthetic speech production can be achieved.

Mathematically speaking, this filtering is represented by a convolution of the excitation signal and the vocal tract impulse response in the time domain or a multiplication of the excitation spectrum and vocal tract transfer function in the frequency domain (Figure 4.4). After the multiplication, the spectrum of the final speech sound is what would be seen upon a Fourier transform of the recorded speech. The spectrum of the excitation signal consists of the fundamental frequency and a set of harmonics at integer multiples of the fundamental frequency. The formants appear as spectral peaks in the transfer function that correspond to the resonances in the vocal tract. Formants contain important information because they are used by the brain to distinguish between different speech sounds. The influence of the excitation signal and the vocal tract are both present in the final signal. It is very important to understand that these are two independent signals in the model. The final signal in Figure 4.4 reveals that the formant *peak* is not always present in the final speech sound. This is because in a sense, the excitation signal samples the transfer function at discrete points, namely the fundamental frequency and its harmonics. As a consequence, the fundamental frequency and its harmonics are the only frequencies in the final signal. These are then spectrally shaped by the vocal tract transfer function which concentrates the energy of the harmonics into the corresponding formant frequency regions. When the fundamental frequency or a harmonic falls precisely at a formant peak, the transmitted energy can be maximized. This is often exploited by singers.

The fundamental frequency and formant frequencies of the vocal tract are two fundamentally different things and it is important to distinguish between the two. The fundamental frequency contains information with respect to intonation and prosody, however it carries no real information with respect to meaning. Also, the fundamental is a property of the excitation signal which is the active energy source of the modeled system. In contrast, the vocal tract transfer function is a passive filter and adds no new energy into the system. Instead, the filter allows the harmonics of the excitation signal to resonate within the formant frequency regions to produce a spectrally colored phoneme. In particular, the first two formants are essential for vowel recognition and can be used classify vowel phonemes as seen in Figure 1.10.

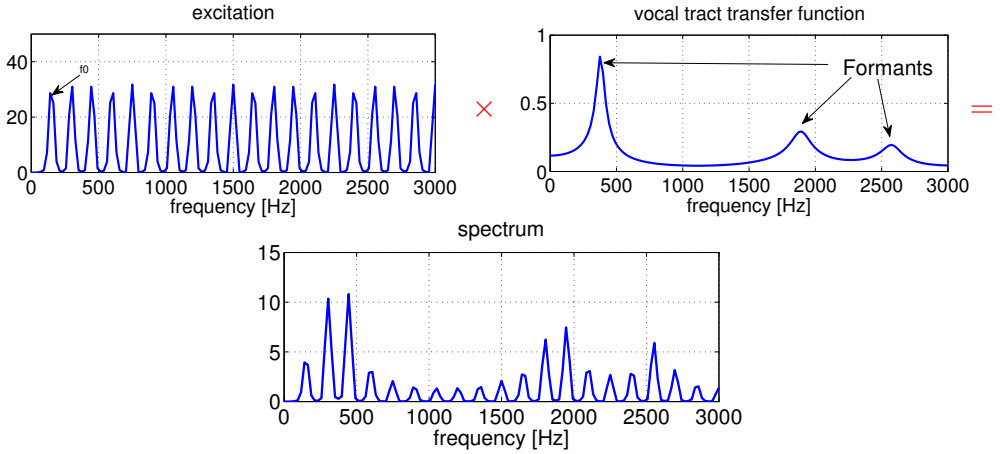
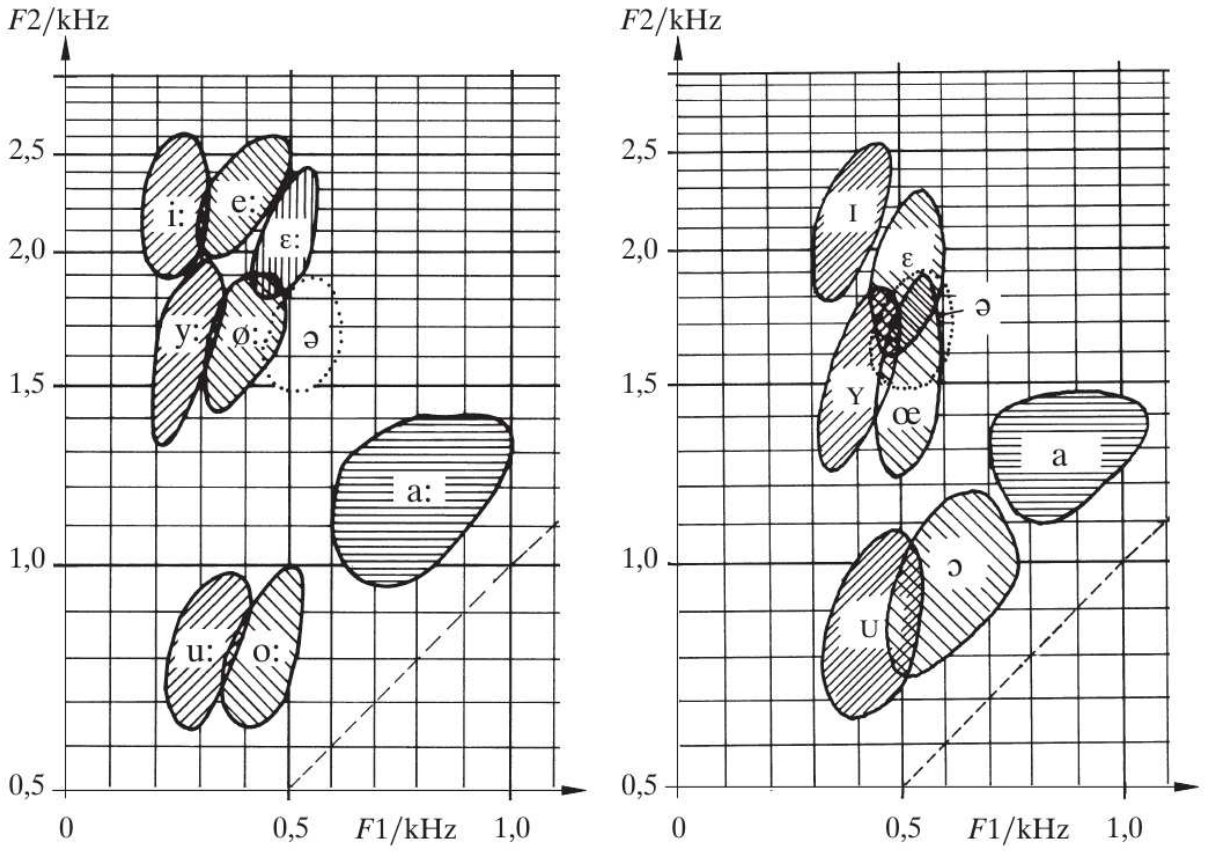


Figure 1.9: Top: Spectrum of voiced excitation signal. Middle: Vocal tract transfer function. Bottom: Spectrum of excitiation signal filtered by the vocal tract.

This section has introduced a formal model for the speech production process in which mathematical operations can be performed using a set of parameters to synthetically produce speech. To derive these parameters, it is necessary to compute whether a speech frame contains voiced or unvoiced speech, the fundamental period if the speech is voiced, and the vocal tract transfer function. A large part of this course will deal with extracting these parameters from real speech signals, so that they can be coded, transmitted, and then re-synthesized using this simplified model of speech production.



Quelle: Vary, Heute, Hess (1998): Digitale Sprachsignalverarbeitung, Teubner, Stuttgart

Figure 1.10: Phonemes as a function of first and second formant frequencies.[7]

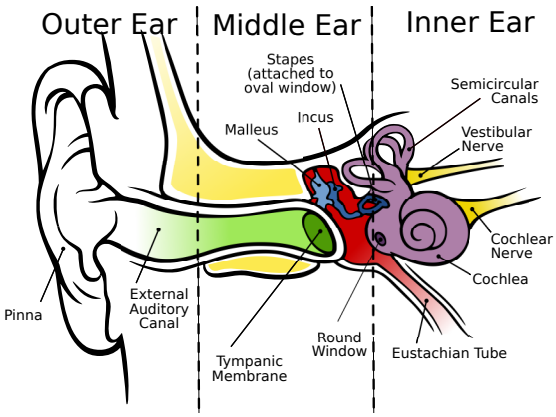


Figure 1.11: The peripheral auditory system. [12]

1.4 Hearing

The physiological components that are involved in the peripheral processes of human sound perception can be divided down into three main parts: the *outer ear*, the *middle ear* and the *inner ear*. The outer ear collects acoustic energy in the form of aerial compression waves and transfers it to the middle ear where it is converted to fluid waves within the inner ear. The inner ear then converts this information into action potentials that the brain can use to interpret the sound. These parts and their corresponding components are depicted in Figure 1.11.

The outer ear consists of the *pinna*, the *ear canal*, and the *ear drum* or *tympanic membrane*. Sound pressure compression waves travel through the ear canal, where they transfer their energy to the tympanic membrane. This membrane is connected to the inner ear by the *malleus*, *incus*, and *stapes*. These small bones transfer energy from the larger area of the tympanic membrane to the smaller area of the *oval window* which is attached to the *cochlea*, a fluid filled spiral shaped cavity within the inner ear. The acoustic compression waves are therefore converted to fluid waves, that produce a traveling wave along the *basilar membrane* within the cochlea. The length of the basilar membrane contains points of varying stiffness that allow this wave to travel until it reaches a frequency dependent position where it can then resonate. The tectorial membrane and the organ of Corti are located on the basilar membrane as shown in Figure 1.13. The resonating basilar membrane creates sheering forces between these two components that result in the stimulation of hair cells which are innervated by auditory nerves. The auditory nerves then fire action potentials that the brain can interpret to perceive sound. Figure 1.12 shows that the peak of a high frequency tone would occur closer to the base of the basilar membrane, whereas the resonances of lower frequencies, would occur more toward the apex. This implies that hair cells close to the base encode the high frequency content of the signal, while the hair cells at the apical end encode the low frequency content of a signal. This tonotopic representation

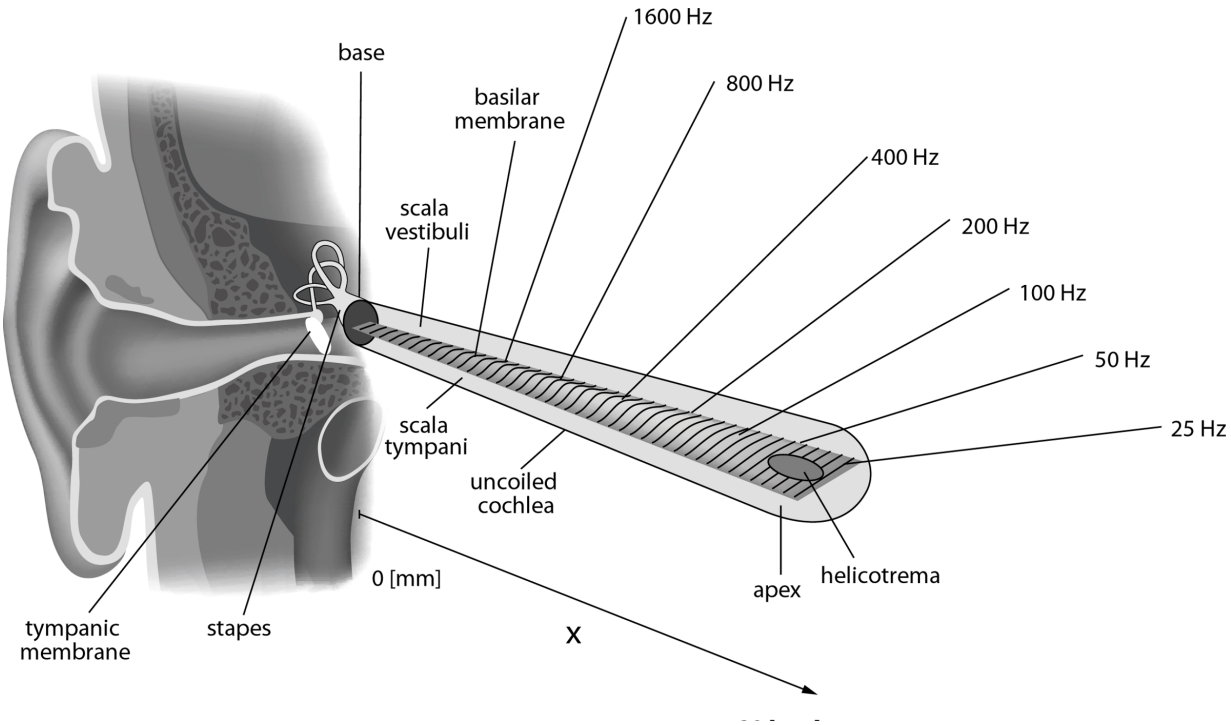


Figure 1.12: Frequency place coding along the cochlea. [15]

is a form of frequency place coding that results in a frequency analysis of the signal analogous to the techniques that we will be implementing in this course. This is also the reason why this frequency analysis is such an intuitive signal representation.

Figure 1.14 depicts the auditory sensation area for a normal hearing person as a function of frequency and level. It can be seen that more energy is required to perceive lower frequency sounds. Humans are able to perceive frequencies between $20\,\mathrm{Hz}$ to $20\,000\,\mathrm{Hz}$ and begin to have problems with higher frequencies at older ages. Figure 1.14 also shows the threshold of pain that defines the sound level at which permanent hearing damage occurs when the listener is exposed to supra threshold sound for an extended period of time. Interestingly, speech formants correspond to a range of the auditory sensation area where the ear is most sensitive. This implies that the ear has evolved to be optimized for speech perception (or the other way around!). The hearing impaired, or those suffering from sensory neural hearing loss, maintain similar threshold levels of pain as normal hearing individuals, however their threshold of hearing is increased. A hearing aid could then be used to amplify softer sounds, but simply implementing linear amplification would result in output signals with levels beyond that of the threshold of pain. Compression algorithms can then be used in hearing aids, so that softer sounds are amplified more than louder sounds. However, this process significantly decreases the SNR and therefore requires some noise reduction to enhance the final signal.

It is interesting to note that there are different types of hearing losses that are associated with different methods of treatment. In a conductive hearing loss, the small bones within the middle ear can stiffen over time so that sound is not properly conducted to the inner ear. Sound is still perceivable, but attenuated, meaning that a hearing aid can be nicely used to amplify the sound and treat the condition. Conductive hearing losses are generally easier to treat than sensory

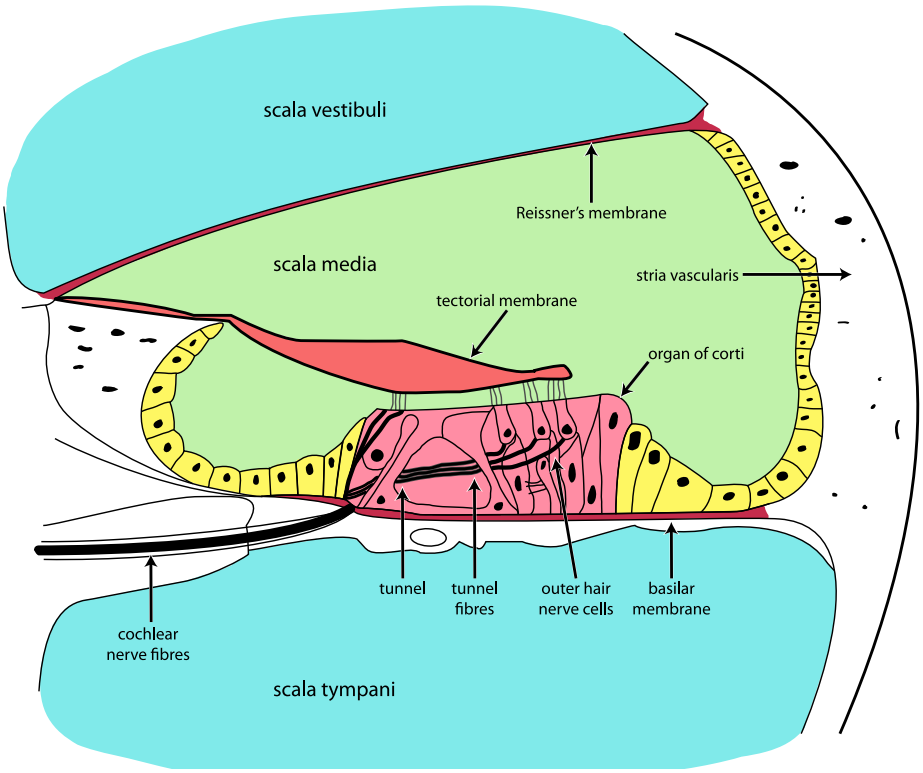


Figure 1.13: Cross section of the cochlea. [13]

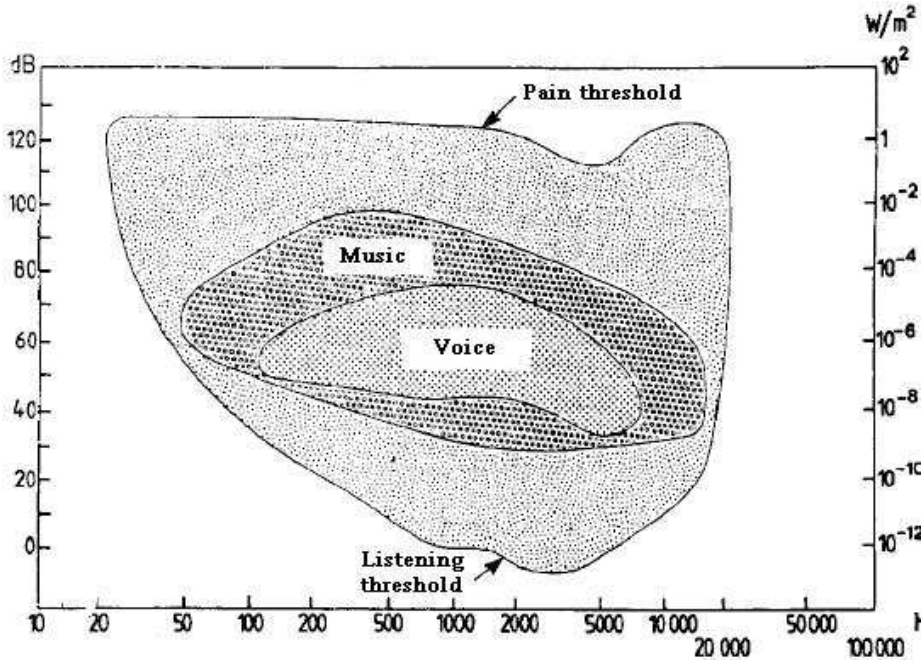


Figure 1.14: Audible range in SPL as a function of frequency. Frequency range of common sounds are color coded. [14]

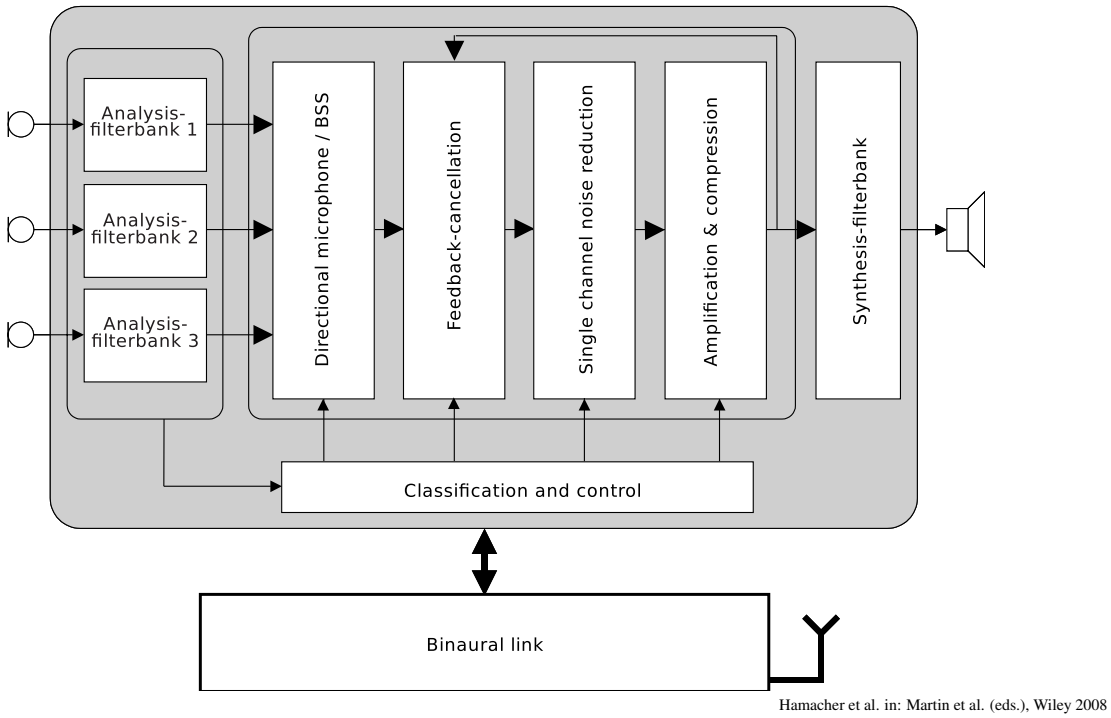


Figure 1.15: Flow chart depicting the processing stages of a hearing aid. [8]

neural hearing losses in which inner hair cells die and the sound cannot be transferred into a neural code. This can result from trauma induced by long term exposure to high level sound or simply by old age. New hair cells do not grow, nor do they regenerate, so that once the hair cells are damaged, they are gone forever. As a result, soft sounds are no longer perceivable, while at the same time the threshold of pain remains the same. This produces a reduced area in the auditory sensation map in Figure 1.14. Sensory neural hearing loss is also problematic because it is often accompanied by a decrease in frequency resolution of the human ear. Because the frequency resolution of the auditory perception is is not sufficient to separate speech from noise, one could theoretically still perform well on an auditory test, but have trouble perceiving speech in noisy environments. This is another application of noise reduction algorithms that can greatly aid the hearing impaired.

Current hearing aids commonly implement multiple microphones. Figure 1.15 depicts a multi-channel setup in which the multiple microphones are all connected to an analysis filter bank that implements a time-frequency analysis similar to the processing performed by the cochlea. Multiple microphones allow for a directional processing that can be used to attenuate sounds originating from a specific direction while sounds originating from a different direction remain transparent. Users can choose to whom they want to listen simply by looking at that person and placing them within the beam pattern. A single hearing aid has closely spaced microphones and results in a rather broad beam. However, binaural hearing aids contain microphones at both ears that can be used to form a more narrow beam. This system requires that information is transmitted between hearing aids which can be done wirelessly, however this requires an increased energy consumption. In the simplest case, control information, such as level control, could be transmitted in order to ensure that both hearing aids are in the same state, e.g. both have the same volume settings, and both apply the same enhancement scheme. Many hearing aids have

auditory scene analyzers that use different processing algorithms for different auditory scenes. If one wanted to understand speech in a noisy environment, then they simply turn on the noise reduction in order to perceive speech better. This same noise reduction would not be so desired at a concert where one wanted to enjoy music. Finally, a feedback cancellation is employed in order to avoid an undesired whistling of the hearing aids, when signal components are amplified in a feedback loop.

2 — Pitch

Learning objectives

- Fundamental Frequency Estimation

Chapter 1 introduced the underlying physiological mechanisms behind speech production and sound perception. Furthermore, the source-filter model was also introduced as a simple model of speech production. The goal now is to take a given speech signal and extract the necessary parameters required for this model. This chapter will focus on creating a parametric representation of the excitation signal, or the "source" in our model. The necessary parameters to accomplish this are: voiced or unvoiced judgment, the speech fundamental frequency, and the energy of the particular speech segment. Given these parameters, the excitation signal for an unvoiced speech segment can be modeled by white Gaussian noise while a voiced speech segment can be modeled by a pulse train with a pulse distance of the fundamental period, T_0, respectively.

Figure 4.4 shows that the excitation signal is readily identifiable in the spectrum of the vocal tract filtered signal in the form of the fundamental frequency and its harmonics. The fundamental frequency is an important parameter in speech signal processing that is required by many speech processing algorithms. It is therefore of great interest to learn how to efficiently and accurately extract this feature. Without it, speech is comprehensible, however the synthesized speech is monotonic and robotic. Furthermore, it is difficult to judge the intended emotion of the speaker and to distinguish between questions and statements. Also for speech enhancement, knowledge of the speech fundamental frequency can help to distinguish speech from noise. Noisy speech can be thought of as white noise filling up the gaps of a spectrogram. If the fundamental frequency is known, the noise could be attenuated in time-frequency bins where speech is not present, while the bins containing energy from the fundamental frequency could be preserved.

Very often in speech processing, the word pitch is used synonymously with fundamental frequency. However, it is important to realize that pitch is a perceptual quantity. The measure of pitch is therefore based on listening experiments with human subjects. This is in contrast to the fundamental frequency, which is a physical parameter, usually obtained using instrumental measures. The difference between pitch and fundamental frequency is best demonstrated in experiments where the loudness of a pure tone, a sinusoid with a constant fundamental frequency, is adjusted and subjects are asked to comment on the pitch of the tone. Subjects very often notice a change of pitch with a change in loudness, but a constant fundamental frequency [6]. This demonstrates that pitch is a qualitative phenomenon, however fundamental frequency is a quantitative, measurable parameter of sound.

The typical range of the fundamental frequency of voiced speech is from $40\,\mathrm{Hz}$ to $600\,\mathrm{Hz}$. $600\,\mathrm{Hz}$ is a bit on the high side and is something that would only really be seen in children. Typically, speech fundamental frequencies are around $100\,\mathrm{Hz}$ for male speakers, and about $200\,\mathrm{Hz}$ for female speakers.

The *residual effect* is a phenomenon in which we perceive the fundamental frequency of a sound even when this fundamental frequency is not present in the signal. Applying a $300\,\mathrm{Hz}$ high-pass filter to a male speech signal with a fundamental frequency of $100\,\mathrm{Hz}$ still results in the perception of a $100\,\mathrm{Hz}$ fundamental frequency. A good example, that everyone is familiar with is traditional telephone speech. For historic reasons, only frequencies above $300\,\mathrm{Hz}$ are transmitted, meaning that the fundamental frequency is not present in phone speech. Still, we are able to distinguish between male and the female speakers because the fundamental frequency can be estimated by the distance between successive harmonics. In fact, if two pure tones at $200\,\mathrm{Hz}$ and $300\,\mathrm{Hz}$ are summed and played together, the perceived pitch is $100\,\mathrm{Hz}$. The time domain signal in Figure 2.1 reveals that the sum of the two tones, and the distance between peaks is $0.01\,\mathrm{s}$ (cf. upper right

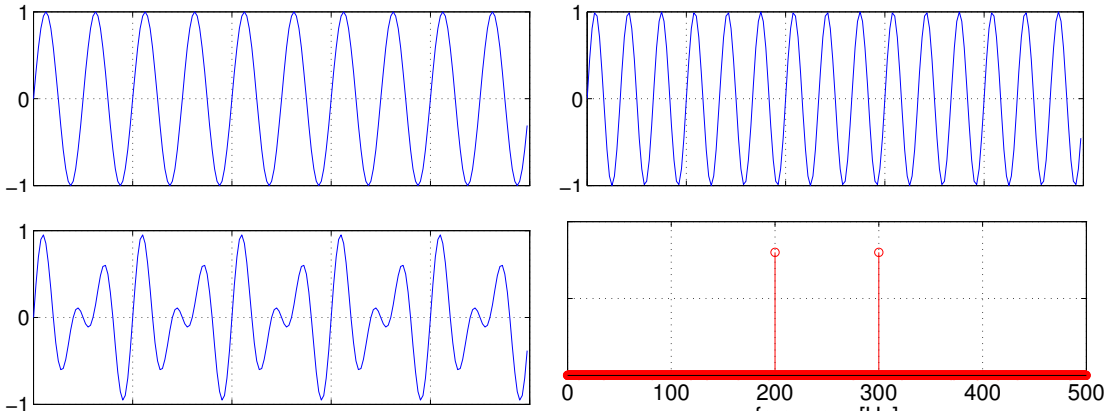


Figure 2.1: Example demonstrating the residual effect. Top: (Left) 200 Hz pure tone. (Right) 300 Hz pure tone. Bottom: (Left) Sum of the two harmonics reveals temporal pattern of fundamental harmonic at 100 Hz. (Right) Fourier transform of tone superposition shows no energy at fundamental harmonic.

of Figure 2.1), or 100 Hz in the frequency domain (cf. lower right of Figure 2.1). Performing a discrete Fourier transform on the time domain signal reveals no energy at 100 Hz because there are no 100 Hz components in the signal. However, the fundamental period is still 0.01 s (upper right of 2.1) and a lower tone is perceived.

In addition to frequency content below 300 Hz, the frequency content above 3400 Hz is also not transmitted in standard telephone speech (e.g. ISDN or GSM). As a consequence it is almost impossible to distinguish some phonemes, such as [s] and [f]. This is the reason why spelling alphabets are commonly used over the phone, e.g. "*c* like Charlie." Despite this, when heard in the context of flowing speech, the speech is still intelligible enough to justify the savings in bandwidth. Efficiently reducing signals down to the minimum bandwidth required to extract the required information is of great interest in order to save costs when transmitting multiple conversations at the same time.

2.1 Fundamental Frequency Estimation

The easiest method of measuring the fundamental frequency would be to take the time domain signal and measure the time between the periodic zero crossings or the periodic peaks. However, this value can vary because we often do not have perfect periodicity in the time domain signal. One could also imagine that there is noise in the signal, making it much more difficult to find the points at which the periodic structure repeats itself. In other words, this method is prone to errors and very difficult to automate. Fundamental period estimators based on this concept are not considered to be very robust. A much more robust estimator of the speech fundamental frequency is based on the the *autocorrelation function* (ACF) of a time domain signal.

Autocorrelation Function: formal definition

$$\varphi_{\mathrm{XX}}(\lambda) = \mathrm{E}(x(n)x^*(n+\lambda)) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u\, v\, p_{x(n)x^*(n+\lambda)(u,v)}\mathrm{d}u\mathrm{d}v \tag{2.1}$$

As seen in Equation (2.1), the autocorrelation function is defined as the expected value of the product of a signal $x(n)$ and a shifted version of itself. If x would be a complex valued signal, then the product would be of the complex signal and the complex conjugate of a shifted version of itself. The complex conjugate is denoted by x^*. However, real-world physical time-domain signals are real-valued, such that the * can be removed.

The expected value is defined as the integral over these two signals multiplied by the joint probability density function of the two signals. This is a formal definition and in practice, the joint probability density function, $p_{x(n)x^*(n+\lambda}$ is generally not available, therefore it must be estimated by replacing the integral with a summation over realizations of the signal resulting in the simpler Equation (2.2).

Autocorrelation Function: estimated

$$\widehat{\varphi}_{\mathrm{xx}}(\lambda) = \frac{1}{N-|\lambda|}\sum_{n=0}^{N-|\lambda|-1} x(n)x^*(n+\lambda) \tag{2.2}$$

The ACF computes the average of the product of the signal and shifted versions of itself. This yields a measure of the self similarity of the signal. The ACF of a periodic time-domain signal at zero lag, $\lambda = 0$, results in the product of two identical signals on top of each other and the ACF is at it maximum value. All of the products would be positive and their sum results in a large positive value that estimates the power of the considered signal segment. If the original signal is shifted by half of a period, then the summation yields zero. The autocorrelation function of a periodic function is therefore also periodic because it begins at a maximum at $\widehat{\varphi}_{\mathrm{xx}}(0)$, decreases, and then achieves another maximum at lags corresponding to the fundamental period, i.e. at $\widehat{\varphi}_{\mathrm{xx}}(T_0)$. Because of the averaging by means of the sum in (2.2), detecting the second maximum $\widehat{\varphi}_{\mathrm{xx}}(T_0)$ yields a rather robust measurement of T_0 and thus of the fundamental frequency $f_0 = 1/T_0$.

It is also important to note that the Fourier transform of the autocorrelation function is called the *power spectral density* (PSD). This is formally defined as

Power spectral density

$$\Phi_X(f) = \sum_{\lambda=-\infty}^{\infty} \varphi_{\mathrm{XX}}(\lambda)\mathrm{e}^{-\mathrm{j}\Omega\lambda} \tag{2.3}$$

The power spectral density can reveal interesting properties of a signal that are not immediately

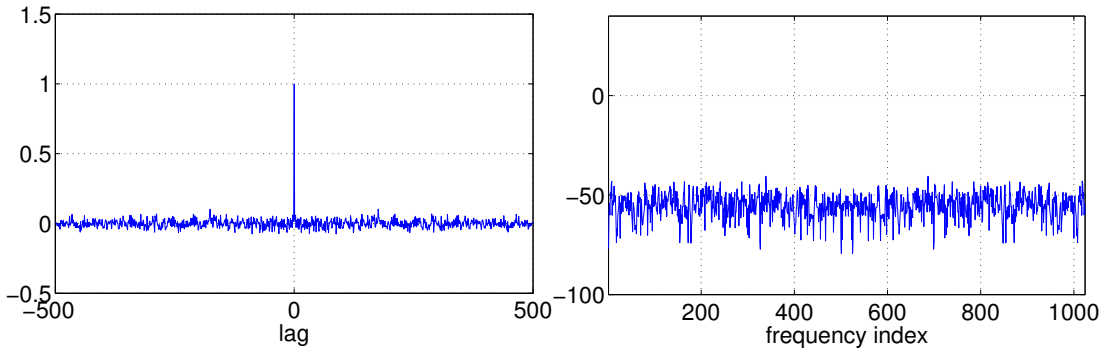


Figure 2.2: Left: Autocorrelation function of a Gaussian white noise signal. Right: Power spectral density of a Gaussian white noise signal.

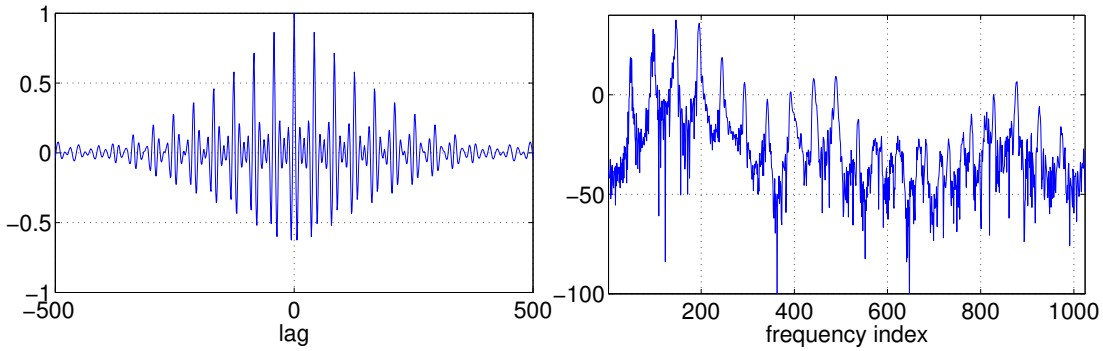


Figure 2.3: Left: Autocorrealtion function of a voiced speech segment. Right: Power spectral density of a voiced speech segment.

seen in the autocorrelation function. To visualize this, we can use the example of an uncorrelated signal. If a signal is *uncorrelated* that means that any two different samples are not correlated, or in other words, the autocorrelation function of a zero mean variable is zero everywhere except at $\varphi_{\text{xx}}(0)$. This can be observed in Figure 2.2 which displays an *estimated* ACF and PSD of an uncorrelated Gaussian noise signal. Both representations reveal some fluctuation because the expected value was approximated by computing an average over a finite number of samples, i.e. we are showing the *estimated* autocorrelation function $\widehat{\varphi}_{\text{xx}}(\lambda)$. From system theory, we will learn that the Fourier transform of a delta peak results in a flat spectrum. Thus, the PSD of an uncorrelated signal should be perfectly flat. Again, as in Figure 2.2 we show that the Fourier transform of the estimated ACF, also the PSD shows some fluctuations. If we were to average several PSD estimates, a flatter spectrum would be achieved. As an uncorrelated signal has a flat PSD, it is also referred to as a *white* signal. Therefore, uncorrelatedness and whiteness are often used synonymously, where the first describes the temporal characteristics of a signal while the latter describes the resulting spectral characteristics.

Speech signals, on the other hand, contain periodicities meaning that successive signal samples are correlated. The ACF of a periodic speech segment is shown in Figure 2.3. It reveals a peak at lag zero, $\widehat{\varphi}_{\text{xx}}(0)$, and peaks at multiples of the fundamental period, $\widehat{\varphi}_{\text{xx}}(kT_0)$. The PSD reveals a first peak at the fundamental frequency $\widehat{\Phi}_X(f_0)$, and then at integer multiples corresponding to the spectral harmonics, $\widehat{\Phi}_X(kf_0)$.

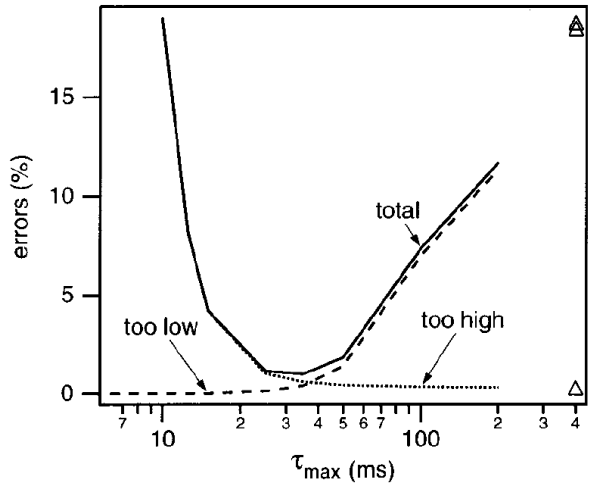


Figure 2.4: Fundamental frequency estimation error as a function of window length. [16]

A very important parameter in estimating the ACF is the window length, i.e. the number of samples, N in Equation (2.2) that are used to approximate the expected value in Equation (2.1). There is a certain trade-off between short and longer window lengths. Longer window lengths allow for multiple periods within one ACF window, thus making the estimation of the fundamental frequency more robust. However, the maximal applicable window length is limited because the fundamental frequency of speech is not stationary and changes over time. Again, note that speaking at a constant fundamental frequency would result unnatural sounding speech. $30\,\mathrm{ms}$ is a typical window length that is long enough to allow multiple periods of the fundamental period fit within one window, but short enough to follow changes in the fundamental period. At a fundamental frequency of 100Hz, there would be roughly three periods within the corresponding ACF window. This can be seen in Figure 2.4 which also shows the estimation error of the fundamental period.

There are also variants of this method, however many pitch estimators are still based on the ACF. A well known estimator, called YIN [16], is based on the following difference function

$$x(t) - x(t + T_0) = 0 \quad \forall t.$$

Similar to the ACF, we take a signal and the same signal shifted by a certain lag, however instead of multiplying, we subtract the two. For a perfectly periodic signal, this difference would be zero at lags corresponding to integer multiples of the fundamental period. To find the fundamental period T_0, we compute the square of this difference function and average over N samples in order to compute $d_{T_0}(t)$.

$$d_{T_0}(t) = \frac{1}{N - |\lambda|} \sum_{n=0}^{N-|\lambda|-1} (x(t) - x(t + T_0))^2.$$

The algorithm for the estimator would try to find the T_0 that minimizes $d_{T_0}(t)$. This method

Method	Error, (%)
ACF	10
Difference	1.95
YIN	0.50

Table 2.1: Percent error of some common fundamental frequency estimation algorithms.[16]

is very much related to the autocorrelation function in the use of shifted versions of the signal and the summation. If we now multiply out this square, we see that the $d_{T_0}(t)$ function actually consists of the autocorrelation function estimate at time $\widehat{\varphi}_{\mathrm{x(t+T_0)}}(0) = \widehat{\varphi}_{\mathrm{x(t)}}(0)$

$$d_{T_0}(t) = \widehat{\varphi}_{\mathrm{x(t)}}(0) + \widehat{\varphi}_{\mathrm{x(t+T_0)}}(0) - 2\widehat{\varphi}_{\mathrm{x(t)}}(T_0)$$

For a perfectly periodic signal, the autocorrelation function at $t = 0$ and the autocorrelation function at $t = t + T_0$ would yield exactly the same result. In that case, the two methods are identical. However in practice, this is not the case. Error measurements of some commonly implemented fundamental frequency estimators are shown in Table 2.1 and it can be seen that the ACF method produces a much larger error than YIN, which is another method that is based on the difference function.

3 — Spectral Transformations

Learning objectives

- Fourier transformation (continuous time vs. discrete time)
- Digitization of speech signals (time-amplitude)
- Discrete Fourier Transform (DFT)
- Short-Time Fourier Transform (STFT)
- Spectrogram (narrow-band vs. wide-band)
- Synthesis: Overlap-add technique

3.1 Fourier transformation (continuous time vs. discrete time)

In most practical applications of signal processing, the goal is to modify (i.e. *process*) a signal to some extent. In order to facilitate this, it is important that the properties to be modified are easily accessible. As seen in Section 2.1 the fundamental period is an example of a speech property that is not so easily accessible in the time domain. It can be estimated in the time domain, however it was argued that time domain based algorithms (i.e. peak and zero crossing measurement) are prone to errors. By employing the autocorrelation function, the signal can be transformed into a domain that allows for a more robust estimation of the fundamental period. Another domain in which it is often convenient to analyze the properties of a signal is the spectral domain.

The concept of decomposing a signal into its frequency content is described by *Fourier theory*. To obtain the Fourier representation, a signal is correlated with linearly independent complex exponential eigenfunctions and the results are stored as complex-valued Fourier coefficients. Because any complex exponential function can be written as a sum of cosine and sine functions, this process can also be visualized as a correlation with cosine and sine functions. The linear independence of the complex exponentials means that Fourier analysis preserves all information contained within the original signal in a compact manner. Because no information is added or removed by the analysis, it can simply be considered as a different way of representing the signal. In the process, certain attributes of the signal are made more visible, whereas others are not visible any more.

A pure tone is a tone consisting of a single sinusoid with a certain amplitude, frequency and phase. Because a cosine function can also be represented as a sine with a $\pi/2$ ($90°$) phase shift as $\sin(\alpha) = \cos(\alpha - \pi/2)$, sines and cosines are both referred to as *sinusoidal* signals. One of the key concepts of Fourier theory is that any periodic signal can always be represented as the weighted sum of a fundamental sinusoid that captures the signal's periodicity and integer multiples of this sinusoid known as *harmonics*. The process of determining these weights is called *Fourier series analysis*.

The Fourier series analysis of a rectangular function can be see in Figure 3.1. The lowest frequency sinusoid is the contribution of the fundamental period to the analysis, and the final signal is the sum of the fundamental and weighted successive harmonics. In this example, the fundamental frequency contributes the majority of the energy to the signal and then there is an exponential decay of the weighting for higher harmonics. The rectangular function is the better approximated as the number of harmonics is increased. Because its edges represent a sudden jump in the time domain, one would theoretically require infinitely many harmonics and therefore an infinitly high frequency content in order to model this signal perfectly.

The formal mathematical definition of the Fourier series analysis is shown in Equation 3.1. The value, b_0, is the *DC offset* and is the mean value about which the signal oscillates. Speech can

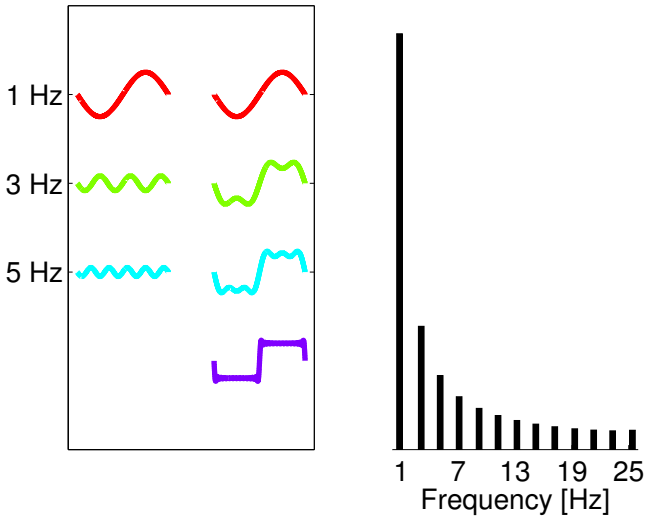


Figure 3.1: Left: Superposition of odd harmonics to approximate a square wave. Right: Spectral content of the square wave.

generally be assumed to be a zero mean signal, meaning that $b_0 = 0$. The periodic signal $x(t)$ is represented by the DC offset and a superposition of weighted contributions of sine and cosine functions at multiples of the fundamental frequency. The weights, b_h and a_h, are determined by correlating the original signal with cosine and sine functions, respectively. However, because the cosine function is even, whereas the rectangular function is odd, for the example in Figure 3.1 there will be zero correlation between the two and all b_h coefficients will go to zero. Therefore, the signal is represented only by the a_h coefficients.

Fourier series analysis

$$x(t) = \frac{b_0}{2} + \sum_{h=1}^{\infty} (b_h \cos(2\pi h f_0 t)) + (a_h \sin(2\pi h f_0 t)) \tag{3.1}$$

As speech is a non-stationary spectro-temporally complex signal it cannot be accurately approximated by the single sinusoid and weighted harmonics employed in Fourier series analysis. Fortunately, Fourier theory can be used to extend Fourier series analysis to arbitrary (non-periodic) signals in a process called *Fourier analysis*. Equation 3.2 displays how the *continuous time Fourier transform* (CTFT) can be used to decompose any continuous time domain signal into an integral over all complex weighted frequencies in the entire spectrum of the signal. The two processes are similar in that the spectral content of the continuous time domain signal is determined by correlating the signal with linearly independent complex exponential eigenfunctions over all frequencies, ω. However, the arbitrary continuous signal can only be accurately represented by correlating over an infinite number of frequencies as opposed to only the fundamental frequency and harmonics.

Continuous-time Fourier transform

$$X(\mathrm{j}\omega) = \int_{-\infty}^{\infty} x(t) e^{-\mathrm{j}\omega t} \mathrm{d}t \qquad x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\mathrm{j}\omega) e^{\mathrm{j}\omega t} \mathrm{d}\omega \tag{3.2}$$

This course mainly deals with digital speech signals, meaning that the signals we will be working with are not continuous, but discretized and sampled in the time domain. It then becomes necessary to define the *discrete time Fourier transform* (DTFT) in Equation 3.3. The time domain signal now contains a discrete time index, $n \in \mathbb{Z}$, but is still represented by an integral over a a continuous frequency spectrum. However, the signal is now sampled and the *Nyquist theorem* limits the frequency content of the signal's spectrum to half the sampling frequency. In order to account for this, the limits of the DTFT frequency to time transform integral must then be adjusted to 0 to 2π. However, it is important to note that there are still an infinite number of values in this range, meaning that the discrete time domain signal, $x(n)$, is still represented by an infinite sum over a bounded spectrum.

Discrete-time Fourier transform

$$X(e^{\mathrm{j}\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-\mathrm{j}n\omega} \qquad x(n) = \frac{1}{2\pi} \int_{0}^{2\pi} X(e^{\mathrm{j}\omega}) e^{\mathrm{j}n\omega} \mathrm{d}\omega \tag{3.3}$$

Certain properties of both the CTFT and the DTFT become important when working mathematically between the time and frequency domains. These properties are quite simple to derive by using only the definition of the Fourier transform. Doing so helps to build a more intuitive notion of time and frequency domain representations.

One important property is the *linearity* of the Fourier transform operator. The Fourier transform is itself a linear operator because it is performed by using linear integration and summation operators. Therefore, the transform of a signal corresponding to the linear superposition of two weighted signals is equivalent to the superposition of the weighted Fourier transform of each signal. This is the basic requirement for defining a linear system.

$$ax(n) + by(n) \quad \circ\!\!-\!\!\bullet \quad aX_k + bY_k$$

Most practical applications of digital speech processing will involve real valued signals in the time domain, such as the set of real valued voltage fluctuations in the recording from a microphone. The nature of the Fourier transform requires that the spectra of such real valued signals are complex conjugate symmetric.

$$\text{real valued} \quad x(n) \quad \circ\!\!-\!\!\bullet \quad X(\mathrm{e}^{\mathrm{j}\Omega}) = X^*(\mathrm{e}^{-\mathrm{j}\Omega})$$

Correlating an even signal with a complex exponential is equivalent to correlating the signal with both an odd sine and an even cosine function. Because the cosine and sine functions correspond

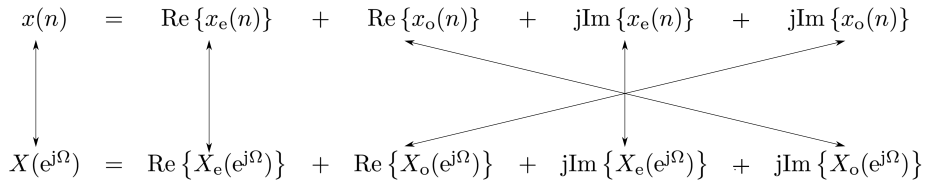


Figure 3.2: Symmetry relations of the Fourier transform.

to the real and imaginary parts of a complex number, the Fourier transform captures the even part of the time domain signal in the real part of the coefficients, whereas the odd part of the time domain signal is captured in the imaginary part of the Fourier coefficients. The complete set of these relations can be seen in Figure 3.2.

$$\begin{aligned} \text{even part} \quad & x_\mathrm{e}(n) = 0.5(x(n) + x(-n)) \quad \circ\!\!-\!\!\bullet \quad \mathrm{Re}\left\{X(\mathrm{e}^{\mathrm{j}\Omega})\right\} \\ \text{odd part} \quad & x_\mathrm{o}(n) = 0.5(x(n) - x(-n)) \quad \circ\!\!-\!\!\bullet \quad \mathrm{j\,Im}\left\{X(\mathrm{e}^{\mathrm{j}\Omega})\right\} \end{aligned}$$

A very important property that will have important applications in filtering is the equivalence of convolution in the time domain and multiplication in the frequency domain

$$x(n) * y(n) \quad \circ\!\!-\!\!\bullet \quad X(\mathrm{e}^{\mathrm{j}\Omega})\, Y(\mathrm{e}^{\mathrm{j}\Omega}).$$

Time shifts correspond to modulation of the spectrum in the Fourier domain and frequency shifts result in temporal domain modulations.

$$\begin{aligned} x(n - n_0) \quad & \circ\!\!-\!\!\bullet \quad \mathrm{e}^{-\mathrm{j}\Omega n_0} X(\mathrm{e}^{\mathrm{j}\Omega}) \\ x(n)\mathrm{e}^{\mathrm{j}\Omega_\mathrm{M} n} \quad & \circ\!\!-\!\!\bullet \quad X(\mathrm{e}^{\mathrm{j}(\Omega - \Omega_\mathrm{M})}) \end{aligned}$$

Parseval's theorem states that the energy of a signal is preserved upon taking the Fourier transform. This property stems from the fact that the transform is perfectly invertible, meaning that no information is lost upon transformation between time and frequency domains.

$$\sum_{n=-\infty}^{\infty} x(n) y^*(n) \quad = \quad \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\mathrm{e}^{\mathrm{j}\Omega}) Y^*(\mathrm{e}^{\mathrm{j}\Omega}) \mathrm{d}\Omega$$

The properties defined above can be used to prove that discrete, time domain signals correspond to periodic spectra, and conversely, periodic time domain signals correspond to discrete spectra. Four relations can be intuitively derived from this statement that generalize time-frequency domain relations regarding the continuity and periodicity of the function to be transformed. These can be seen in Figure 3.3.

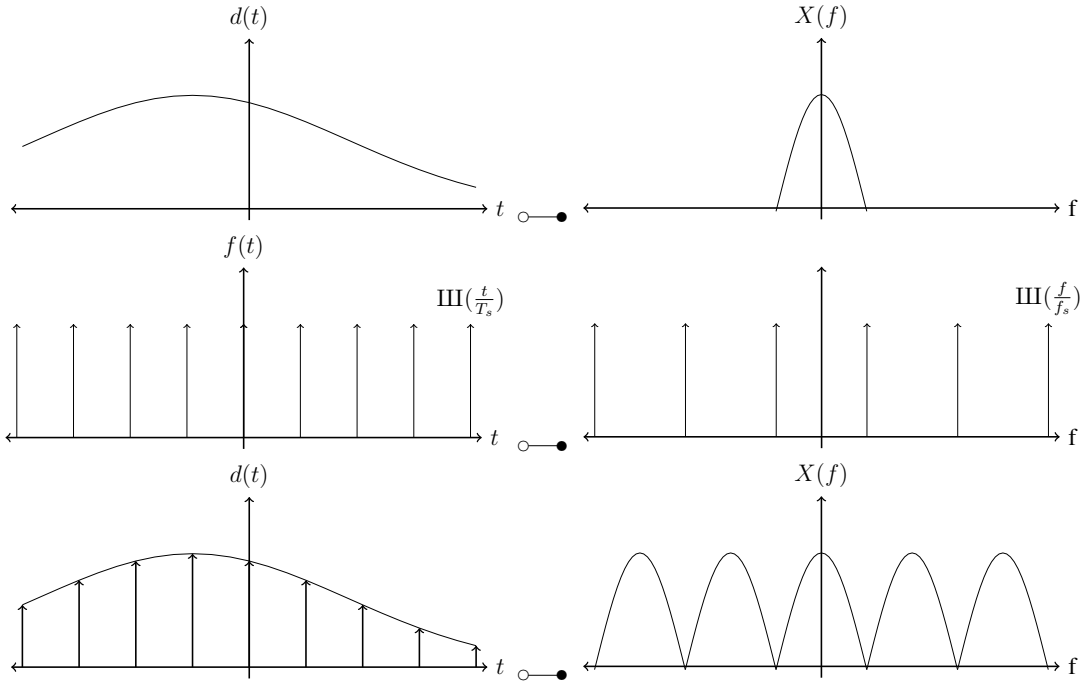


Figure 3.3: Periodicity and continuity relations of the discrete time Fourier transform.

continuous/non-periodic	○—●	continuous/non-periodic
continuous/periodic	○—●	discrete/non-periodic
discrete/periodic	○—●	discrete/periodic
discrete/non-periodic	○—●	continuous/periodic

3.2 Digitization of speech signals

Modern data storage and transmission techniques require that we deal predominately with digital speech and audio signals. In order to digitize the analog signal recorded from our microphone, it must be *quantized* in the amplitude domain and *sampled* in the temporal domain. Only then can it be stored on our hard drives or transmitted from our mobile phones. Section 6.1 deals with the quantization process, and the current section will introduce the concept of sampling and its effects upon the frequency content of a signal.

The sampling process is a discretization of the time axis that results in snapshots of the analog signal at uniform instances in time. As can be seen in Figure 3.4, this process implies that there can be no knowledge of the signal between two successive samples. However, if we have a priori knowledge about the smoothness of the signal, i.e. how fast the signal changes accross time, it may be possible to reconstruct the signal between to samples by interpolation. The amount of smoothness is reflected in the highest frequency content of the signal at hand. Signals containing higher frequencies can change faster over time, whereas signals containing lower frequencies will appear smoother in the time domain. The *sampling theorem* states that the signal can be *perfectly* reconstructed when the sampling frequency is twice that of the highest frequency in the signal. This implies that it is often advantageous to use the lowest possible sampling frequency as anything higher would be redundant. In these cases, it is also helpful to have a priori knowledge of the signal so that the frequency content can be modified in an effort to control the amount of information in the sampled signal. In practice, the signal is often low-pass filtered and then sampled.

Discretization can be formally defined as the multiplication of a time domain signal, $x(t)$, with an impulse train or a delta comb, $\text{Ш}(\frac{t}{T_s})$, spaced at the sampling period, T_s. This is shown in Figure 3.4. The equivalence between multiplication and convolution between time and frequency domains can be used to analyze the frequency domain effects of sampling. This will help to derive the necessary conditions that must be fulfilled in order to perfectly reconstruct the signal.

$$d(t) = x(t) \cdot \text{Ш}(\frac{t}{T_s}) \quad T_s = \frac{1}{f_s} \text{ the sampling period}$$

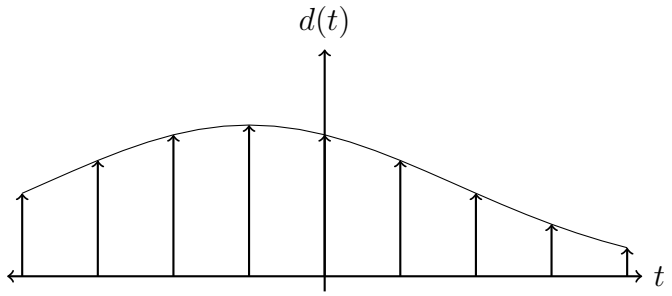


Figure 3.4: An arbitrary continuous function sampled by multiplication with a delta comb.

The delta comb in Figure 3.5 is formally defined as an infinte summation of delta pulses spaced at T_s.

$$\text{Ш}(\frac{t}{T_s}) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s)$$

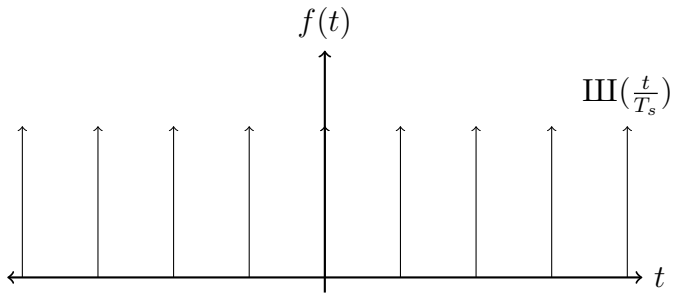


Figure 3.5: Delta comb used to sample a continuous function.

The spectrum is then computed by performing the Fourier transform of the delta comb.

$$\mathfrak{F}(\text{Ш}(\frac{t}{T_s})) = \int_{-\infty}^{\infty} \sum_{n} \delta(t - nT_s) e^{-jwt} dt$$

This integral is solved by use of the sifting property which states that integrating the product of a function, $f(t)$, and shifted impulses over all t, is equivalent to evaluating the function at the point T.

$$\boxed{\int f(t)\delta(t - T)dt = f(T) \quad \text{"Sifting Property"}}$$

The sum and integral operators can be exchanged as they are both linear operators and then the sifting property can be employed to yield a sum over infinitely many complex exponential functions.

$$\text{Ш}(\frac{t}{T_s}) = \sum_{n=-\infty}^{\infty} e^{-j\omega n T_s}$$

If $\omega T_s = 0$ or integer multiples of 2π, the complex exponential evaluates to one and the summation goes to infinity. The infinite summation of any other value of ωT_s results in zero. In phasor representation, this would be equivalent to summing symmetric phasors around the real and imaginary axis. An infinite number would cancel each other out and ultimately sum to 0:

$$\text{Ш}(\frac{t}{T_s}) = \begin{cases} \infty, & wT_s = k2\pi, k \in \mathbb{Z}. \\ 0, & \text{else} \end{cases}$$

The result is a function in the spectral domain that is infinite at some points and zero for all other values. This function definition should look familiar because it a delta comb similar to that with which we started. However, instead of being a series of pulses spaced at the sampling period, T_0, the spectrum is a pulse train spaced by the sampling frequency, f_s. This can be seen in Figure 3.6.

$$= \sum_{k=-\infty}^{\infty} \delta(w - k2\pi f_s)$$

$$= \text{Ш}(\frac{f}{f_s})$$

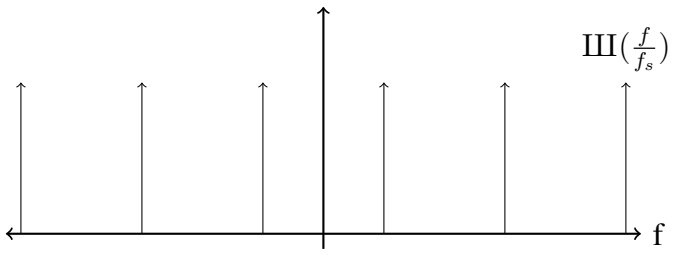


Figure 3.6: The Fourier transform of a delta comb is also a delta comb.

It was introduced in Section 3.1 that discrete time domain signals correspond to periodic frequency domain signals. The previous computation has not only proved this by analyzing the frequency domain behavior of the sampling process, but the distances between these periodic repetitions is now known. This distance was shown to be exactly the sampling frequency, f_s. If the time domain sampling is modeled as the multiplication of a continuous signal with a delta comb spaced at the sampling period, then sampling in the frequency domain corresponds to a convolution of the continuous frequency spectrum and a delta comb spaced at the sampling frequency.

$$D(f) = X(f) * \text{Ш}(\frac{f}{f_s})$$

We assume that the Fourier transform of a continuous time domain signal, $x(t)$, is displayed in Figure 3.7.

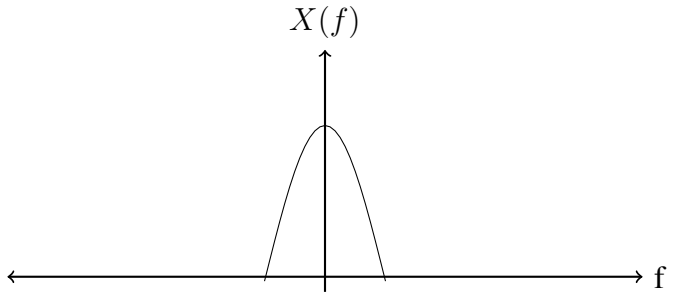


Figure 3.7: Frequency domain representation of a continuous time domain signal.

Figure 3.8 depicts the sampling process in the frequency domain as a convolution of the continuous spectrum and a delta comb spaced at the sampling frequency.

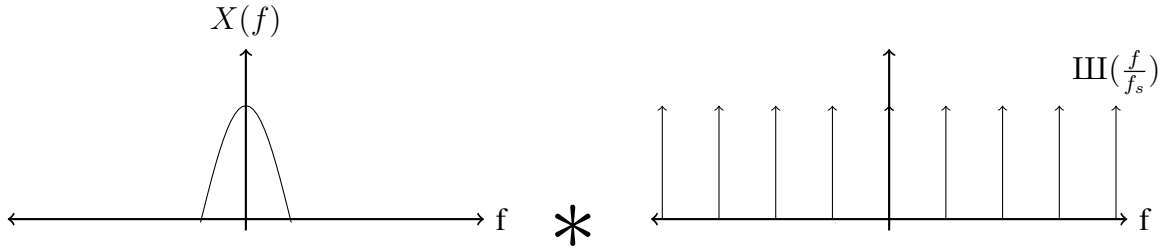


Figure 3.8: Convolution of continuous frequency domain signal with frequency domain delta comb.

This convolution forces the spectrum to be periodic. The *sampling theorem* can now be visually derived from Figure 3.9. If the sampling frequency is perfect, low pass filtering can be utilized to eliminate the replicas, thus allowing for a perfect reconstruction when the signal is inverse transformed back to the time domain. It is interesting to note that from a time-domain perspective, this low-pass filtering corresponds to an interpolation of the signal between two samples. When the chosen sampling frequency is too low and the signal is not sampled often enough, the replicas of the spectra will then overlap allowing for energy from lower frequencies to bleed into high frequency bands. This effect is called *aliasing* and results in distortions in the inverse transformed time domain signal.

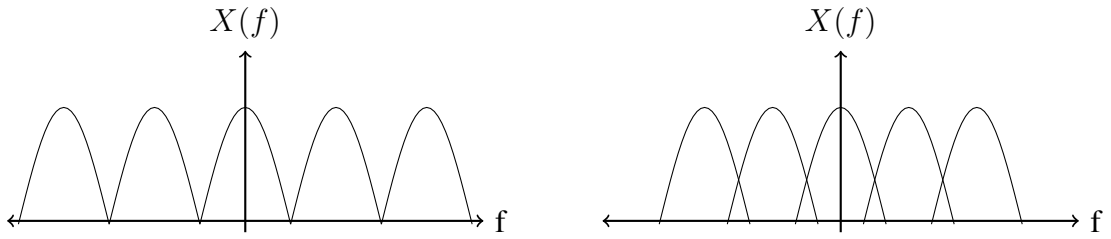


Figure 3.9: Left: Spectrum of a sampled time domain signal when the perfect sampling frequency is chosen. Low pass filtering would remove replicas of the spectrum and allow for perfect reconstruction. Right: Aliasing as a result of choosing to low of a sampling frequency.

The Sampling Theorem

If a continuous time signal is band-limited and contains frequency components up to frequency f_{max}, then the signal can be correctly represented and uniquely reconstructed from a set of equally spaced samples if the sampling frequency is at least $2f_{max}$.

Telephony applications, such as ISDN or GSM, implement sampling rates of about $8\,\mathrm{kHz}$. The sampling theorem thus limits any information above $4\,\mathrm{kHz}$, therefore requiring a low pass filtering of the signal before sampling. In practice, a finite-length filter cannot be realized that completely attenuates all frequencies above or below a certain frequency bin. Because the steepness of the low-pass filter is limited, the spectrum will be distorted from a certain value up to the sampling frequency. To account for the limited steepness of practical low-pass filtering, in telephony, speech signals that are sampled at $8\,\mathrm{kHz}$ are low-pass filtered at a cut-off frequency at $3.4\,\mathrm{kHz}$. However, telephone speech is still quite understandable as mostly only unvoiced speech sounds, such as plosives, are effected within this frequency range. The formant transitions that are crucial to speech understanding mainly lie below $3.4\,\mathrm{kHz}$. An example for an exception are the phonemes [f] and [s]. You can check this, by trying to distinguish "fox" from "socks" when calling a friend. At present, the coding strategies implemented in voice over IP (VoIP) clients and modern smartphones (HDvoice) do use higher sampling frequencies (e.g. $16\,\mathrm{kHz}$) that allow for wide-band speech telephony and even higher (e.g. ultra-wideband at $32\,\mathrm{kHz}$ sampling rate). While wideband speech coders have been around for quite some time and the required data-rate is low enough to enable transmission in GSM networks, wideband telephony on mobile phones in Germany has only been available since November 2011 after rebranding wideband speech coders to HD-voice.

The unique frequency spectra of the numerous instruments used to create music result in an even wider range of audio bandwidth. Higher sampling rates are therefore required to capture the higher frequency instruments. Hi-fidelity (HiFi) audio applications utilize $44.1\,\mathrm{kHz}$ sampling rates and some standards even go as high as $48\,\mathrm{kHz}$. However, the limit of human perception is in the range of $20\,\mathrm{kHz}$ and sampling rates above $48\,\mathrm{kHz}$ are usually not necessary when aiming at human listeners.

3.3 Discrete Fourier Transform (DFT)

The discrete Fourier transform (DFT)

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{j\frac{2\pi}{N}kn}, n = 0,\ldots,N-1. \qquad X_k = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}, k = 0,\ldots,N-1$$

The discrete time Fourier transform (DTFT) allowed for the spectral transformation of a sampled signal, however the transform is still performed over a bounded range of continuous frequencies. In addition, the time domain signal is also defined over an infinite range, thus making the DTFT impossible to realize in practice. The time domain signal must therefore be *windowed* meaning that the transform is computed on a finite time sequence that is extracted from the signal. The simplest example of windowing is the multiplication of the signal with a rectangular function. This is displayed in Figure 3.10.

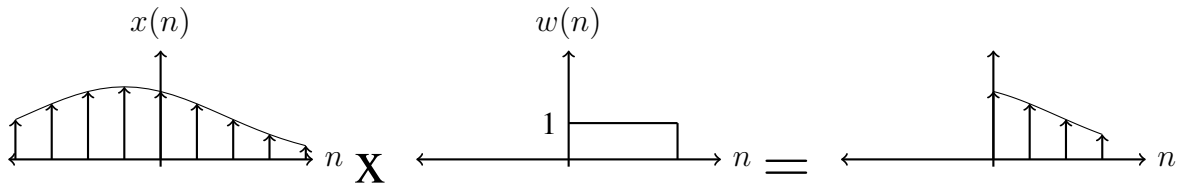


Figure 3.10: Rectangular windowing of a sampled continuous function.

The DTFT of the discrete, non-periodic windowed signal is the continuous, periodic frequency spectrum seen in Figure 3.11. The Fourier transform of a rectangular function is a Sinc-function $(\sin(x)/x)$, meaning that a time domain windowing is equivalent to a spectral domain convolution with a Sinc-function. This convolution smears out the frequency content of the signal and results in a decrease in frequency resolution due to distortions in the spectral domain representation of the signal. This is an unavoidable consequence of the windowing process that results in a trade-off between time and frequency domain resolution. If the rectangular window is made longer, there is a loss in temporal resolution, however the Sinc-function becomes more narrow, resulting in a higher frequency resolution. The opposite case is also true, i.e. shorter windows result in a higher temporal resolution but a reduced spectral resolution.

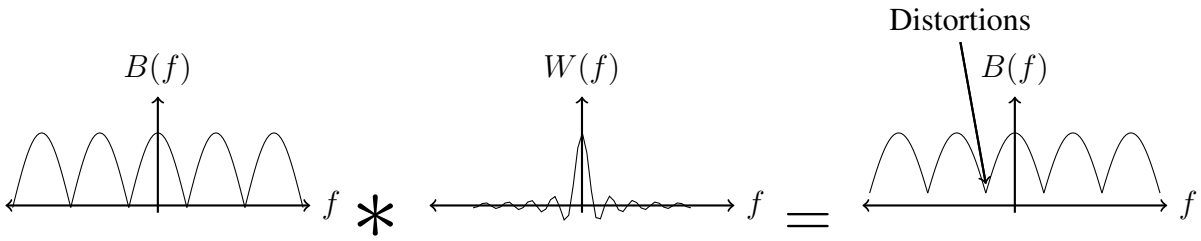


Figure 3.11: Frequency domain effects of time domain windowing.

The continuous frequency spectrum must now be sampled so that it can be realized using digital methods. This is performed in Figure 3.12 by multiplying the spectrum with a pulse train.

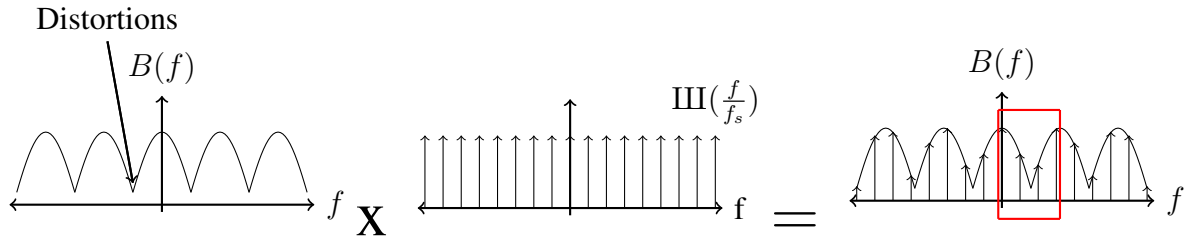


Figure 3.12: Sampling the windowed time domain signal in the frequency domain.

This frequency domain sampling results in the convolution of the time domain signal with a pulse train, and Figure 3.13 shows that this convolution results in a periodic repetition of the windowed segment.

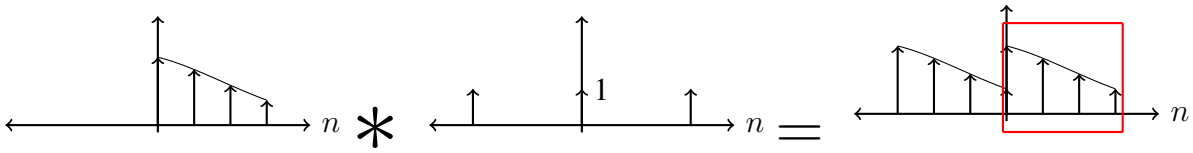


Figure 3.13: Time domain replicas that result from frequency domain sampling

Figure 3.13 also shows that there is a potential problem with the overlapping of the time domain windows when the chosen window length and sampling frequency are not carefully selected. This would result in signal distortions called *time domain aliasing*. To avoid this, the time domain pulse train must have a period length that is larger or equal to the window length in seconds, $\tau < \frac{N}{f_s}$.

$$\sum_{n=-\infty}^{\infty} \delta(t - n\tau), \tau < \frac{N}{f_s}$$

The Fourier transform of this function would be another pulse train that is then multiplied by the frequency domain spectrum for the purpose of sampling.

$$\mathfrak{F}\Big(\sum_{n=-\infty}^{\infty}\delta(t-n\tau)\Big)=\sum_{k=-\infty}^{\infty}\delta(f-k\frac{1}{\tau})=\sum_{k=-\infty}^{\infty}\delta(f-k\frac{f_s}{N})$$

To avoid time domain aliasing, the continuous frequency spectrum can only be sampled by multiplication with a pulse train of maximum width $\frac{1}{\tau}=\frac{f_s}{N}$. These sample points are the frequency bins that determine the spectral resolution of the DFT. We can now see this inverse relationship between resolution in the temporal and frequency domains emerging. If the spectral resolution is $\Omega=\frac{f_s}{N}$, and the window length N is increased, the result is a finer frequency resolution, but this is at the expense of a longer temporal window. However, this longer temporal window means that we have lost the ability to track changes in the temporal domain and have therefore reduced the temporal resolution. This window can then be shortened, but not without increasing the bin width in the frequency domain. It is also important to note that this computation develops the claim made in Section 3.1 where it was introduced that a discretized signal in time domain corresponds to a periodic spectrum, and that a discrete frequency domain signal corresponds to a periodic time domain signal.

Spectral Resolution

When the signal samples $x[n]$ are generated by sampling a continuous signal $x(t)$ with sampling rate f_s, the center frequencies of the DFT bins are $\Omega_k=\frac{2\pi k}{N}$, or $f_k=\frac{f_s}{N}k$.

It is also possible to artificially increase the length of the window by appending zeros to the windowed signal, referred to as *zeropadding*. The DFT is simply the discrete, sampled frequency spectrum of the continuous DTFT spectrum. When the window length, N, is increased, the DTFT is sampled more densely (Figure 3.14). However, it is important to understand that we are not adding any new information by adding zeros, and technically speaking, the spectral resolution is not being increased. Instead, spectral bins are merely interpolated. In fact, the DTFT of a finite data sample can be mathematically approximated by a DFT with a large amount of zeropadding.

Many practical applications of digital speech processing involve some computational processing in the spectral domain, thus resulting in the use of windowed time domain signals and sampled frequency spectra. The DFT will therefore be the most frequently implemented spectral transformation meaning that it is important to analyze to what extent the windowing process distorts the signal.

As an example, two full cycles of a single sinusoid are windowed in Figure 3.15. The DFT of the windowed signal contains two distinct mirrored peaks at frequency bins corresponding to the signal frequency. Because the window is aligned perfectly with full cycles of the sinusoid, the zeros of the Sinc-function side lobes are coincident with the frequency domain samples and there is no spectral leakage. The resulting spectrum contains peaks that perfectly resolve the frequencies present in the signal. However, this is a very rare occurrence and we will mostly see situations in which the analysis window does not perfectly frame whole number cycles of a sinusoid. The side lobes of the convolved Sinc-function now introduce energy from other frequency bands in a process termed *spectral leakage*. The result of spectral leakage can greatly reduce the ability to resolve closely spaced sinusoids as can be seen in Figure 3.15. This effect of

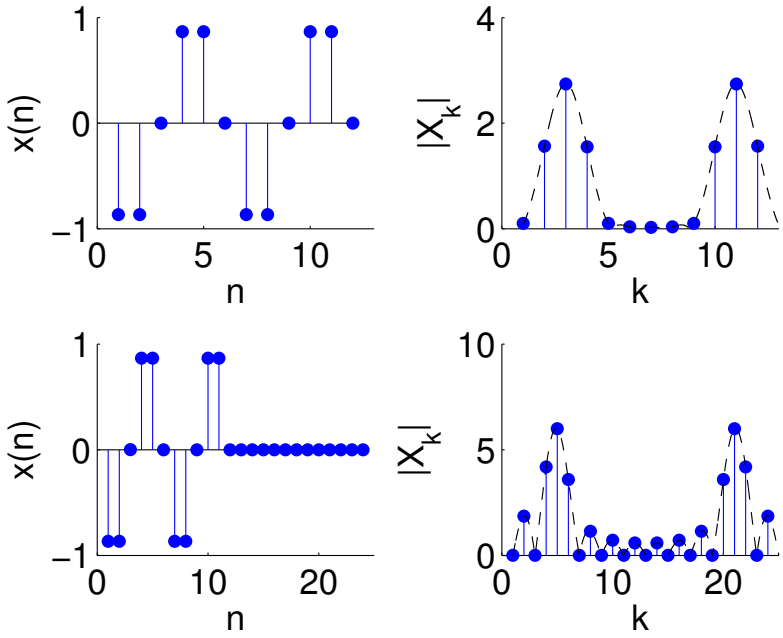


Figure 3.14: (Top) Temporal and spectral representations of a sampled pure tone. (Bottom) Effects of zero padding the same tone allow for the resolution of peaks although no new information is added.

windowing can have adverse effects on spectra, thus requiring special attention when interpreting the results of spectral transformations.

These adverse effects can be greatly reduced by the application of tapered analysis windows, such as the Hann or Hamming windows. Figure 3.16 shows several examples of time domain windowing functions and their corresponding frequency domain representations. The frequency response of each window function reveals main lobes and side lobes that vary in width and height, respectively. In comparison to the rectangular window function, the tapered analysis windows offer greater attenuation of the side lobes, a quality that helps to reduce spectral leakage effects. However, the attenuation of side lobes is achieved at the expense of an increased main lobe width. As is often the case in frequency domain representations, there is a trade-off when choosing a particular analysis window for a given application. One can have either high frequency resolution at the expense of spectral leakage or vice versa. It is then important to determine the requirements of the given application in order to choose the correct analysis window.

The same two sinusoids from Figure 3.15 are now analyzed using a Hann window as opposed to a rectangular window, and the corresponding frequency domain representations are shown in Figure 3.17. Even though the window length is an integer number of sinusoid periods, the spectrum still contains energy in the two samples adjacent to the main peak as opposed to the two distinct peaks produced by the rectangular window. This demonstrates the effect of a reduced frequency resolution at the price for an increased side-band attenuation. The benefit of using a tapered analysis windows becomes clear when comparing the frequency response of the imperfect window length cases in Figures 3.15 and 3.17. The balance between frequency resolvability and resolution that is provided by the use of the Hann window is generally preferred over rectangular windowing.

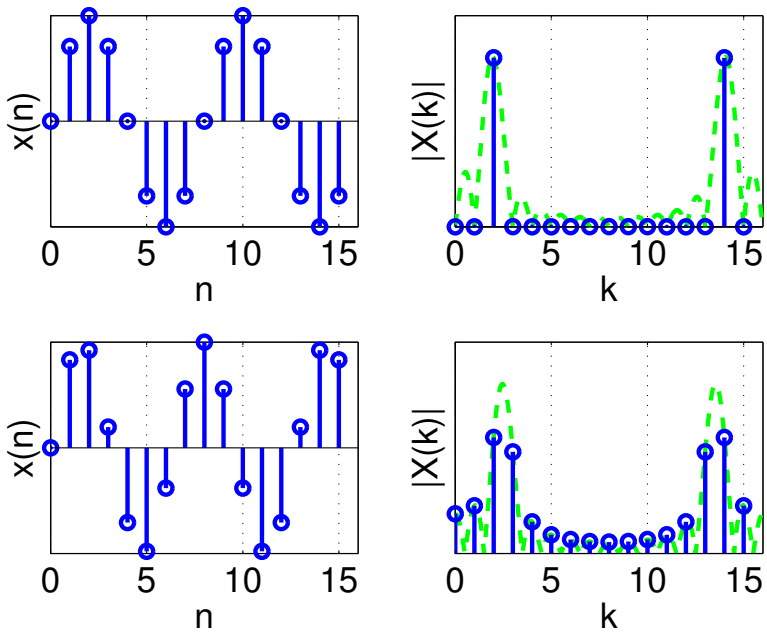


Figure 3.15: (Top) Sampled pure tone sinusoid and corresponding spectrum when windowed at the beginning and end of 2 cycles of the sinusoid. (Bottom) Rectangular windowed sinusoid and corresponding spectrum when sinusoid is shifted.

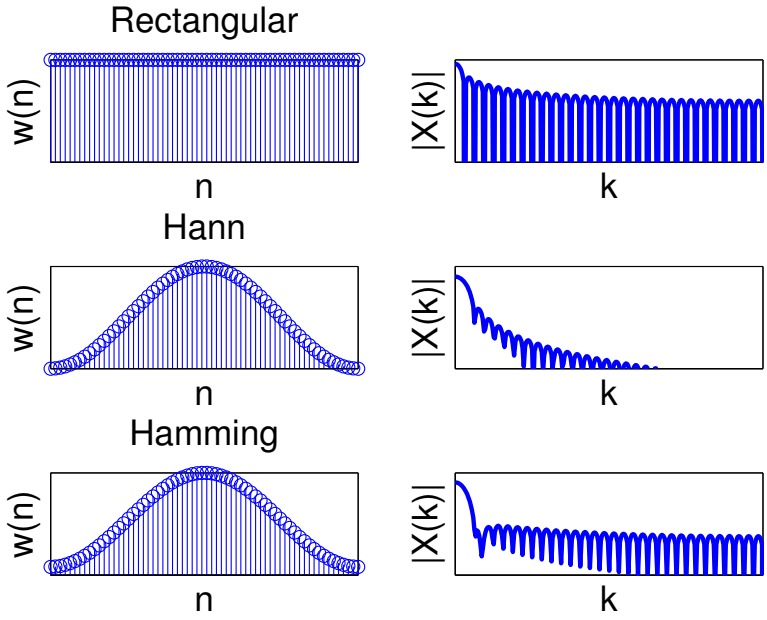


Figure 3.16: (Top) Rectangular window and resulting spectrum. (Middle) Hann window and resulting spectrum. (Bottom) Hamming window and resulting spectrum. Note that spectrum of windowed signal will be convolved with spectrum of the window.

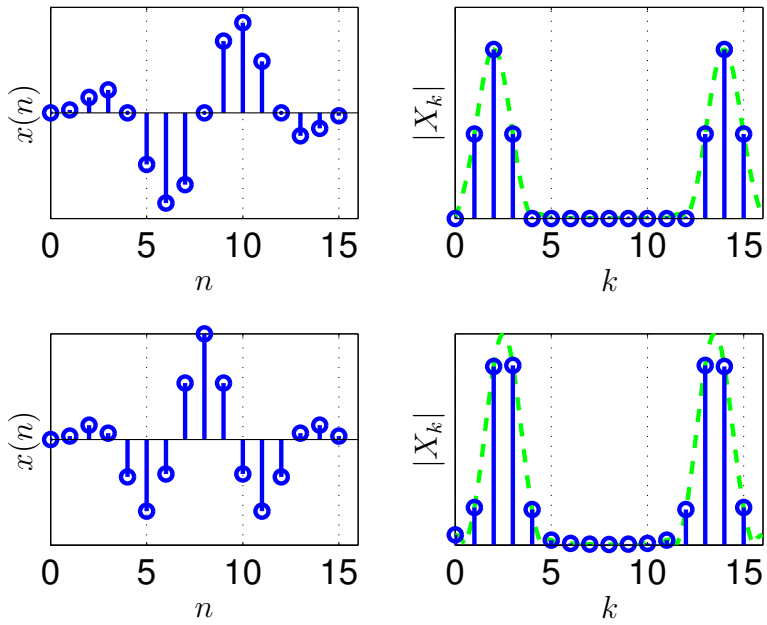


Figure 3.17: (Top) Hann windowed sinusoid and corresponding spectrum with window length equal to 2 cycles of the sinusoid. (Bottom) Hann windowed sinusoid and corresponding spectrum with window length not equal to a whole number of sinusoidal cycles.

3.4 Short-Time Fourier Transform (STFT)

3.4.1 Spectrogram (narrow-band vs. wide-band)

Because speech is a time varying signal, it is mostly not desirable to compute and work with the DFT of an entire speech signal as this fails to capture how the frequency content changes over time. The signal is therefore be split into short segments which are then be transformed to frequency individually. This is accomplished by means of the *short time Fourier transform*, (STFT), probably one of the most frequently used tools in audio processing to analyze spectral content. All of the properties that were derived for the DFT will still hold for local STFT frames. In addition, there will also be a new set of parameters that must be chosen for the computation of the STFT. These additional parameters depend upon the information that we are looking to extract from the transformation.

Figure 3.18 shows a chirp, a slower time varying tone of a single decreasing frequency, with a splash, a rapid burst of sound containing all frequencies. These two signals behave very differently over time and will also be perceived differently, however, perhaps surprisingly, the DFT

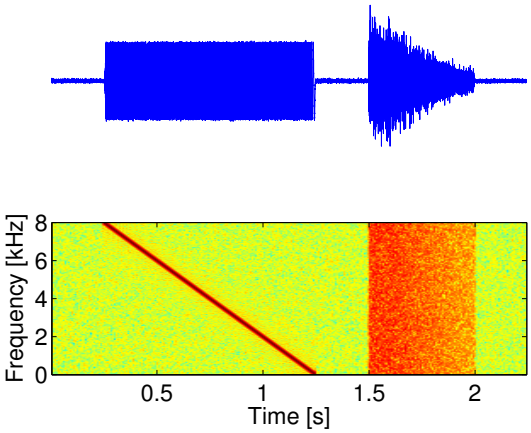


Figure 3.18: (Top) Time domain signal of a frequency chirp followed by a frequency splash. (Bottom) Short time Fourier transform of the splash and chirp. Because the spectrum of the two signals would be a mean over time, the spectra would look very similar. The STFT provides a frequency domain representation in which the two signals can be distinguished.

magnitude spectrum of each signal would look the same: the splash and the chirp both contain energy in all frequency bands, therefore the mean over time would result in two very similar spectra, and it would be almost impossible to distinguish the two signals. Therefore the STFT is used to produce the *spectrogram* on the bottom of Figure 3.18. In this visualization, the frequency content of both signals can be viewed in addition to the time varying behavior. We now have a three dimensional representation that represents the time segment index on the x-axis, frequency bins on the y-axis, and the magnitude of the time-frequency energy coded in color.

The STFT is computed by using a sliding window transform. A long speech signal is divided into segments upon which the DFT is performed. However, simply performing the DFT on each segment would correspond to using a rectangular analysis window. Using the same tricks that were introduced with the DFT in Section 3.3, a Hann or Hamming window is usually used to improve the spectral content or to reduce spectral leakage. This is represented as $w[n]$ in the formal definition of the STFT in Equation (3.4).

The short time Fourier transform (STFT)

$$X[k,I] = \sum_{n=0}^{N-1} w[n]x[n+IL]\mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{N}}, \quad k = 0,\ldots,N-1 \tag{3.4}$$

I = frame index, n = local time index,
N = window length, L = frame shift,
$N-L$ = overlap

In Equation (3.4), the new time index I corresponds to a windowed signal segment of length N at time IL. A DFT is performed on each frame to produce a time varying even, complex valued spectrum of length N. The frequency corresponding to a frequency index k can be calculated by using the sampling frequency, $f_k = \frac{k}{N} f_s$. Of course, by Nyquist, the highest frequency that can be represented in the spectrum is $\frac{f_s}{2}$ so half of the symmetric spectrum is redundant

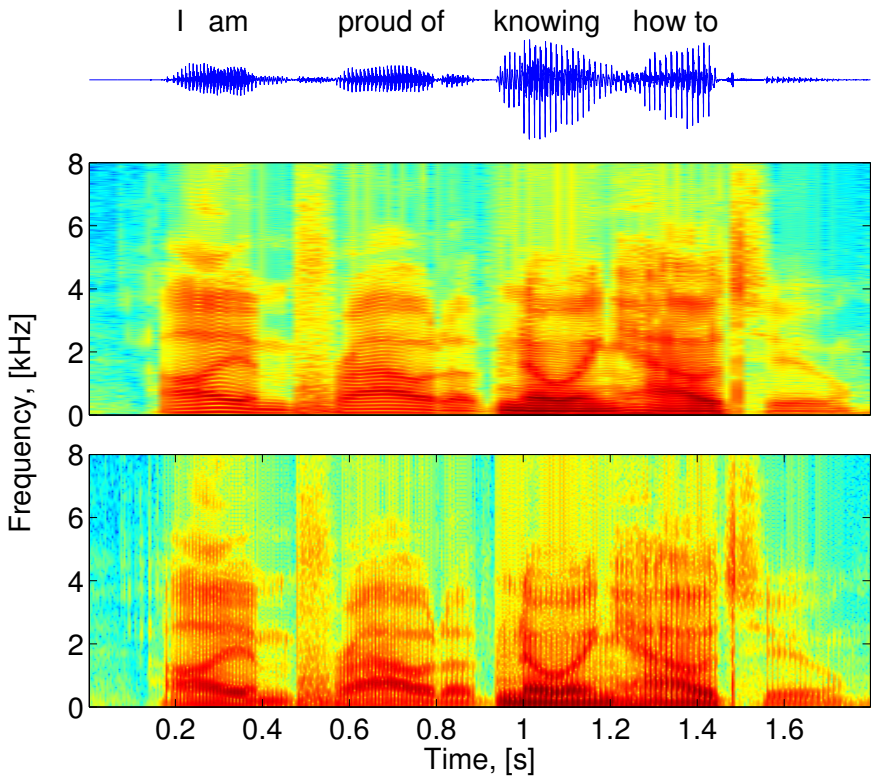


Figure 3.19: (Top) Narrowband spectrogram of speech that trades poor temporal resolution for a higher frequency resolution (horizontal lines). (Bottom) Wideband spectrogram of speech that trades poor frequency resolution for a higher temporal resolution (vertical lines).

and typically not used for visualization purposes. The frequency resolution is a function of the window length and is represented by: $\Omega_k = \frac{k}{N} 2\pi$. The results of the STFT are typically visualized as *spectrograms* such as those displayed in Figure 3.19. The relative energies of each time (x-axis) and frequency (y-axis) bin are generally converted to a logarithmic scale before plotting and then distinguished by a color scale.

Changing the different parameters involved in computing the STFT allows for the analysis of different properties of the signal. For example, the narrow and wide band spectrograms in Figure 3.19 are generated by the different window lengths $N = f_s \cdot 32\,\mathrm{ms}$ and $N = f_s \cdot 8\,\mathrm{ms}$, respectively. The sampling frequency of the processed speech signal is $f_s = 16\,\mathrm{kHz}$ and, by Nyquist, this implies that the bandwidth of the signal is $\frac{f_s}{2} = 8\,\mathrm{kHz}$. The shorter $8\,\mathrm{ms}$ window of the broad band spectrum, results in the frequency information being updated more often. This allows for a better tracking of formant transitions and the resolution of the fundamental period in the voiced speech segments (represented by vertical lines within the spectrogram of Figure 3.19). However, it was shown that the frequency resolution is calculated by $\Omega = \frac{2\pi}{N} = 125\,\mathrm{Hz}$, so it is more difficult to resolve the spectral harmonics and closely spaced formants. The effects of a longer window length can be seen in the narrowband spectrogram in Figure 3.19. The window length of $N = f_s \cdot 32\,\mathrm{ms}$ corresponds to a higher frequency resolution of $\Omega = \frac{2\pi}{N} = 31.25\,\mathrm{Hz}$ for which individual spectral harmonics of the fundamental frequency (horizontal lines) can be resolved in the spectrogram. However, unvoiced speech sounds, such as plosives, are typically on the order of $40 - 60\,\mathrm{ms}$ so this typical narrowband time window makes it difficult to observe their

dynamics.

The narrow and wide band spectrograms reveal the fundamental relationship between the time and frequency domains. As temporal resolution is increased, there is a reciprocal decrease in frequency resolution forcing us to choose between high temporal resolution and poor frequency resolution using short windows or bad temporal resolution and high frequency resolution using long windows. This resolution problem is limiting, therefore window lengths must be chosen dependent upon application. If we would like to analyze a quickly changing signal such as a plosive, then it would be nice to use short windows. However, longer windows can be used with voiced sounds such as the vowel [a] which is comparatively stationary.

A typical narrow band analysis setup uses overlapping windows on the order of $32\,\mathrm{ms}$ with frame shifts between 50 and 75 percent. Acoustic sounds in nature often follow a 6dB attenuation per octave, meaning that there is usually an attenuation towards higher frequencies. Therefore, a high pass pre-emphasis filter is sometimes employed on each frame to boost the higher frequencies. Because the windowing process will most often fail to capture whole numbers of the inherent periodicities, filtered frames are then multiplied by an application dependent analysis window as introduced in Section 3.3. An STFT is then computed on the processed temporal signal to produce the final spectrogram.

3.5 Spectral Envelope

The source filter model of speech production was introduced in Section 1.3. It was shown that speech production can be modeled as the filtering of an excitation signal (produced by the lungs and glottis) by a vocal tract transfer function that represents the current geometry of the vocal tract. It was also shown that this time domain filtering corresponds to a frequency domain multiplication of the two modeled signals. The process of extracting the vocal tract transfer function in the frequency domain is referred to as extracting the *spectral envelope* of the signal.

Naturally flowing speech is characterized by rapid spectrotemporal fluctuations. The fundamental frequency corresponds to prosody and speaker identification. More importantly, the formants of the vocal tract transfer function correspond to the phonemes and therefore the intended meaning of the speech signal. In order for the source filter model to accurately reproduce speech, it is essential to capture these time varying elements. By assuming that these remain stationary within a certain time window, the STFT can be employed so that these parameters can be computed for each frame.

Figure 3.20 depicts this process for a small section of the word ”proud”, [ˈpraud]. The time varying signal is shown for three $32\,\mathrm{ms}$ consecutive windows. The choice of window length has allowed for the resolution of the fundamental frequency and its harmonics in the spectra of the windowed signals. To separate this excitation signal from the vocal tract transfer function for the speech production model, the spectral envelope is computed by taking the wide band spectrogram of the signal. The decreased spectral resolution cannot resolve between the harmonics and results in a signal that reveals the formant frequencies that reflect the effects of the vocal tract. Although the three spectra look similar, their subtle differences represent the transition between phonemes that are essential for the reproduction of intelligible speech with our model.

Whether or not a wideband or a narrowband spectral analysis is chosen depends on practical considerations. If we are only interested in analyzing spectral envelopes a wideband analysis is a good choice, e.g. for speech coding. If we are interested in resolving the spectral harmonics, a narrowband analysis is preferred, e.g. in speech enhancement to reduce noise between spectral harmonics. Furthermore, it should be noted that long spectral analysis and synthesis windows also increase the algorithmic latency. Thus, for latency critical applications often a wideband analysis is employed.

Instead of simply taking a wideband spectral analysis, more sophisticated methods for obtaining spectral envelopes are LPC analysis and cepstral analysis as will be explained in the forthcoming chapters.

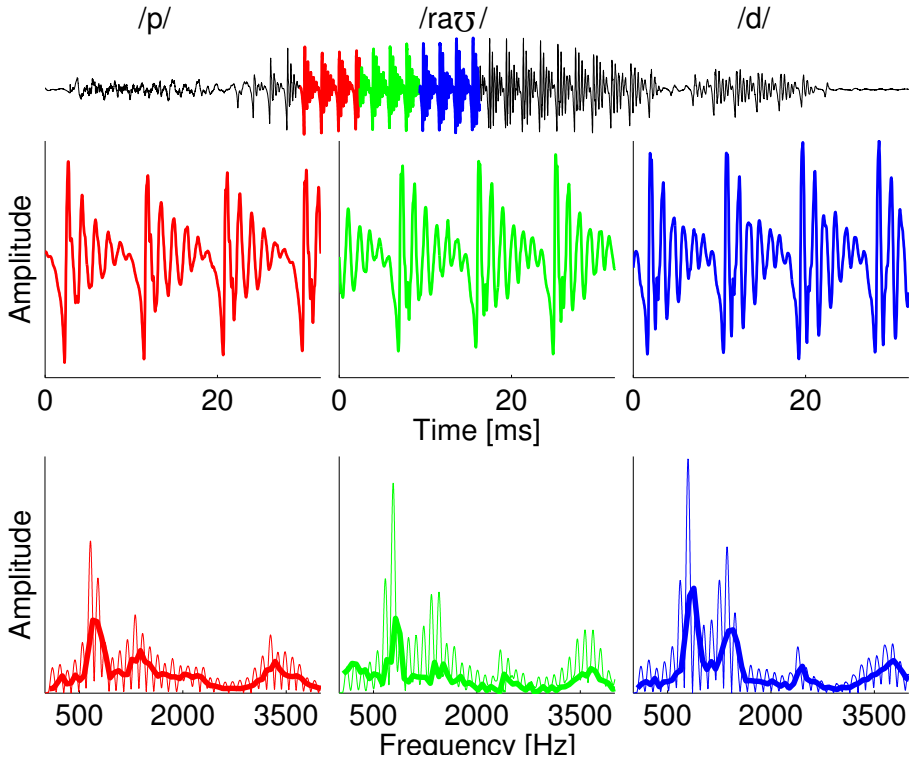


Figure 3.20: Narrow band spectra (bottom) of three consecutive windowed speech sounds taken from a speech signal (top). The colored lines superimposed over the spectra represent the spectral envelopes.

3.6 Synthesis: Overlap-add technique

The benefits of transforming speech signals into the frequency domain is that it is more efficient to perform certain modifications and that certain signal properties are more easily accessible. For example, a noise reduction process can be applied to remove undesired background noise from a speech signal. After these modifications are performed, the time domain signal must then be re-synthesized because a loudspeaker is only capable of reproducing time domain signals. To accomplish this, the set of processed analysis windows must be strung together in a manner so that the listener cannot perceive any discontinuities.

$$x[n] \xrightarrow{windowing} w[n]x[n+IL] \xrightarrow{DFT} X[k,I] \xrightarrow{modification} Y[k,I]] \xrightarrow{IDFT} y[n]$$

The windowing process introduces an inherent time delay into the output signal so that a perfectly reconstructed signal is delayed by one window length (plus processing time).

$$y[n] = x[n - \Delta].$$

As was shown previously, overlapping analysis windows are applied to the original speech signal segments. The DFT is then performed on each segment to transform it into the frequency domain so that some modification step can then be performed. If there is no modification, then we are simply windowing our signal and transforming it back into the time domain by using the *inverse discrete Fourier transform*, IDFT.

$$X[k,I] = Y[k,I] \xrightarrow{IDFT} y[n] = w[n]x[n+IL]$$

Substituting $n' = [n + IL]$, this equation can be rewritten as

$$\begin{aligned} y[n] &= w[n]x[n+IL] \\ y[n'-IL] &= w[n'-IL]x[n'] \end{aligned}$$

Remembering that these windowed segments were taken as overlapping frames from the original signal, we synthesize the time domain signal by aligning the frames accordingly and adding them together. A problem now arises because the overlapping sections of the synthesized signal would contain more energy than the non-overlapping sections. We therefore do not have perfect reconstruction and the synthesized signal would contain artifacts in the form of modulations corresponding to the frame shift. The frame shift cannot be simply discarded as the transitions between adjacent frames become less smooth. It can be seen that the conditions for perfect reconstruction require that the sum of the energy in the overlapping signal segments equals that

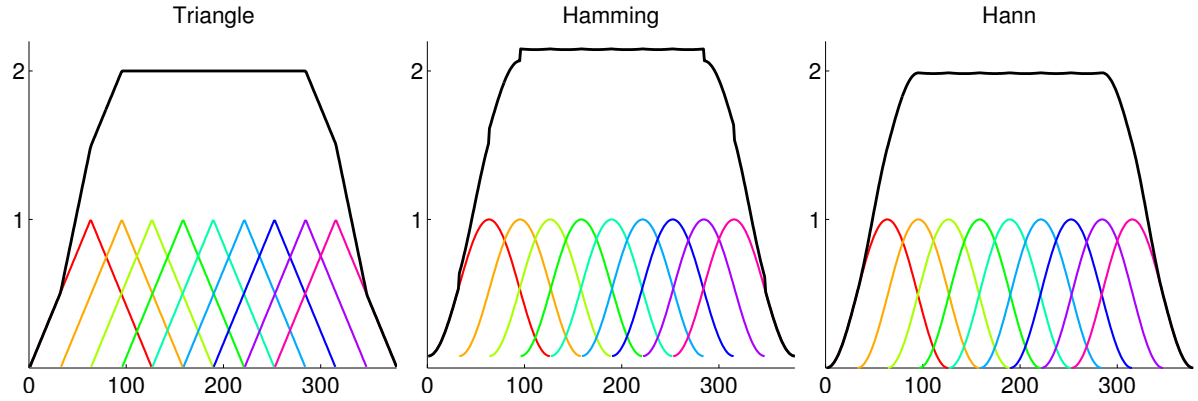


Figure 3.21: Addition of the weights imposed by three different windowing functions. The overlap for each is set properly so that they sum to a constant value thus allowing for less artifacts in the temporal reconstruction.

of the energy in the non-overlapping sections. This is can be performed by creating a synthesis window v.

$$y[n'] = \sum_I v[n' - IL] y_I[n' - IL] = x[n'] \sum_I v[n' - IL] w[n' - IL]$$

It can now be seen that for perfect reconstruction, analysis and synthesis windows must be chosen such that

$$\sum_I v[n' - IL] w[n' - IL] = 1$$

If the synthesis window v is assumed to be rectangular, then it is obvious that overlapping analysis windows must sum to a constant value. However, also for tapered windows this can be the case. Figure 3.21 displays three types of analysis windows that are overlapped such that their sum is a constant value. For example, the Hann window in this case sums to 2 and has an overlap of 75%. Typically, overlaps of 50% are applied in practice, as this cuts the computational load in half. Care must be taken in the choice of window length, shift and the form of window such that they allow for perfect reconstruction, or at least quasi-perfect reconstruction, but still minimize computational load in applications where power consumption is an issue.

This idea of a designing analysis and synthesis windows with perfect reconstruction has been introduced by assuming that there are no modifications performed on the signal in the frequency domain. In practice, this is the purpose of transforming the signal in the first place.

As an example, let us consider the convolution of a long signal with a fixed filter function $h(n)$. In Figure 3.22, this signal $x[n]$ is rectangularly windowed and then each segment is convolved with the impulse response $h[n]$. Taking advantage of the efficient fast Fourier transform, this convolution is applied in the Fourier domain by simply multiplying the spectra of the signal and the filter. However, in time domain this multiplication corresponds to a *circular* rather than a *linear* convolution as was shown in Figure 3.13: due to cyclic convolution effects, the samples add the beginning of each segment are distorted. To avoid undesired effects due to cyclic con-

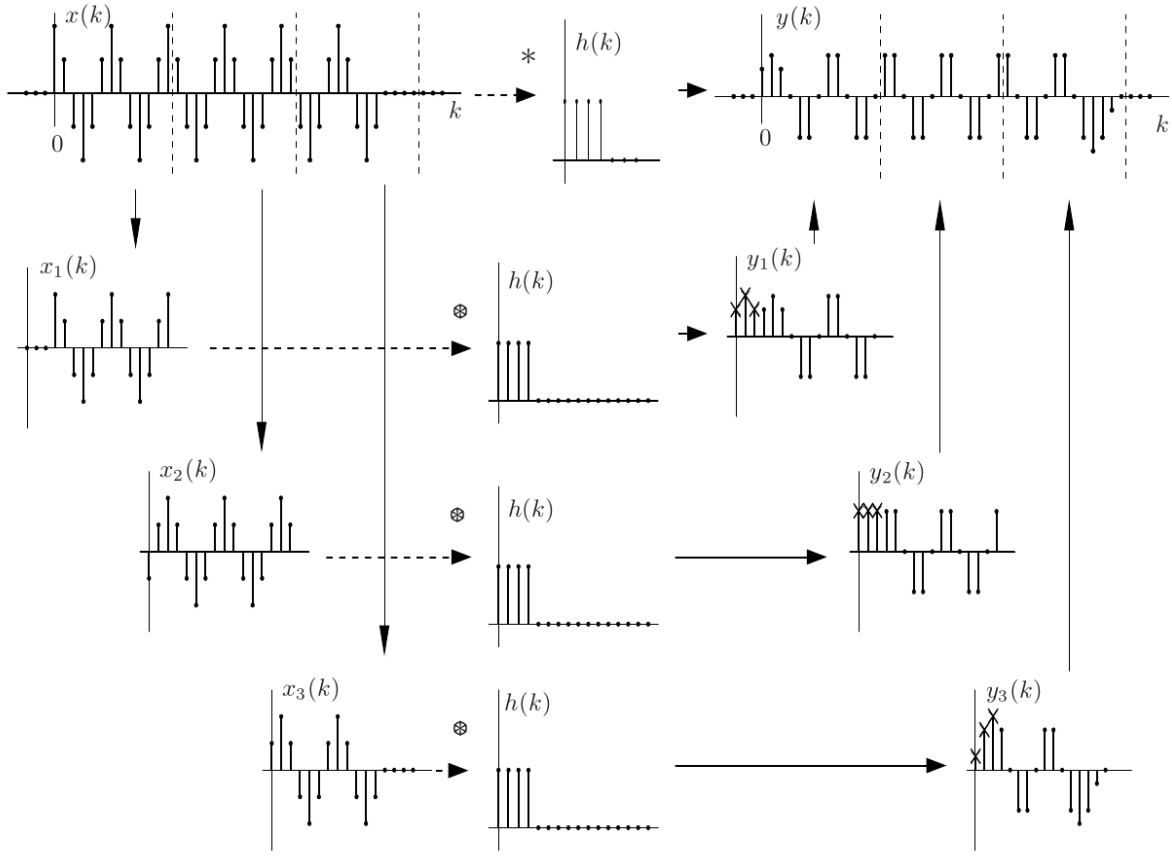


Figure 3.22: Overlap-save process displaying effects of cyclic convolution.[8]

volution, in the *overlap-save* procedure the erroneous samples are discarded, as denoted by the '$\times$'-symbols in Figure 3.22.

A different approach is to pad the signal segments with zeros, so that distortions due to cyclic convolution are avoided. The signals are then reconstructed by overlapping and adding the processed signal segments (see Figure 3.23). However, adding zeros has the disadvantage, that the computational load is increased.

A third method is therefore to not pad with zeros but to attenuate the erroneous samples at the frame boundaries that result from cyclic convolution using a tapered synthesis window. The time-domain signal is then reconstructed using overlap-add. While the resulting time-domain signal does not perfectly correspond to a linear convolution anymore, the advantage is a reduced computational complexity. A simple and practical choice is to use the square-root of a Hann-window both for spectral analysis and synthesis. This approach is for instance often used for speech enhancement.

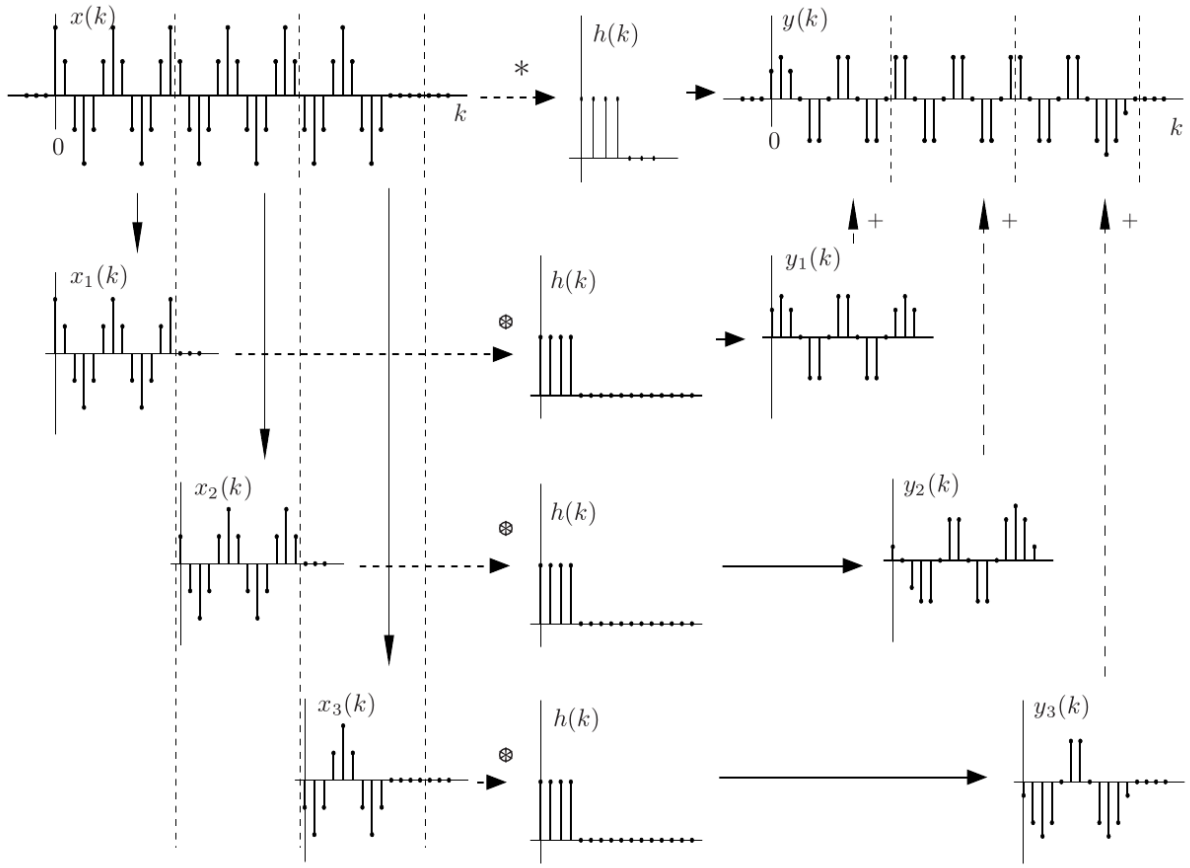


Figure 3.23: Overlap-add process.[8]

4 — The Vocal Tract and LPC analysis

Learning objectives

- Tube Model of the Vocal Tract
- Linear Prediction

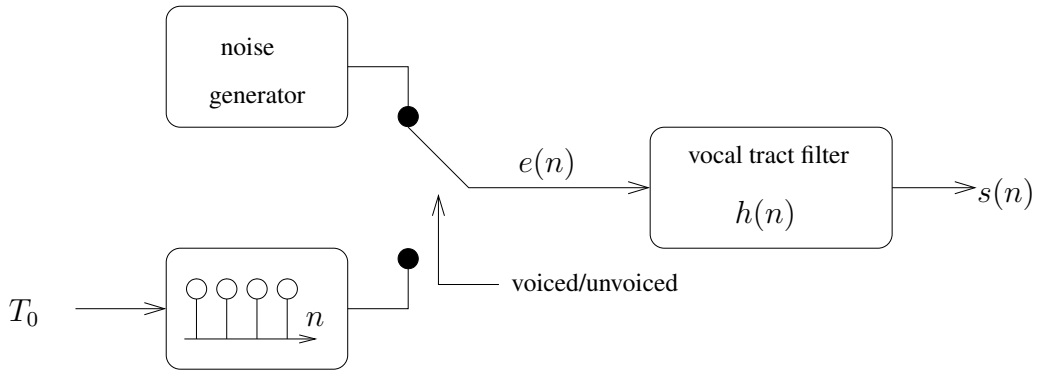


Figure 4.1: Source filter model of the vocal tract

4.1 Tube Model of the Vocal Tract

The first chapter introduced the concept of the source filter model of speech production in which an excitation source (air passing through the glottis) is passed through a filter (the vocal tract) to produce speech. The air flow from the lungs can be regulated by a periodic opening and closing of the glottis to produce a voiced speech sound or, when the glottis remains opened, an unvoiced sound. It was also shown in Section 3.5 that the vocal tract filter can be separated from the spectrum of windowed speech segment by extracting the spectral envelope. The shape of the vocal tract produces resonances (formants) that are critical in the differentiation of speech sounds. The goal of this chapter is to derive a mathematical model of the vocal tract filter that mirrors the physical processes involved in speech production. This model will also allow us to extract this spectral envelope and compute parameters that estimate the instantaneous shape of the vocal tract from a windowed recording of speech. These parameters, in addition to the estimated fundamental frequency derived in Section 2.1, will be the fundamental components for further processes such as speech coding, transmission and synthesis.

Figure 4.1 displays this model of speech production decomposed into an excitation source filtered by a vocal tract filter. In Section 2.1, voiced speech was processed by a fundamental frequency estimator so that an artificial excitation signal could be produced. This signal was simply a series of impulses with a distance corresponding to the estimated fundamental period which is essentially a model of the opening and closing of the vocal chords. On the other hand, it was shown that an unvoiced speech sound can be simply modeled with a random noise generator. To complete the excitation model, a switch must be included to distinguish between the two cases. One could also imagine a weighted sum of the two excitation signals so that mixed excitation signals, like [v] or [z] can be more accurately modeled.

We will begin by modeling the vocal tract as a tube. This simplified model can be seen in the left of Figure 4.2. An upper and lower tube correspond to the nasal and oral cavity, respectively.

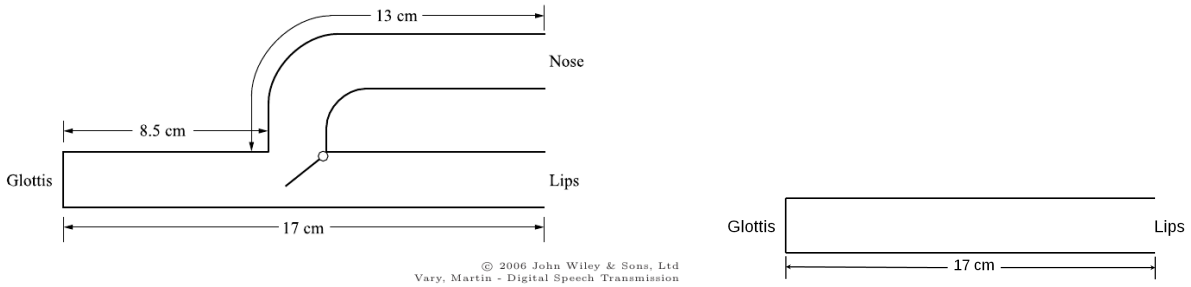


Figure 4.2: (Left) Tube model including nasal tract model. (Right) Simplified tube model of the vocal tract. [7]

The switch in the middle represents the velum which decouples the nasal tract form the oral cavity. It turns out that this model can be even further simplified to the simple tube on the right of Figure 4.2. This simplification facilitates the modeling and still produces intelligible results, however the more nasal speech sounds are not very well represented.

Sound is a pressure wave, meaning that what we hear is not the result of air particles traveling from the source of the sound to our ear but a traveling wave created by these particles pushing against each other and oscillating at a specific amplitude and frequency. If these air particles are in a tube, the tube walls would only permit a lengthwise direction of propagation. A snapshot in time of this wave would reveal areas of compression and areas of refraction as seen in Figure 4.3. If we were to look at a single point along the tube, we would see that it undergoes a periodic oscillation of pressure minima and maxima over time. This can be plotted as a waveform that represents sound pressure level as a function of position along the length of the tube. Because this pressure oscillates at each position, another waveform can be plotted for each position as a function of time. If we now consider a tube that is not endlessly long, but has an opening or a closing, we enforce certain boundary conditions onto the pressure wave.

We can now define mathematically the acoustic pressure, $p(x,t)$, that we have seen is a function of both space and time. We also define the particle speed, $u(x,t)$, as the speed at which the air particles oscillate. And finally, the volume velocity, $v(x,t) = u \cdot A$, is defined as the velocity of the air particles in an infinitesimal volume at each position in the tube.

From this we can form an acoustic impedance, a concept that is analogous to an electric impedance:

$$Z(x,t) = \frac{p(x,t)}{v(x,t)}.$$

Whenever there are sudden jumps in impedance $Z_1 \rightarrow Z_2$, for instance a change in diameter of the tube, some energy is reflected and some energy is transmitted. This is characterized by the reflection coefficient and is a function of the two impedances.

$$r = \frac{Z_2 - Z_1}{Z_2 + Z_1}$$

When the air molecules of a sound wave encounter the closed end of a tube, they encounter a

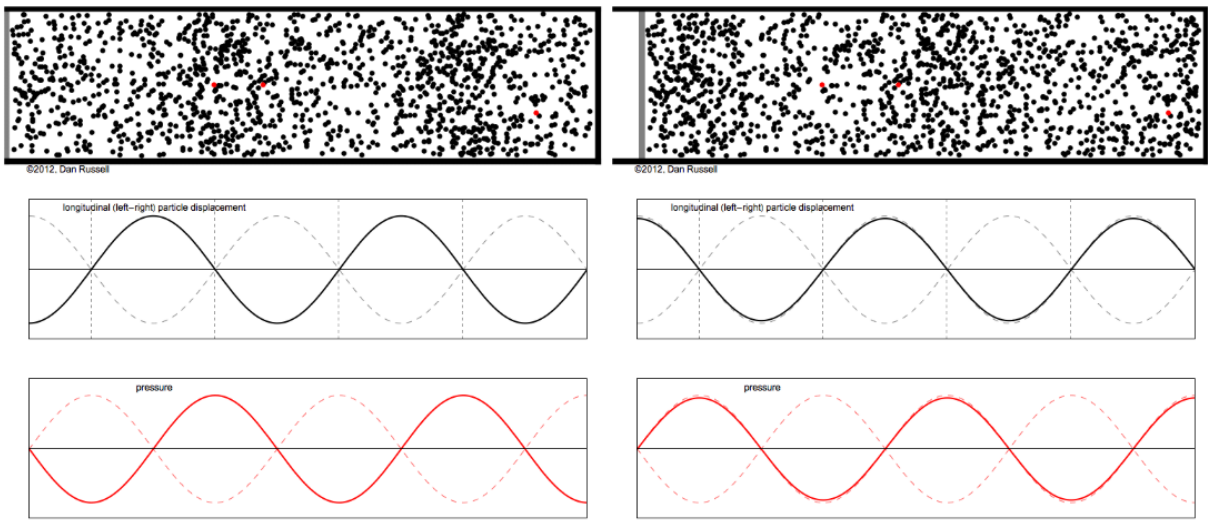


Figure 4.3: Particle motion in a resonating tube. (Top) Particle location, (Middle) particle displacement, (Bottom) pressure oscillations within a tube with a closed end.

medium that can be considered to have infinite impedance, $Z_2 = \infty$. This results in a reflection coefficient of $r = 1$ and corresponds to no phase change in the reflected wave. We now distinguish between the pressure and the velocity at the open and closed end of a tube. Exactly at the closed end of the tube, the particles do not move and the velocity is zero. However, the pressure at the closed oscillates between the maximum and minimum pressure. At the open, end it is the other way around: as the particles move freely, the velocity oscillates between the maximum and minimum velocity. However, as the particles move freely, also no pressure is being built up. Thus, the pressure at the open tube boundary is "zero", or the average value about which the pressure oscillates. If the pressure is zero then we can say that $Z_2 = 0$ and that the reflection coefficient at the open end would therefore be $r = -1$. The exiting air molecules are more free to move and transfer their energy to the surrounding air and the reflected wave is therefore 180° phase shifted. This results in the air molecules of the sound oscillating about their max and min values as they transfer their energy to the new medium.

- At the closed end of a tube (schallhart):
  $v(x,t) = 0$; $p(x,t)$ maximal; $Z_2(x,t) = \infty$; $r = 1$

- At the open end of the tube (schallweich):
  $p(x,t) = 0$; $v(x,t)$ maximal; $Z_2(x,t) = 0$; $r = -1$

The forward and back traveling waves can interfere constructively to produce a standing wave. This standing wave is known as a resonance that corresponds to the geometry of the resonating chamber. Figure 4.4 shows a closed and open ended tube which contains a standing pressure wave that is the first resonance of the tube. We can see that for wavelength of λ, the first resonance appears at $l = \frac{\lambda}{4}$.

Using the above concept, the vocal tract can be modeled as a tube of $l = 17\,\mathrm{cm}$ and the first

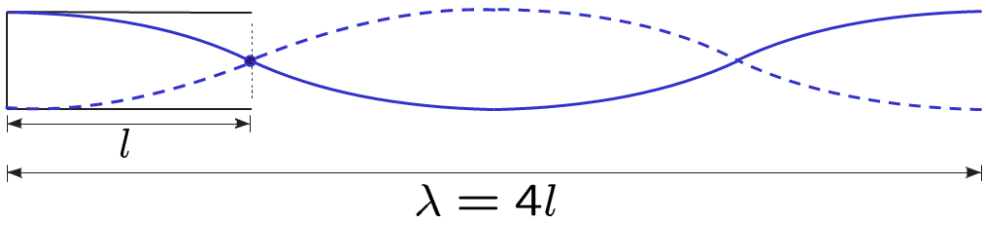


Figure 4.4: Resonances of a pressure wave in a tube with one closed end and one open end.

resonance frequency can be calculated.

$$l = \frac{\lambda}{4} \quad \Rightarrow \quad \lambda_o = 4l \quad \Rightarrow \quad f_o = \frac{c}{\lambda_o} = \frac{340\frac{m}{s}}{4l} = \frac{340\frac{m}{s}}{4 * 17\,\text{cm}} = 500\,\text{Hz} \;\; \text{first resonance}$$

A further analysis of Figure 4.4 reveals that there will be additional resonances at $l = [\frac{\lambda}{4}, \frac{3\lambda}{4}, \frac{5\lambda}{4}, \frac{7\lambda}{4}, ...]$, a process that continues infinitely. Of course, realistically, the length of the vocal tract tube is held constant, and the above process can be used to determine each additional resonance frequency:

$$f_{\text{res}} = [500\,\text{Hz}, 1500\,\text{Hz}, 2500\,\text{Hz}, 3500\,\text{Hz}, ...]$$

Notice that there is roughly one resonance per $1000\,\text{Hz}$. For telephone speech with a $8\,\text{kHz}$ sampling frequency and a $3500\,\text{Hz}$ bandwidth, we would therefore need to model 4 resonances to produce intelligible speech. This is of course an extremely simple model of the vocal tract in which only one speech sound can be modeled. The resonances corresponding to formants do not change over time because they are bounded by the geometry of the tube. In order to produce different resonances that correspond to different phonemes, it is therefore necessary to concatenate tube segments that are able to change their diameters and thus their cross-sectional area as seen in Figure 4.5.

We now have a vocal tract model in which different geometries can be modeled at each instance in time by concatenating tubes of varying diameters. The cross sectional area is plotted as a function of each tube segment in Figure 4.6 in what is called the *area function*. If the length of the vocal tract is given as L and we divide it up into n segments, then each segment will be of length $\Delta x = \frac{L}{n}$ and correspond to a specific cross sectional area. If we assume that each tube segment is of constant diameter, we can therefore calculate the impedance at the boundary between each concatenated tube segment as a function of the two surface areas. Once the impedances are known, the reflection and transmission of energy at each boundary can be determined and the concatenated model will allow for the calculation of the vocal tract transfer function.

The model must first make several assumptions in order to facilitate calculation. By assuming that the tubes are idealistic tubes of fixed cross sectional area per segment in which there is no friction, it follows that that the sound propagates as a plane wave. We can then use the plane

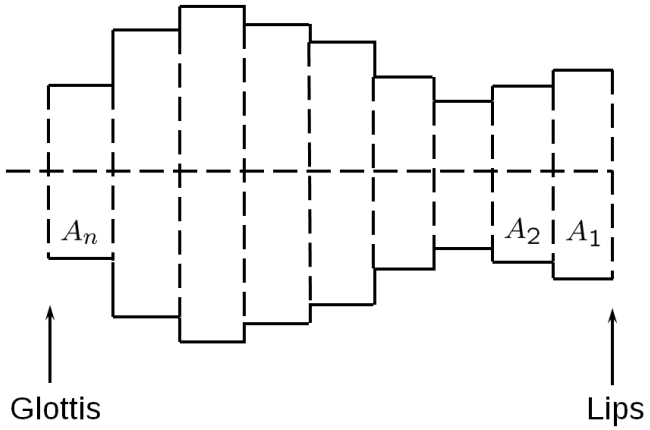


Figure 4.5: Vocal tract modeled as concatenated tube segments of different cross sectional areas [8].

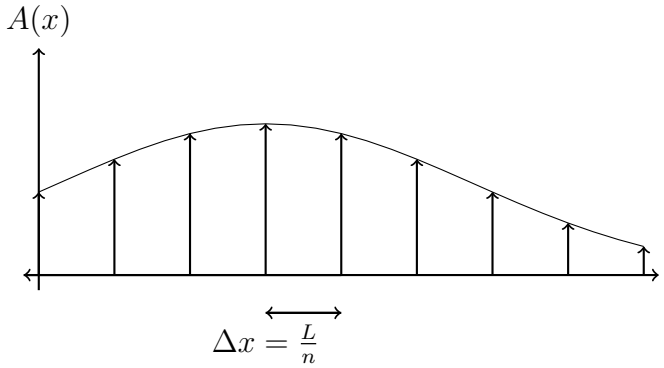


Figure 4.6: The area function of the vocal tract model. Cross sectional area is plotted for each tube segment representing the vocal tract [8].

wave equations for sound propagation that relate pressure and volume velocity. Looking at Equations (4.1), we see that a change in pressure in space is proportional to the change of the particle velocity over time. At the same time, the particle velocity across space is also proportional to the change in pressure over time.

$$-\frac{\partial p}{\partial x} = \rho \frac{\partial u}{\partial t} \qquad\qquad -\frac{\partial u}{\partial x} = \frac{1}{\rho c^2}\frac{\partial p}{\partial t} \tag{4.1}$$

$$\rho : density$$

Because the model deals with volumes of flowing air, the Equations (4.1) must be written in terms of the volume velocity, $\frac{\partial v}{\partial t}$. If A_n is the area of the n^{th} segment then the volume velocity is simply the product of the cross sectional area and the particle velocity through the segment:

$$\partial v = A_n \partial u, [\frac{m}{s} m^2]$$

This can be substituted into the differential equations for pressure and particle velocity:

$$-\frac{\partial p}{\partial x} = \frac{\rho}{A}\frac{\partial v}{\partial t} \qquad\qquad -\frac{\partial v}{\partial x} = \frac{A}{\rho c^2}\frac{\partial p}{\partial t}$$

We would now like to relate these two expressions by finding the change of the pressure over space as a function of the change of pressure over time. Both equations can be differentiated with respect to ∂x and ∂t and then substituted into Equations (4.1) to produce:

$$-\frac{\partial^2 p}{\partial x^2} = \frac{\rho}{A}\frac{\partial^2 v}{\partial t \partial x} \qquad -\frac{\partial^2 v}{\partial t \partial x} = \frac{A}{\rho c^2}\frac{\partial^2 p}{\partial t^2} \quad \Rightarrow \quad \frac{\partial^2 p}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2}$$

$$-\frac{\partial^2 p}{\partial x \partial t} = \frac{\rho}{A}\frac{\partial^2 v}{\partial t^2} \qquad -\frac{\partial^2 v}{\partial x^2} = \frac{A}{\rho c^2}\frac{\partial p}{\partial x \partial t} \quad \Rightarrow \quad \frac{\partial^2 v}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 v}{\partial t^2} \tag{4.2}$$

These results show that the acceleration of pressure over space is proportional to the acceleration of pressure over time. The same can be shown for the particle velocity. These second order differential equations can be solved using harmonic solutions.

We know that there is a reflection at each tube boundary in which there is a change in cross sectional area, A_i. This implies that there will be reflection coefficients, r_i, for each of these boundaries. At each of these boundaries, the energy of the wave separates into a backward (reflected) and a forward (transmitted) traveling wave. Figure 4.7 depicts this process with f representing the forward traveling wave and b representing the backward traveling wave. The transmitted and reflected waves both overlap with the incoming wave at the next time point in the system. We can therefore expect that our reflection coefficients will not be 1 and -1 like for the open and close tube, but some number in between.

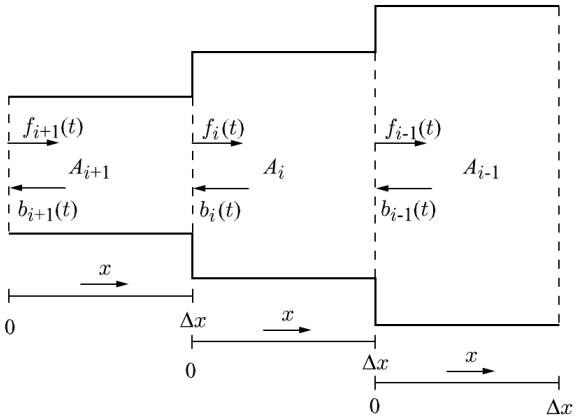


Figure 4.7: Backward and forward traveling waves resulting from differences in cross sectional area of adjacent tube segments. [8]

So we begin the approach to our solution by working with the superposition of the forward and backward traveling waves. Snapshots of a traveling wave would all be the same, the only difference is that they are shifted in time. If x is a spatial dimension and c is the speed of the wave, then the time shift can be represented as $(t - \frac{x}{c})$ We can therefore represent our forward traveling wave as $f(t - \frac{x}{c})$ and the backward traveling wave as $b(t - \frac{x}{c})$. In the previous description of the superposition of reflected air molecules, it was shown that pressures add because there is an increase in internal energy and velocities must subtract. We can therefore write:

For velocity, forward and backward travelling waves subtract.

$$v(x,t) = f(t - \frac{x}{c}) - b(t + \frac{x}{c})$$

For pressure, forward and backward traveling waves add. Also to make the units correct, the pressure must be multiplied by the acoustic impedance. $Z = \frac{\rho c}{A}$

$$p(x,t) = Z(f(t - \frac{x}{c}) + b(t + \frac{x}{c}))$$

These equations apply to each segment. The pressure and velocity values at each segment boundary must be the same, and therefore imposes boundary conditions on the calculation. We have shown that Δx is the segment length, so we can write the boundary conditions for the i^{th} segment as:

$$v_i(x = \Delta x, t) = v_{i-1}(x = 0, t)$$

$$p_i(x = \Delta x, t) = p_{i-1}(x = 0, t)$$

These equations can be used in conjunction with Equations (4.2) that relate the forward and backward traveling waves for pressure and velocity. And for this we introduce τ which is $\frac{\Delta x}{c}$ as the discretized time the wave takes to travel through one of the segments.

$$f_i(t-\tau) - b_i(t+\tau) = f_{i-1}(t) - b_{i-1}(t)$$

$$Z_i(f_i(t-\tau) + b_i(t+\tau)) = Z_{i-1}(f_{i-1}(t) + b_{i-1}(t))$$

We have established a relation between the forward and backward traveling waves of two successive segments. The next step is to find a function that relates the forward and traveling waves at neighboring segments of each end of the tube, (f_{i-1}, f_i, b_{i-1}) and (f_i, b_i, b_{i-1}). Looking at Figure 4.5 we can see that a forward traveling wave is transmitted into the adjacent segment, but parts of it are reflected back resulting in a backward traveling wave in the other direction. The reflection coefficient at the boundary determines how the wave is transmitted and reflected. The same also holds for when we look at the backward traveling waves. We can then write for both boundaries:

$$f_{i-1} = g_1(f_i, b_{i-1}) \qquad b_i = g_2(f_i, b_{i-1}) \tag{4.3}$$

Equations (4.3) are two equations with two unknowns and can be solved by solving them for f and plugging them into each other. The result is very nice and intuitive as it shows that the relation between the transmitted and reflected parts depend on the reflection coefficients .

$$f_{i-1}(t) = (1+r_i)f_i(t-\tau) + r_i b_{i-1}(t)$$

$$b_i(t) = -r_i f_i(t-2\tau) + (1-r_i)b_{i-1}(t-\tau)$$

r_i comes into the equation because it is related to the impedances between the successive segments

$$r_i = \frac{Z_i - Z_{i-1}}{Z_i + Z_{i-1}}$$

Parts of the forward traveling wave are transmitted and at the same time, parts of the backward traveling wave are reflected at the boundary. These two add waves sum to form the new wave in the new segment. The backward traveling wave in the next segment is composed how of the reflected part of the previous forward traveling wave and the part of the backward traveling wave transmitted from the next segment. This can be drawn into the Kelly Lochbaum structure shown in Figure 4.8.

We see that the forward traveling wave, f_{i-1}, that leaves segment i then enters segment $i-1$ and consists of a portion of f_i delayed by τ. This is the time that the wave f_i needs to travel through the segment. It is then partly propagated into $i-1$ with weight $1+r_i$ and a portion of b_{i-1}

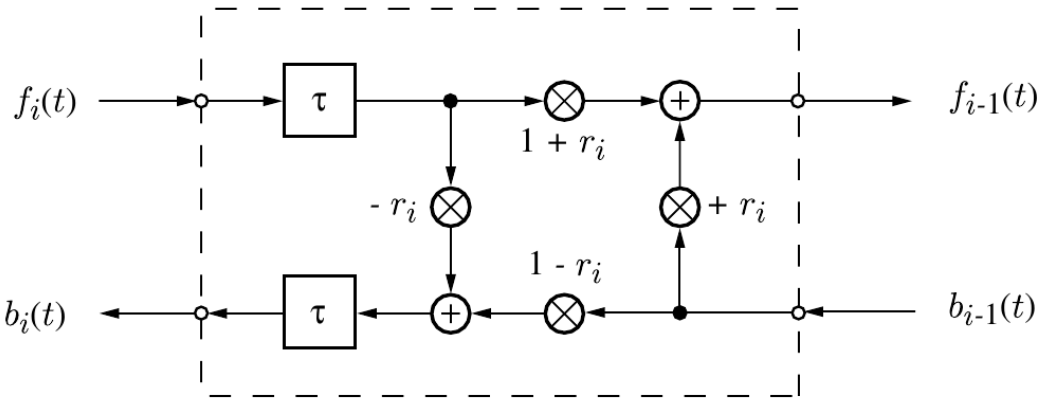


Figure 4.8: The Kelly-Lochbaum structure for representing a time based model for the reflection of a sound wave between adjacent tube segments. [8]

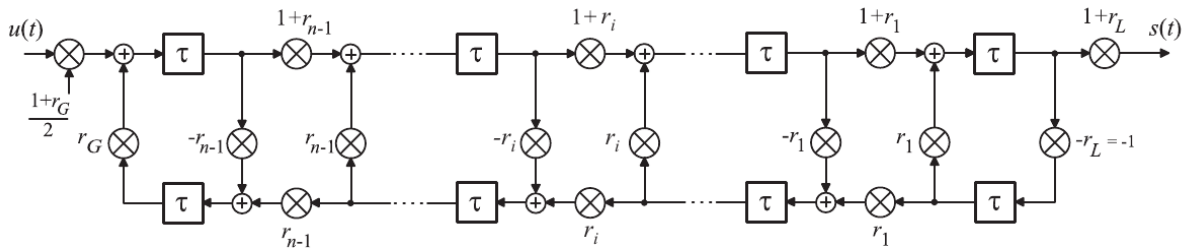


Figure 4.9: The concatenated Kelly Lochbaum structure. This will serve as the mathematical model of the vocal tract transfer function. [8]

partly reflected with weight r_i. The process can also be seen for the backward traveling wave in Figure 4.8.

This process completely describes the behavior of only one of the tube segment boundaries, however we can now concatenate these results to produce the final mathematical model of the vocal tract in Figure 4.9. Further conditions can be imposed on the model by assuming that there is no backward traveling wave at the first segment (corresponding to the glottis) and the last segment (corresponding to the lips). The schematic in Figure 4.9 can be viewed as the time domain response of an input, $u(t)$ and an output $s(t)$.

A filter can be characterized by its impulse response computed in continuous time. In our concatenated model, each segment introduces a fixed delay τ. Let us assume that we put a pulse into the system and measure the impulse response. We expect that the first non-zero output will be after $n\tau[s]$. If n is the number of segments and τ is the time that it takes for the signal to travel through a segment, then the time domain output is also a series of weighted impulses that are temporally spaced at τ. This is because we have spatially discretized the vocal tract function. We know that the infinite impulse response for a stable system continuously decays over time, but because the first reflection will go through the delay and then back again, the impulse response will be discretized into segments of length 2τ. This feedback structure is a recursive system and can be very nicely modeled as a digital filter. However, if we want to model the vocal tract

impulse response perfectly, then we will need to have a sampling frequency that captures the temporal processing of the digital filter. If the filter produces a new output at $2\tau[\frac{samples}{sec}]$ then, using the Nyquist sampling theorem, the sampling rate must be at least $f_s = \frac{1}{2\tau}$.

4.2 Linear Prediction

Now that we have a mathematical representation for the source filter model of the vocal tract, we must create a method by which we can estimate the reflection coefficients of the vocal tract filter from a given speech signal. These coefficients determine the weights of a digital filter through which an excitation signal is passed to produce synthesized speech. This filter is *causal* meaning that future values of the signal cannot affect the current output. This property allows us to use the method of *linear prediction* to define a process that allows us to extract these reflection coefficients. Because the concatenated tube model of the vocal tract is a stable system and modeled using a recursive loop, the impulse response of the corresponding digital filter will decay over time, however it will be theoretically infinitely long and cannot be realized computationally in the present form. The current section will attempt to workaround this issue by introducing the *autoregressive moving average* (ARMA) model of a digital filter. We now define the output filtered signal, $s(n)$, as the convolution of the impulse response of the vocal tract filter, $h(m)$, and the excitation signal, $e(m)$:

$$s(n) = \sum_{m=0}^{\infty} h(m)e(n-m)$$

Because we cannot model infinitely many coefficients, we replace the infinite sum by two finite sums in which one of the finite sums represents a recursive structure. We now introduce the ARMA structure by writing the output of the vocal tract filter, $s(n)$, as the sum of a finite moving average part and an autoregressive part. Because the model is recursive, the signal at time point n will be a function of past samples of the signal.

$$s(n) = \underbrace{\sum_{m=0}^{q} b_m e(n-m)}_{\text{MA}} - \underbrace{\sum_{\nu=1}^{p} a_\nu s(n-\nu)}_{\text{AR}}$$

We can now take the z-transform of the signal, move the transform operator into the sum because it is a linear operator, and then do a bit of algebra to simplify.

$$S(z) = E(z)\sum_{m=0}^{q} b_m z^{-m} - S(z)\sum_{\nu=1}^{p} a_\nu z^{-\nu}$$

$$S(z)(1+\sum_{\nu=1}^{p} a_\nu z^{-\nu}) = E(z)\sum_{m=0}^{q} b_m z^{-m}$$

The transfer function of the system can now be defined as the output over the input.

$$H(z) = \frac{S(z)}{E(z)} \overset{a_0=1}{=} \frac{\sum_{m=0}^{q} b_m z^{-m}}{\sum_{\nu=0}^{p} a_\nu z^{-\nu}} \tag{4.4}$$

The transfer function, $H(z)$, consists of two polynomials in z. We can employ the fundamental theorem of algebra in order to find a different representation for these polynomials. This states that every non-zero single variable polynomial has at least one root. This implies that a polynomial of order n has n roots.

$$\frac{p(z)}{(z-a)} = q(z) + \frac{R}{z-a} \Longrightarrow \text{ R: not a function of z}$$

$$p(z) = (z-a)q(z) + R \Longrightarrow \text{q(z) is a polynmomial w than p(z)}$$

$$\text{if } a = z_0 \text{ is a root, then}$$

$$p(z_0) = 0 = (z_0 - z_0)q(z) + R \implies \text{R = 0}$$

$$\text{if } a = z_0 \implies \text{ is a root, then the residual is zero}$$

To make the current transfer function into a polynomial, we must have a form in which we have positive exponents and the first coefficient is weighted by one. We do this to the transfer function in Equation (4.4)and then factorize it.

$$H(z) = \frac{S(z)}{E(z)} \overset{a_0=1}{=} \frac{\sum_{m=0}^{q} b_m z^{-m}}{\sum_{\nu=0}^{p} a_\nu z^{-\nu}} = \frac{b_0 z^{-q}}{z^{-p}} \frac{z^q + \frac{b_1}{b_0} z^{q-1} + ... + \frac{b_q}{b_0}}{z^p + a_1 z^{p-1} + ... + a_p} = z^{p-q} b_0 \frac{\prod_{m=1}^{q}(z - z_{0m})}{\prod_{\nu=1}^{p}(z - z_{\infty\nu})}$$

$$z_{0_m}\text{: Roots of numerator polynomial} \implies \text{zeros of H(z)}$$

$$z_{0_\nu}\text{: Roots of denominator polynomial} \implies \text{poles of H(z)}$$

We can now see that an ARMA model in the time domain corresponds to a pole/zero model in the frequency domain.

$$\text{ARMA model (time-domain)} \quad \circ\!\!-\!\!\bullet \quad \text{pole/zero model (Z-domain)}$$

If we assume that a process can be perfectly represented by a moving average part, then the autoregressive part of our finite filter goes to zero. The moving average part of the equation would then correspond to an all zero process in the frequency domain. MA process→all zero filter.

$$s(n) = \sum_{m=1}^{q} b_m e(n-m) \quad \circ\!\!-\!\!\bullet \quad S(z) = E(z) z^{-q} b_0 \prod_{m=1}^{q} (z - z_{0_m})$$

And then the reverse is true for the autoregressive part. AR process→all pole filter.

$$s(n) = b_0 e(n) - \sum_{\nu=1}^{p} a_\nu s(n-\nu) \quad \circ\!\!-\!\!\bullet \quad S(z) = E(z) b_0 \frac{z^p}{\prod_{\nu=1}^{p} (z - z_{0_\nu})} \rightarrow \text{frequency- domain}$$

The poles of our vocal tract filter correspond to the formants or the resonances of our vocal tract geometry. When we modeled the vocal tract by a tube model and neglected the nasal cavity, we showed that it can be best approximated by an autoregressive process. We therefore neglect the moving average part of the finite filter representation and represent the frequency response of our vocal tract filter as an all pole filter.

$$H(z) \approx b_0 z^p \frac{1}{\prod_{\nu=1}^{p} (z - z_{\infty\nu})} = \frac{b_0}{\sum_{\nu=0}^{p} a_\nu z^{-\nu}}. \tag{4.5}$$

This recursive structure resembles the recursive structure obtained from the tube model (Nasal tract, glottal, and labial filter neglected), meaning that the impulse response of the vocal tract is infinitely long. Therefore, the magnitude of the filter's frequency response will be better approximated by increasing the AR filter order so that an infinite filter order would approximate the signals amplitude characteristics perfectly. This is not true for the phase characteristics, however, still intelligible speech at decent quality can be produced.

4.3 Computation of AR coefficients

The ARMA model for our sythesized speech signal was given as follows:

$$s(n) = \underbrace{\sum_{m=0}^{q} b_m e(n-m)}_{\text{MA}} - \underbrace{\sum_{\nu=1}^{p} a_\nu s(n-\nu)}_{\text{AR}}$$

We then said that the model is better approximated by an autoregressive model. We now take the moving average part and replace it with a function that we will call the innovation, $e(n)$. This function is scalable, by b_0, a scalar coefficient that will determine the power of the current speech sound. The autoregressive part is a digital filter of order p that is the time domain representation of the vocal tract frequency response calculated in the previous section. The output is a linear combination of past samples of our signal weighted by the autoregressive coefficients. Notice how this equation implies that successive speech samples are correlated.

$$s(n) = b_0 e(n) - \sum_{\nu=1}^{p} a_\nu s(n-\nu)$$

However, at time n, the prediction of the speech signal by AR coefficients, $s(n)$, cannot account for the innovation as it is uncorrelated with past samples.

$$\text{Prediction: } \hat{s}(n) = -\sum_{\nu=1}^{p} \hat{a_\nu} s(n-\nu)$$

We would now like to estimate these autoregressive coefficients, a_ν so that the predicted signal $\hat{s}(n)$ is as close to the true signal $s(n)$ as possible.

$$d(n) = s(n) - \hat{s}(n) = b_0 e(n)$$

To best estimate the AR coefficients, we want to minimize the average error between the speech and the predicted speech. We do this mathematically by minimizing the mean of the squared error between the signal and its prediction.

$$a_\nu = \underset{a_\nu}{\operatorname{argmin}}\, E\left\{(s(n) - \hat{s}(n))^2\right\}$$

Multiplying out the terms in the parenthesis gives us a second degree polynomial. The common method of minimizing such a function is to take the first derivative and set it equal to zero. The derivation can be moved into the expected value because it is a linear operator. The chain rule can then be implemented to simplify the expression.

$$0 \stackrel{!}{=} \frac{\partial E(d^2(n)}{\partial \hat{a_\nu}} \stackrel{chainrule}{=} E(2d(n)\frac{\partial}{\partial \hat{a_\nu}}(s(n) + \sum_{\nu=1}^{p} \hat{a_\nu} s(n-\nu))$$

Here, $\hat{s}(n)$ is a function of $\hat{a}_\nu$ but $s(n)$ is not. As we are only taking the derivative with respect to the ν^{th} component all other terms of the sum are considered constants and thus goto zero.

$$= E(2d(n)s(n-\nu))$$

$$= 2E((s(n) + \sum_{\nu=1}^{p} \hat{a_\nu} s(n-\nu))s(n-\nu))$$

$$= E(s(n)s(n-\nu)) + E(\sum_{\mu=1}^{p} \hat{a_\mu} s(n-\mu)s(n-\nu))$$

The expected value of a signal shifted against itself is defined as the autocorrelation function, $\phi_s(\nu)$.

$$\phi_s(\nu) = E(s(n)s(n-\nu))$$

The final equation that we must now solve is now:

$$0 = \phi_s(\nu) + \sum_{\mu=1}^{p} \hat{a_\mu} \phi_s(\nu - \mu)$$

Formally, to show that this is really a minimum, we would also have to show that the 2^{nd} derivative is positive. This is however the case here because when we again take the derivative with respect to a_ν, everything goes to zero except for the coefficient corresponding to the ν^{th} term. This means that the autocorrelation of the function at zero will always be positive. The autocorrelation of a signal at $lag = 0$, is basically the power of the signal and this is always positive.

$$\frac{\partial^2 E(d^2(n)}{\partial \hat{a_\nu}^2} = E(2\phi_s(0)) \geq 0 \rightarrow minimum$$

We can now write the equation as:

$$\phi_s(\nu) = -\sum_{\mu=1}^{p} \hat{a_\mu} \phi_s(\nu - \mu) = -(\phi_s(\nu-1)\hat{a_1} + \phi_s(\nu-1)\hat{a_1} + \phi_s(\nu-2)\hat{a_2} + \cdots + \phi_s(\nu-p)\hat{a_p})$$

In order to solve this, it helps to write the entire expression in matrix form.

$$\begin{pmatrix} \phi_s(1) \\ \phi_s(2) \\ \vdots \\ \phi_s(p) \end{pmatrix} = - \begin{pmatrix} \phi_s(0) & \phi_s(-1) & \phi_s(-2) & \dots & \phi_s(1-p) \\ \phi_s(1) & \phi_s(0) & \phi_s(-1) & \dots & \phi_s(2-p) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_s(p-1) & \phi_s(p-2) & \phi_s(p-3) & \dots & \phi_s(0) \end{pmatrix} \begin{pmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_p \end{pmatrix}$$

$$\phi_{\mathbf{s}} = -\mathbf{R_s}\hat{\mathbf{a}}$$

To compute the AR coefficients, we must find the solution for all $\hat{a_\nu}$. These are known as the *Wiener Hopf/Normal equations*. The solutions to the Wiener Hopf equations gives us the MMSE-optimal linear predictive coefficients, $\hat{a_\nu}$.

$$\implies \hat{\mathbf{a}}_{\mathbf{opt}} = -{\mathbf{R}_{\mathbf{s}}}^{-1}\phi_{\mathbf{s}}$$

We now have a method by which we can estimate our autoregressive coefficients. To solve these equations, we can employ a procedure called the *Levinson-Durbin algorithm*. This algorithm allows for an efficient computation of the solution. However, we must first know the autocorrelation function. It is defined as the expected value of the signal and a shifted version of the signal:

$$\phi_s(\nu) = E(s(n)s(n-\nu))$$

In practice, we must approximate this expected value operator by replacing the expectation by a sum, so we use a temporal averaging to approximate the expected value. The problem then arises that in order to perfectly approximate the expected value, the sum would have to be infinitely long. This is not possible for speech because our vocal tract changes over time and therefore speech is not stationary. In order to estimate these coefficients, we must compute the sum only within a range in which speech can be considered to be stationary. Speech is typically stationary within frame lengths of $5\,\mathrm{ms}$ to $50\,\mathrm{ms}$. The frame length that we choose also depends on the application that we have in mind. In practice, we estimate the autocorrelation, $\mathbf{R_s}$, only on short segments.

$$\widetilde{\phi}_s(\nu) = \frac{1}{n_2 - n_1} \sum_{n=n_1}^{n_t} \widetilde{s}(n)\widetilde{s}(n-\nu)$$

After windowing the signal, it doesn't matter if we shift the signal to the left or the right when computing the autocorrelation: the resulting linear sum is the same. This means that the correlation matrix becomes much simpler because $\hat{\phi}_s(\nu) = \hat{\phi}_s(-\nu)$. This now makes the matrix symmetric and *Toeplitz*. This special structure makes it easier to implement faster algorithms, namely the Levinson-Durbin recursion. The end result of the computation produces the reflection coefficients.

The AR coefficients, $\hat{\mathbf{a}}_{\mathbf{opt}}$, and the reflection coefficients, r_i, both capture the same information and can be computed from each other. The reflection coefficients are related to impedance differences at segment boundaries in our vocal tract model. These impedance differences arise as a result of the change in cross sectional area between tube segments. We can therefore approximate the shape of the vocal tract from the linear prediction of the original speech signal.

$$r_i = \frac{A_i - A_{i+1}}{A_i + A_{i+1}}$$

This process of synthesizing speech by filtering an excitation signal through an all pole vocal tract filter can be modeled with a vocoder. We now have a more explicit model of our source filter model. The excitation signal is multiplied by a certain gain that regulates the amount of energy in our signal. We then have the recursive structure of the autoregressive vocal tract filter in which $\hat{\mathbf{a}}_{\mathbf{opt}}$ is a vector of autocorrelation coefficients. We have shown that human speech contains roughly one formant per $1\,\mathrm{kHz}$. To model each formant, we must therefore model two AR coefficients. This means that an $f_s = 8\,\mathrm{kHz}$ sampling rate, would require 8 AR coefficients. We must also use an anti-aliasing filter that attenuates higher frequencies, so we add two more filter coefficients. This gives us roughly 10 AR coefficients to model telephone speech. Using this parametrization of a speech signal, a speech coding algorithm would only need to transmit the unvoiced/voiced decision, the fundamental period, the scaling parameter, and the AR coefficients. And from this information, we can reconstruct synthetic speech in a very efficient manner and with a very low bit rate. We transmit these parameters using a time window in which stationarity is assumed, namely $8\,\mathrm{ms}$ to $32\,\mathrm{ms}$. These windows can also be chosen to overlap, which allows for using non rectangular windowing functions as shown in Section 3.6.

Typically we also apply a pre-emphasis filter. The long time spectral envelope of human speech has a natural slope meaning that there is more energy in lower frequencies than in higher. This translates to about $-6\frac{db}{octave}$ in the frequency domain. By applying a pre-emphasis filter to our signal, this can be accounted for so that we have a more precise estimate of our formants. This process is also invertible.

We have now seen that a speech signal can be modeled as an excitation source filtered through a vocal tract filter. We have then developed a mathematical representation of this model so that we can parameterize the input speech signal into values representing the AR coefficients (or reflection coefficients, log area ratios, filter coefficients), the type of excitation (voice/unvoiced), and the gain of the excitation, b_0. The advantages of the AR Model are that these parameters can be computed efficiently by using the Levinson Durbin algorithm and that the process is always invertible if the vocal tract filter is stable. For these reasons, LPC analysis has important implications in speech coding and synthesis. It is also useful in speech analysis because it provides an estimation of the spectral envelope and allows for formant tracking.

5 — Cepstrum

Learning objectives

- Cepstrum definition
- Real and complex Cepstrum
- LPC to cepstral coeffcients

5.1 Cepstrum definition

The cepstrum is yet another transformation that allows us to better look at different characteristics of a signal in a different domain. The computed cepstral coefficients are simply another representation of the signal that allows for more efficient processing in applications such as speech recognition. The cepstrum is defined as the inverse Fourier transform of the logarithm of the spectrum. The analysis of this log spectrum allows us to differentiate between slowly and rapidly varying spectral components of the signal.

(Real) cepstrum

$$c[n] = \frac{1}{2\pi}\int_{-\infty}^{\infty} \log|X(e^{jn\Omega})|e^{jn\Omega}d\Omega \qquad X(e^{jn\Omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-jn\Omega}$$

Because the inverse Fourier transform of the magnitude log spectrum is computed, a new list of terms is given that are analogous to their spectral counterparts. For instance, the word cepstrum is just a twist of the word spectrum.

Cepstral terms

Cepstrum	$\Longleftrightarrow$	Spectrum
Liftering	$\Longleftrightarrow$	Filtering
Quefrency	$\Longleftrightarrow$	Frequency
Rahmonics	$\Longleftrightarrow$	Harmonics
Saphe	$\Longleftrightarrow$	Phase

Time domain speech signals can be obtained by convolving an excitation signal with the impulse response model of the vocal tract.

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

This convolution becomes a multiplication in the frequency domain.

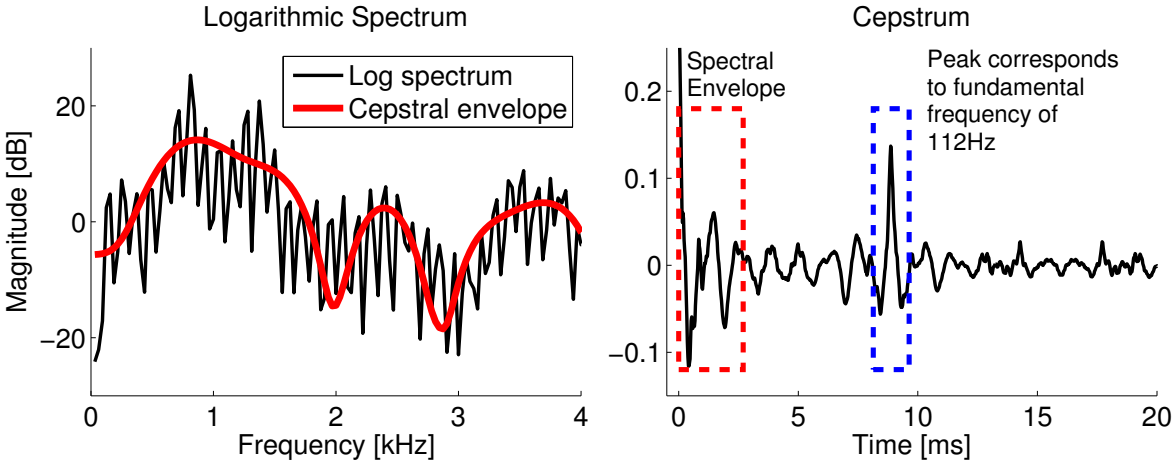


Figure 5.1: (Left) Log magnitude spectrum of voiced speech signal. (Right) Signal transformed into cepstral domain. The resulting separation of the envelope and fundamental frequency are shown in corresponding rectangles.

$$Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega})$$

The logarithm of the multiplication of the two spectra is the sum of the logarithms.

$$logY(e^{j\omega}) = logX(e^{j\omega}) + logH(e^{j\omega})$$

This is a linear superposition of the influence of the excitation signal and the vocal tract. Due to the linearity of the IDFT operator, this summation property remains even after computing the inverse Fourier transform.

$$\widetilde{y(n)} = \widetilde{x(n)} + \widetilde{h(n)}$$

Therefore, the complex cepstrum also corresponds to a linear superposition of the excitation signal and the vocal tract filter. Of course, the segment length used in the analysis must be long enough to capture the influence of the vocal tract filter and a significant number of pitch periods.

The same property also holds for the real cepstrum. In computing the real cepstrum, the logarithm of the magnitude of the spectral coefficients is used, however this additive superposition of the excitation signal and the vocal tract filter still exist. The log spec magnitude is commonly used in visualizing the short time spectrum of a time varying signal as shown in Figure 5.1. The left plot depicts the log spectrum magnitude of a voiced speech signal. The fundamental frequency and its harmonics can be identified as the rapid periodic components of the signal. The formants produced by the vocal tract model can be seen in the spectral envelope of the signal.

The Fourier transform of the log magnitude spectrum separates the frequency (now ”quefrency”) content of the spectrum. Therefore, lower quefrencies in the transformation correspond to slower fluctuations in the log magnitude spectrum. These slower fluctuations are seen in the spectral

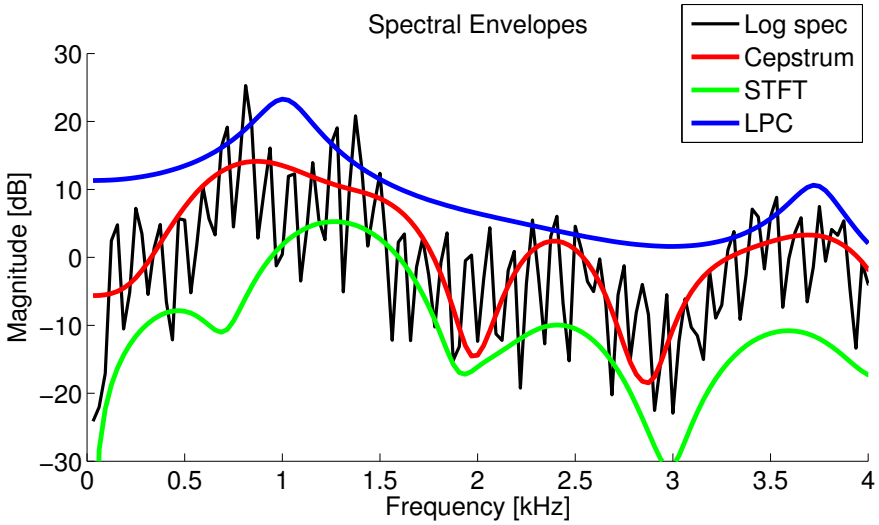


Figure 5.2: Different methods of spectral envelope extraction are contrasted for a voiced speech signal.

envelope of the log magnitude spectrum, and are a result of the vocal tract filter. Higher quefrencies correspond to more rapid fluctuations in the log magnitude spectrum and are the result of the higher frequency harmonics produced by the excitation signal. These harmonics have a more periodic structure that would be represented as a peak in the cepstral domain. The quefrency value of this peak corresponds to the fundamental period.

The quefrency values corresponding to the vocal tract transfer function and the fundamental period can be extracted through a process called cepstral liftering. The idea is similar to LPC analysis, however it employs a different technique. Like in LPC analysis, pre-emphasis may be applied and the signal is windowed into overlapping segments with a chosen analysis window. To extract the vocal tract transfer function, all higher cepstral coefficient are set to zero and only the lower coefficients are inverted. By re-synthesizing only the lower cepstral coefficients with an inverse Fourier transform, an estimation of the spectral envelope and thus the vocal tract transfer function is produced. Figure 5.1 shows a plot of a spectrum produced after inverting a liftered cepstrum. Notice that the liftering process captures only the spectral envelope of the original spectrum and produces a smoothed version of the original log magnitude spectrum.

Figure 5.2 also shows that the cepstrally smoothed envelope is the local average of the log magnitude spectrum. Because we are taking an average in the logarithmic domain, a certain bias with respect to power is introduced. This means that the cepstrally smoothed envelope will always be lower in power than the LPC spectral envelope. Additional methods can be implemented to compensate for this bias. Figure 5.2 shows this effect in addition to another fundamental difference between the two methods. Because the LPC analysis implements an all pole model to produce the vocal tract filter, there are sharp peaks associated with filter behavior near the poles.

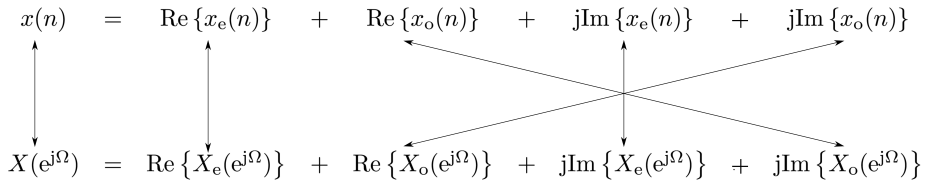


Figure 5.3: Symmetry relations of the Fourier transform.

5.2 Real and complex cepstrum

Complex cepstrum

$$\hat{X}(\mathrm{e}^{\mathrm{j}\Omega}) = \log X(\mathrm{e}^{\mathrm{j}\Omega}) = \log|X(\mathrm{e}^{\mathrm{j}\Omega})| + \underbrace{\mathrm{j}\phi_X}_{\mathrm{j}\arg X(\mathrm{e}^{\mathrm{j}\Omega})} \tag{5.1}$$

There are basically two definitions of the cepstrum. The real cepstrum is computed by taking the Fourier transform of the *magnitude* log spectrum. The complex cepstrum is computed from the *complex* log spectrum coefficients, thus preserving phase information. Because a complex signal can be represented in terms of its amplitude and phase, the logarithm of these two terms results in the summation of the log magnitude and of the complex valued phase as shown in Equation (5.1).

For real-valued time-domain signals (such as a speech signal), the spectrum, X, will be a complex conjugate signal, consisting of a real even part and a complex odd part. The real valued even part is represented when we plot a log magnitude spectrum and is the reason why these plots produce a mirror spectrum. When the real cepstrum is computed, the complex terms vanish with the magnitude calculation and we are only taking the inverse transform of the real even part. Looking at Figure 5.3 we see that the real cepstrum would have to be real valued and even. When computing the complex cepstrum of the signal, we take the inverse transform of the complex valued odd part of the spectrum. Looking again at Figure 5.3, we see that the complex cepstrum would also have to be real-valued. That is important point to remember and is often a source of confusion between the two cepstrum definitions. The complex cepstrum is called complex because it is computed on the complex spectral coefficients, *however it is real-valued*. It can also be seen in Figure 5.3 that the real cepstrum would be a subset of the complex cepstrum. It can be obtained by simply computing the even part of the complex cepstrum by using Equation (5.2).

$$c(n) = \frac{\hat{h}(n) + \hat{h}(-n)}{2} \tag{5.2}$$

The real cepstrum is used much more frequently in practice partly because our interpretation of magnitude spectra is much more intuitive. Phase information is much more difficult to interpret for humans and also for algorithms. Therefore, most algorithms that work on cepstral representations only evaluate the absolute value and work with the real cepstrum or some variant. One example are the Mel Cepstral coefficents that are used in speech recognition. Because the real cepstrum discards the phase information, it is non-invertible. To reproduce the original real valued time domain signal, the complex spectral coefficients would need to be preserved by using the complex cepstrum. When the process is invertible, then it is called *homomorphic*.

5.3 LPC to cepstral coefficients

The cepstral coefficients are just another representation of the vocal tract filter. When an auto regressive model for speech is assumed, it is possible to compute the cepstral coefficients from the LPC coefficients. Of course, we know that speech can never be perfectly modeled by an auto regressive model of finite order. For instance, nasal sounds with the velum open would need to be modeled by another tube and would therefore require zeros in the numerator of the vocal tract transfer function. These zeros are usually neglected to simplify the transfer function calculation because synthesized speech that neglects the nasal tract is still clear and intelligible. The cepstral coefficients can be directly calculated from the spectrum and from the auto regressive coefficients. These two methods will not be exactly the same, however when we assume that the auto regressive model perfectly describes speech production, it is possible to produce one from the other. We have shown that the auto regressive coefficients are also analogous to the reflection coefficients calculated between tube segments and the log area ratios of the tube segments. This section will introduce methods that relate all of these characterizations of the vocal tract so that they can be calculated from each other.

In LPC predictive coding, the vocal tract is modeled by an all pole system Using LPC analysis, the transfer function is represented by the quotient between the power of the signal, represented by the scaling factor, b_0, and a polynomial with the computed autoregressive coefficients of the speech segment as coefficients. Thistransfer function can also be represented in terms of a product of the poles.

$$H(z) = \frac{S(z)}{E(z)} = \frac{b_0}{1 + \sum_{\nu=1}^{p} a_\nu z^{-\nu}} = \frac{b_0}{z^{-p} \prod_{\nu=1}^{p} (z - p_\nu)}$$

$$z_{\infty\nu} = p \rightarrow poles$$

Because the product in the denominator is over p elements, we can split z^{-p} into $p_\nu z^{-1}$ by multiplying the term in the parenthesis by z^{-1}.

$$= \frac{b_0}{\prod_{\nu=1}^{p} (1 - p_\nu z^{-1})}$$

In order to compute the cepstrum of this expression, we need to take the logarithm. Because we are taking the logarithm of a quotient, we can express this as the difference between the logarithms of the numerator and denominator.

$$\hat{H(z)} = \log H(z) = \log b_0 - \sum_{\nu=1}^{p} \log\left(1 - p_\nu z^{-1}\right)$$

In order to compute the cepstrum, we must compute the inverse Z transform of this expression. Because of the logarithmic terms, the computation is difficult, however it can be simplified by using the series representation of the logarithm shown below.

$$\log(1-a) = -\sum_{m=1}^{\infty} \frac{a^m}{m}, |a| < 1$$

Substituting this identity into our expression for the logarithm of the transfer function, we get:

$$\widehat{H}(z) = \log b_0 - \sum_{m=1}^{\infty} \frac{p_\nu^m z^{-m}}{m}$$

It is now much easier to compute the inverse Z transform. Doing this and simplifying yields the following:

$$Z^{-1}\left\{\sum_{\nu=1}^{p} \log\left(1 - p_\nu z^{-1}\right)\right\} = \frac{1}{2\pi j} \oint \sum_{\nu=1}^{p} \sum_{m=1}^{\infty} \frac{p_\nu^m}{m} z^{n-m-1} dz$$

And now we can use the Cauchy integral theorem to simplify this expression.

$$\frac{1}{2\pi j} \oint Z^{n-m-1} dz = \begin{cases} 1 & n = m \\ 0 & \text{else} \end{cases}$$

Because the sums are all linear operators, they can be shifted so that the integral can be applied to the term within the sum. This can be done because the $\frac{p_\nu^m}{m}$ term is not a function of z. Using Cauchy's integral theorem, we can see that there is only a contribution to the summation when $n = m$. This now means that the expression can be further simplified.

$$= \begin{cases} \sum_{\nu=1}^{p} \frac{p_\nu^n}{n} & n \geq 1 \\ 0 & \text{else} \end{cases}$$

We still need to compute $Z^{-1}\{\log b_0\}$. The Fourier transform of a constant results in a delta spike at zero. Thus, the final expression for the complex cepstrum of our impulse response is:

$$\hat{H}(z) = \begin{cases} \sum_{\nu=1}^{p} \frac{p_\nu^n}{n} & n \geq 1 \\ \log b_0 & \text{n=0} \\ 0 & \text{else} \end{cases}$$

In this result, we see that for the auto regressive signal model, the complex cepstrum is zero for $n < 0$. When $n = 0$, the zeroth cepstral coefficient represents the logarithm of our scaling

parameter, b_0, and is a representation of the power of a signal. The higher cepstral coefficients are then a function of the poles, p_ν. If the poles of the auto regressive transfer function are given, the cepstral coefficients can be computed. If given the power, b_0, then the system is completely described. It is interesting from this derivation that the impulse response is zero for negative quefrencies of the auto-regressive system. This is generally not the case as we have already looked at examples in which the complex cepstrum is not zero for negative quefrencies. The explanation for this is that the filter is *minimum phase*. Minimum phase is a special characteristic of a system that applies when all zeros of the transfer function fall inside the unit circle. For an all pole system this is always the case because there are no zeros except trivial ones at 0. We can therefore say that an autoregressive system is always minimum phase, and for these kinds of systems, the negative cepstral coefficients will always be zero. Vice versa, a minimum phase system can also be computed from an arbitrary system by setting the negative quefrencies of the cepstral representation to zero and then possibly rescaling to account for energy loss.

It has already been shown that the real cepstrum can be calculated from the complex cepstrum, because the real cepstrum is the even part of the complex cepstrum.

$$c(n) = \frac{\hat{h(n)} + \hat{h(-n)}}{2}$$

Computing the even part of our complex cepstrum yields:

$$\hat{H(z)} = \begin{cases} \frac{1}{2}\sum_{\nu=1}^{p} \frac{p_\nu^{|n|}}{|n|} & n \neq 0 \\ \log b_0 & n = 0 \end{cases}$$

This is another interesting result because we see that the energy spreads is distributed in half to the negative and positive quefrencies. So now given an arbitrary system, the real cepstrum can be computed in the above manner by simply multiplying the coefficients corresponding to positive non-zero quefrencies by 2 and setting those coefficients corresponding to the negative quefrencies to zero. The minimum phase cepstral representation can then be transformed back into the spectral domain or even the time domain, where we have created a minimum phase signal from an arbitrary signal.

$$\hat{H}(z) = \begin{cases} c(0) & n = 0 \\ 2c(n) & n > 0 \\ 0 & else \end{cases}$$

This minimum phase characteristic can have important uses in low-latency filtering. Many filters that are usually applied are symmetric and this means that they have a certain delay. Because the process of filtering in the time domain corresponds to a convolution of the signal with the time domain impulse response of the filter, the delay for a symmetric filter would be half the filter length. A minimum phase signal has the exact same magnitude response as the original signal, however they differ in their phase responses. The benefit of the minimum phase signal is that the group delay of the filter is as small as possible. This can be an important trait given latency critical applications.

6 — Quantization

Learning objectives

- Uniform Quantization
- Non-uniform Quantization
- Adaptive Quantization
- Vector Quantization

6.1 Uniform Quantization

In order to perform digital processing, we must describe the output of a microphone recording a speech signal as a series of ones and zeros. We have already discussed discretization along the time axis in Section 3.2. A predetermined sampling frequency, F_s, is chosen, and the amplitude of the signal is recorded at constant time intervals. Mathematically, this process is represented by multiplying the signal with a delta comb, and could also be realized by means of a switch that closes every T_s seconds as depicted in Figure 6.1. Using the Nyquist theorem, it was also shown that sampling was a *lossless process* when the highest frequency in the signal is half the sampling frequency $\frac{F_s}{2}$. This means that even though we are discarding information between samples in the time domain digitization of the signal, it can still be perfectly reconstructed if the input signal is first lowpass filtered to remove frequencies above $\frac{F_s}{2}$. After the signal has been sampled, the amplitudes of the recorded samples must also be discretized, a process requiring that the signal is mapped onto discrete bins of predetermined amplitudes. Unlike sampling, this is a *lossy process* that is referred to as *quantization*.

Quantization also extends to parametric forms of speech coding shown in Figure 6.1. LPC analysis decomposes a signal into an excitation signal and its vocal tract parameters which could be represented by AR coefficients. Because these coefficients are continuous values, they must be discretized so that they can be transmitted over a channel. Therefore, this chapter is essential for to the topic of speech coding as well. In speech coding, we are interested in minimizing the number of bits spent (or word length) because it is more cost efficient or because physical restraints require a limited bit rate. It is therefore in our best interest to investigate smart algorithms for coding so that the speech parameters can be represented by the lowest number of bits that preserve intelligible speech.

It is important to realize that while correctly performed sampling is lossless, quantization is not. The amplitude continuous signal cannot be reconstructed from the amplitude quantized signal. The goal of this section is to quantify the error that quantization introduces. We can then see the effect of bits spent on this quantization error and ultimately on the quality of the sound. To measure the quality of the quantized sound, we use the *signal to noise ratio (SNR)*, a ratio of signal power to noise power.

There are different ways to quantize a signal, and the most basic is to implement is *uniform quantization*. Uniform quantization means that the range of the input signal is predetermined and then divided into uniform quantization levels referred to as *step size*. We can then build a *quantization characteristic curve* that maps the input signal to a specified output. This can be seen in Figure 6.2 which portrays the uniform midrise quantizer that will be the focus of this chapter. There is also a mid tread which differs in the assignment of bits around zero, however the behavior of both quantizers is similar for large numbers of bits spent. The x-axis depicts the values present in the continuous input signal, and these values are then mapped to the discrete

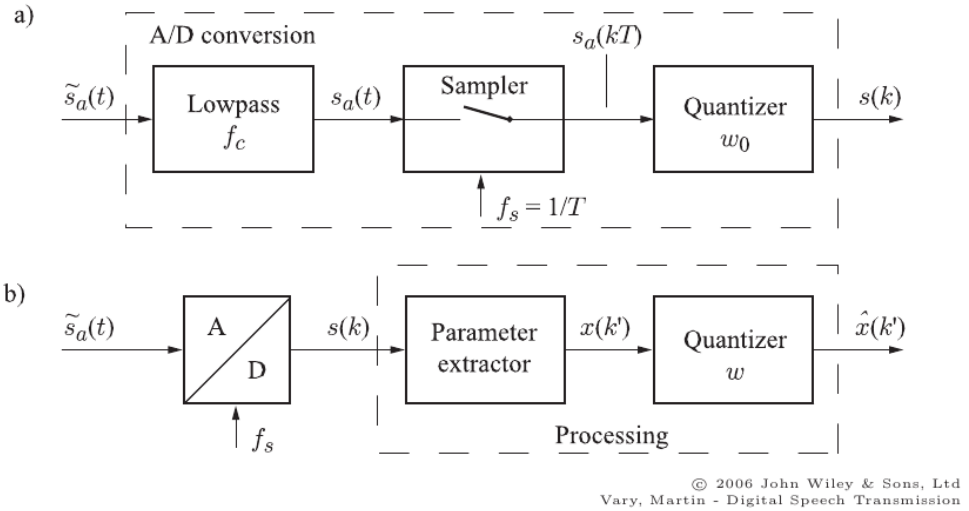


Figure 6.1: Top: Flow diagram of a waveform quantizer. Bottom: Flow diagram of a parametric quantizer. [8]

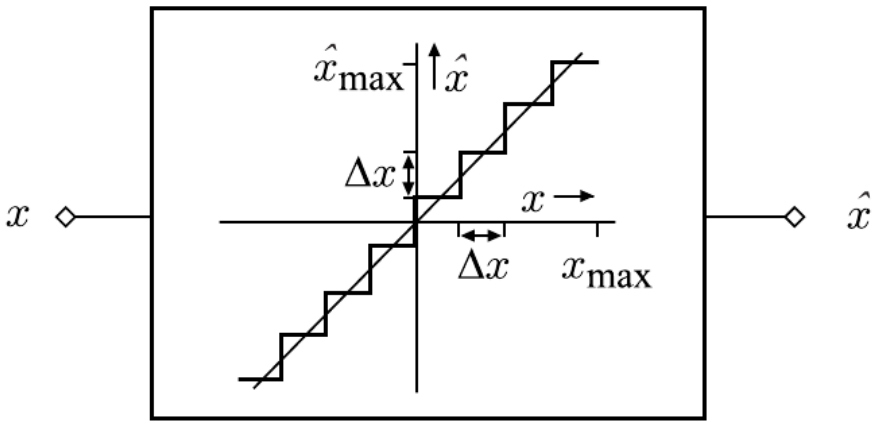


Figure 6.2: A midrise uniform quantization function. [7]

quantization levels denoted on the y-axis. The height between two quantization levels step size is called the *step size*, Δx. The maximum amplitude refers to half of the range that has been chosen for the input signal. This is symmetric about the origin, meaning that the signal can only contain values between $-x_{max}$ and x_{max} .

The number of quantization levels, and therefore the step size Δx is controlled by the number of bits chosen to digitize the signal. For a given number of bits, w, it is possible to represent 2^w quantization levels which can be seen in Figure 6.3. Typical word lengths used in the uniform quantization of speech are at about $12\,\text{bit}$ to $16\,\text{bit}$ and $16\,\text{bit}$ to $132\,\text{bit}$ for audio. Some applications such as ISDN telephony only use $8\,\text{bit}$, however this is done by using non-uniform quantization that will be discussed in the next chapter.

It is necessary to make some initial assumptions in order to derive some formulas that will help to understand the impact of quantization. Some of them are a bit rough at the start but they will turn out to be very useful in practice. We must first make the assumption that quantization error

$w = 3 \Rightarrow 2^3 = 8$ levels	
Δx_1	000
Δx_2	001
Δx_3	010
Δx_4	011
Δx_5	100
Δx_6	101
Δx_7	110
Δx_8	111

Figure 6.3: Digitization of quantization levels using three bits.

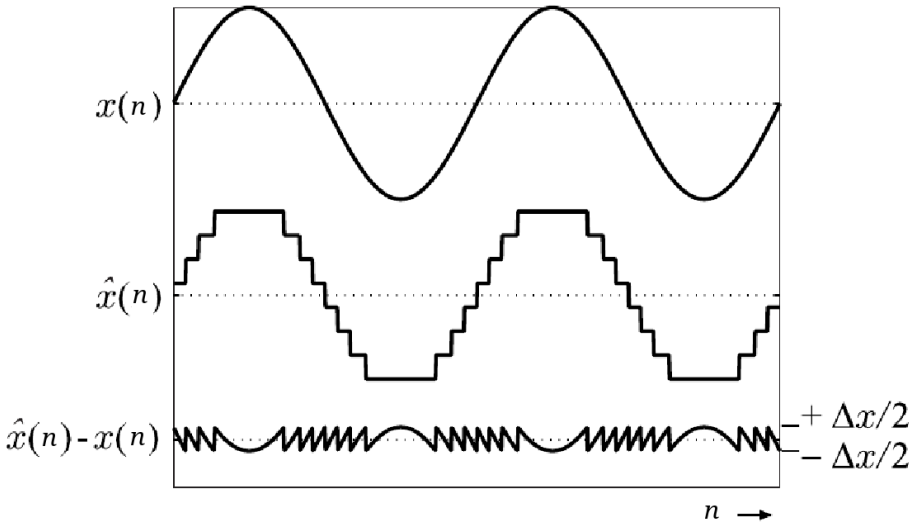


Figure 6.4: Top: Input sinusoidal signal. Middle: Quantized Signal. Bottom: Quantization error. [7]

is uncorrelated with the signal. This holds approximately when using a sufficiently large number of bits, however if the number of bits is low, then the derivations that will be made will no longer hold. This can be seen in Figure 6.4 in which the error signal is quite correlated with the original signal. However, one could imagine that this becomes less pronounced as the word length in increased and the quantization levels become finer.

In order to derive the SNR as a function of the bits spent, we create a model of the quantization process in which the output signal, $\hat{x}$ is the represented as the sum of the input signal x and an error signal e that will represent the noise added from the quantization process. Because a signal can always be represented as the difference of two other signals, the quantization error is now represented by $e = \hat{x} - x$. For large w, this error can be modeled by a white uniformly distributed noise signal that is not correlated to the original signal. This means that the expected value of the signal times the noise is equal to zero. Because both signals are zero mean, this can be written as $\mathrm{E}(xe)$.

In order to compute the SNR, we use its definition as the power of the signal over the power of the noise:

$$SNR = \frac{P_x}{P_n}$$

The power of a zero mean random signal is computed by taking the expected value of the squared random variable and then applying the definition of expectation:

$$P_x = \mathrm{E}(x^2) = \int_{-\infty}^{\infty} x^2 p_x(x) dx$$

Using the definition of the expectation value requires that we know the probability density function (PDF) of the signal. The simplest assumption is that the signal x is uniformly distributed thus resulting in a constant valued PDF. Because the range of our signal is between $-x_{max}$ and x_{max}, the constant value of the PDF must be $\frac{1}{2x_{max}}$ so that the area under the PDF is one. If the number of quantization levels is $k = 2^w$ then $x_{max} = \frac{k\Delta x}{2}$.

$$\begin{aligned} P_x &= \int_{-x_{max}}^{x_{max}} x^2 \frac{1}{2x_{max}} dx \\ &= 2\int_{0}^{\frac{k\Delta x}{2}} x^2 \frac{1}{k\Delta x} dx \\ &= 2\frac{1}{k\Delta x}\frac{1}{3}\frac{k^3\Delta x^3}{8} \\ &= \frac{k^2\Delta x^2}{12} \end{aligned} \tag{6.1}$$

The next step is to compute the power of the quantization error or the quantization noise. Once again, the noise power introduced from quantization is obtained by taking the expected value of the squared error term. So we utilize the same definition used to compute the signal power:

$$P_n = \mathrm{E}(e^2) = \int e(x)^2 p_x(x) dx$$

$$e(x) = \hat{x} - x$$

Once again it is necessary to derive a PDF, but this time it is for the noise. By looking at Figure 6.2, it can be seen that at the value of $\frac{\Delta x}{2}$, the quantized output value is equal to that of the input value and that the error is zero. The same also holds for $\frac{\Delta x}{2} + \Delta x$ and so on. Between these values, there is simply a linearly increasing error, thereby yielding an error signal that is the same between each quantization interval looking somewhat like a sawtooth signal. This now makes the process simpler, because the integration limits can be restricted between one quantization

interval and then simply multiplied by the number of intervals, k.

$$\begin{aligned} P_n &= k \int_0^{\Delta x} (x - \frac{\Delta x}{2})^2 \frac{1}{k\Delta x} dx \\ &= \frac{1}{3}\frac{1}{\Delta x}\left[(x - \frac{\Delta x}{2})^3\right]_0^{\Delta x} \\ &= \frac{1}{3}\frac{1}{\Delta x}\left[(x - \frac{\Delta x}{2})^3 + (\frac{\Delta x}{2}^3)\right] \\ P_n &= \frac{\Delta x^2}{12} \end{aligned} \tag{6.2}$$

This states that the quantization noise is purely a function of the step size Δx. This conclusion reveals that the noise power is only a function of the chosen step size. Typically the input signal will be scaled between -1 and 1, so the step size is determined by the number of bits used to describe the signal. In this case, it can be implicitly stated that the quantization error is dependent upon the word length.

Now that the signal and the noise power have been found, the computation of the SNR yields the nice result:

$$SNR = \frac{P_x}{P_y} = 2^{2w}$$

The result shows that the SNR is directly related to the number of bits w spent and becomes even more accessible when transformed into the decibel range.

Quantization SNR

$$SNR\Big|_{\text{dB}} = 10{\cdot}log_{10}\frac{P_x}{P_n} = 10{\cdot}log_{10}2^{2w} = 6\,\text{dB}{\cdot}w$$

Thus, for each bit spent, the SNR is increased by 6dB. An audio CD in which the audio is coded with $w = 16\,\text{bit}$, the approximate SNR is thus 96dB at maximum. However, this is only if the scaling of the signal is optimal meaning that the signal is uniformly distributed between $-x_{max}$ and x_{max}. When the input signal is not uniformly distributed, such as speech, or when the maximum values of the signal lie below the chosen x_{max} then the SNR will be lower than predicted. This can be seen in Figure 6.5.

To further examine this point, we think of a quantizer with a predetermined range between $-x_{max}$ and x_{max}, but we are attempting to quantize a signal with a PDF that lies well within this range. When the signal is quantized into different intervals, there is no part of the signal in the region between $-x_{max}$,x_{max}, and the boundaries of the PDF. These regions are therefore wasted and the signal is actually quantized into fewer intervals thus resulting in a worse SNR than predicted. Figure 6.5 displays SNR as a function of the ratio of signal power signal to x_{max}. We can see that the SNR increases with increasing signal power. It cannot go to infinity because then

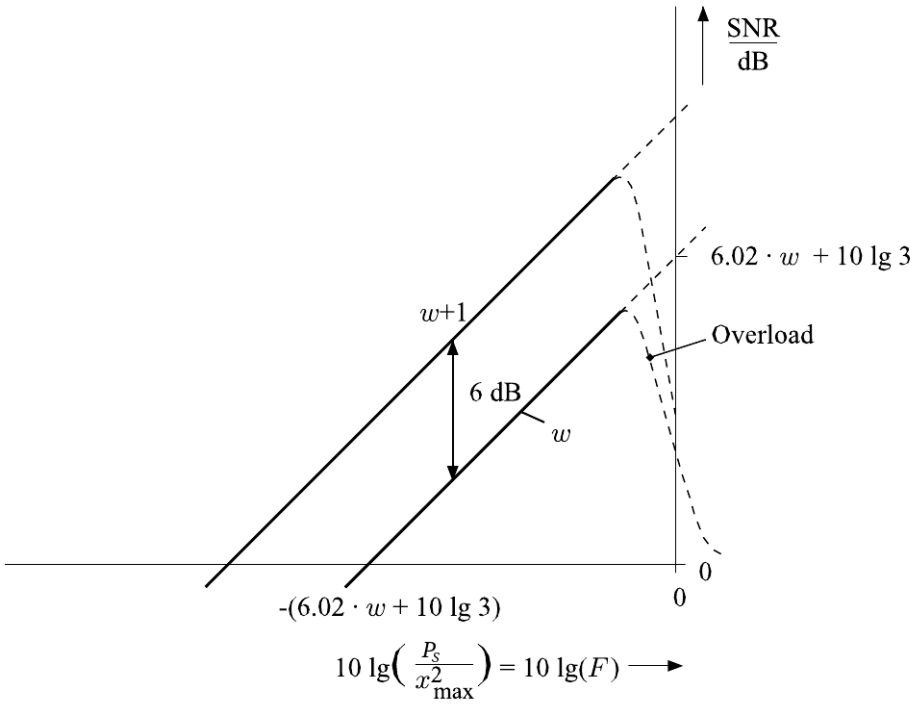


Figure 6.5: SNR is plotted as a function of bitlength. The form factor is also graphically introduced. [7]

we would be attempting to quantize a signal with a PDF whose boundaries lie outside $-x_{max}$ and x_{max}. Everything over $-x_{max}$ and x_{max}, would then be mapped to the last bit and the output signal would be distorted. This is called *overload* and it causes a drastic decrease in the SNR. Furthermore, Figure 6.5 shows the 6dB increase in SNR when one more bit is used in the quantization process.

The assumption that the signal is uniformly distributed is not fulfilled in practice for many signals. For instance, speech contains the majority of its values close to zero rather than x_{max}, therefore its distribution is better represented by a peaky distribution with heavy tails, such as the Laplacian or Gamma distribution. Using a uniform quantizer in this case will degrade the signal quality because it would be better to spend more bits on the signal amplitudes that appear most frequently. This can be taken into account by introducing the form factor F. This is the ratio of the signal power to the second power of the signal scale and would be about $F = \frac{1}{3}$ for a uniform distribution. However for speech, it could be on the order of $F = \frac{1}{300} - \frac{1}{20}$ meaning that the SNR would be significantly degraded. Figure 6.5 could also be viewed as a form factor to SNR function. The form factor is computed by taking the ratio of the speech power to the square maximum limit of the quantizer and this is also what is shown in the figure. As the signal approaches a uniform distribution, the form factor would increase along with the SNR.

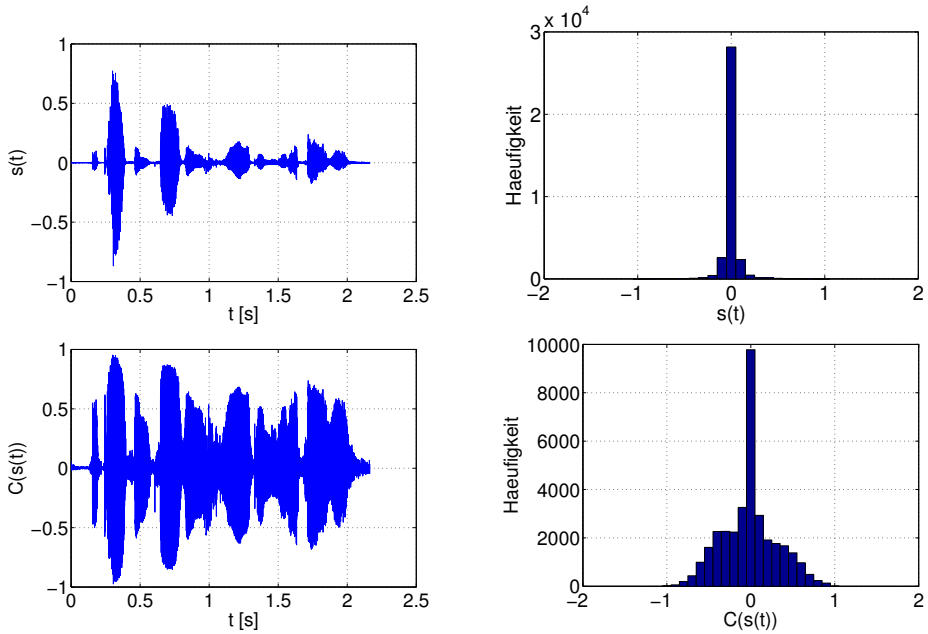


Figure 6.6: Top: Time domain speech signal and histogram. Bottom: Compressed speech signal and histogram.

6.2 Non-uniform Quantization

The previous section revealed that the SNR degrades significantly when the quantized signal does not have a uniform distribution. This degradation can be measured using the form factor, however this does not solve the problem of a low quality output signal. To solve this problem, non-uniform quantization methods are employed that are more suited to the distribution of the signal and ensure a higher quality output signal.

The top right of Figure 6.6 displays the distribution of a speech signal. It can be seen that a better quality quantization coding would be to utilize finer quantization for the lower values that occur more frequently and coarser quantization intervals for the large amplitudes that don't occur that often. This process is called *non-uniform quantization* and requires the definition of a different quantizer.

In practice, instead of defining a nonuniform quantization characteristic curve, a commonly employed trick is compress the signal and then stick to a uniform quantization and expand the signal afterwards to undo the compression. These methods, that are based on a compression and expension of the signal, are often referred to as *companding*. This effect is illustrated in Figure 6.6 in which the histogram of a speech signal and the histogram of the compressed speech signal can be compared. For the non-compressed histogram, there are lots of values around zero that may

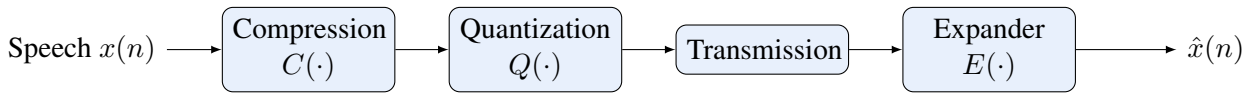


Figure 6.7: Flow chart of the non-uniform quantization process.

not be accurately coded using a uniform quantizer. However, after the compression, the dynamic range of the signal is reduced, and the histogram closer resembles a uniform distribution.

Figure 6.7 shows a flow chart of the basic process. The input signal $x(n)$ is passed through a compressor that reduces its dynamic range. The compressed signal is then uniformly quantized in the same manner that was introduced in the previous section. Using this method, the efficient coding of speech is dependent upon the compression step and in choosing the correct compression curve for the signal that is to be quantized. If the main application is telephony than the compression curve can be optimized for speech. This is done in ISDN telephony in which the A-law (Figure 6.8) is employed as the European standard and the μ law is employed as the American standard. It is important to keep in mind that the receiver must remove the compression, so that the speech is once again intelligible. The function used to perform this task is referred to as an *expander*.

Applying a logarithmic compression requires a function $C(x)$ that compresses the signal x. A simple logarithmic compression such as $\log x$ accomplishes this, however we have to keep in mind that the output of the compression function must lie in the range of $x_{\max}$ and $-x_{\max}$ so that it can be input to the uniform quantizer. Because the output of the logarithm function lies between $\pm\infty$, some modification must be performed by introducing constants such that $C(x_{\max}) = x_{\max}$.

$$C(x) = c_1 \log x + c_2$$

In order to force $C(x_{\max}) = x_{\max}$, the signal is scaled by $x_{\max}$. Now, when $x = x_{\max}$, then the ratio is one and the logarithm is zero. By adding $x_{\max}$ to this, then the goal is achieved. The upper limit is automatically applied in the next step by the midrise quantization curve.

$$C(x) = c_1 \log \frac{x}{x_{\max}} + x_{\max}$$

However, there is still a problem. When x goes to zero then the logarithm goes to $-\infty$. The A law solves this problem by using two solutions, one for large values and one for small values. These solutions are then applied in a way so that there is a continuous transition between the two. For smaller values, there is no logarithmic compression, only a linear input/output characteristic. The denominator of the two solutions corresponds to the constant c_1 and is a used for normalization. Note that $x_{\max} = 1$ meaning that the signal is normalized before the compression is applied.

$$C(x) = \begin{cases} \operatorname{sign}(x) \frac{1+\log(A|x|)}{1+\log(A)} & \frac{1}{A} \leq |x| \leq 1 \\ \operatorname{sign}(x) \frac{A|x|}{1+\log(A)} & |x| < \frac{1}{A} \end{cases}$$

The characteristic curve describing this function is displayed in Figure 6.8. The parameter A controls the amount of compression that is applied, so for $A = 1$ there is no compression. For larger

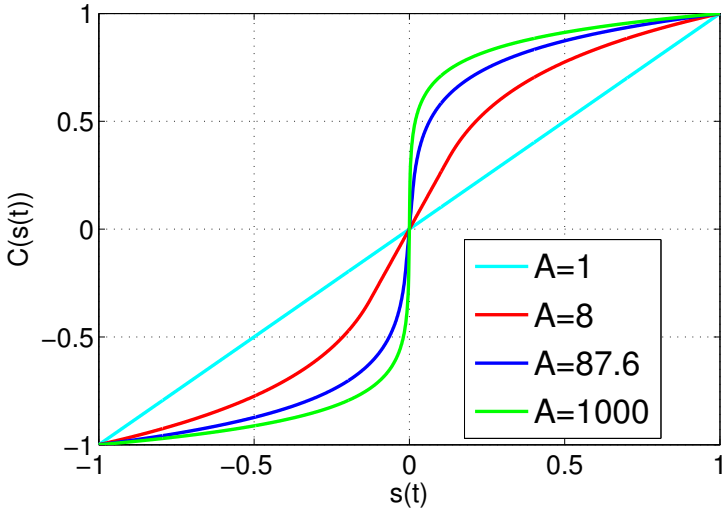


Figure 6.8: A-Law compander plotted with different values of A.

values of A, the signal is compressed meaning that values around zero will be amplified while larger values are compressed. A typical value that is chosen is $A = 87.6$ twhich is approximate to the standard value that is used in ISDN telephony in Europe.

Figure 6.9 is similar to Figure 6.5 shown in the previous section. The dashed line represents the form factor to SNR function obtained with an $8\,\mathrm{bit}$ uniform quantization encoding scheme. The solid line represents the same function using the same word length, but after utilizing a companding mechanism. There is a significant gain in SNR of $24\,\mathrm{dB}$ which would necessitate an extra $4\,\mathrm{bit}$ when using a uniform quantization. However, this linear increase only applies for range II, after which the there is a flat plateau for the larger signal powers, depending on the form factor. In range I , the gain decreases and then there is an area where it is more efficient to use a uniform quantization. This is the point there is a large form factor that is only achieved when the signal is close to uniformly distributed between $-x_{max}$ and x_{max}. The optimal quantization in this region would then be uniform.

In the previous section, it was said that the typical word length used to encode speech is on the order of $12\,\mathrm{bit}$ to $16\,\mathrm{bit}$. By utilizing non-uniform quantization techniques, the word length is effectively reduced to $8\,\mathrm{bit}$. A and μ law compression is commonly used in ISDN telephony in which a sampling frequency of $8\,\mathrm{kHz}$ is employed. This results in a bit rate of $64\,\mathrm{bps}$ which is $\%50$ lower than using a simple uniform quantization. However, it is important to keep in mind that this method is only effective when it is possible to predict the distribution of the input signal and build a compression curve accordingly. For encoding music onto a CD, there is no way to predict the type of signals that will be encoded, and so it is better to utilize a standard uniform quantizer.

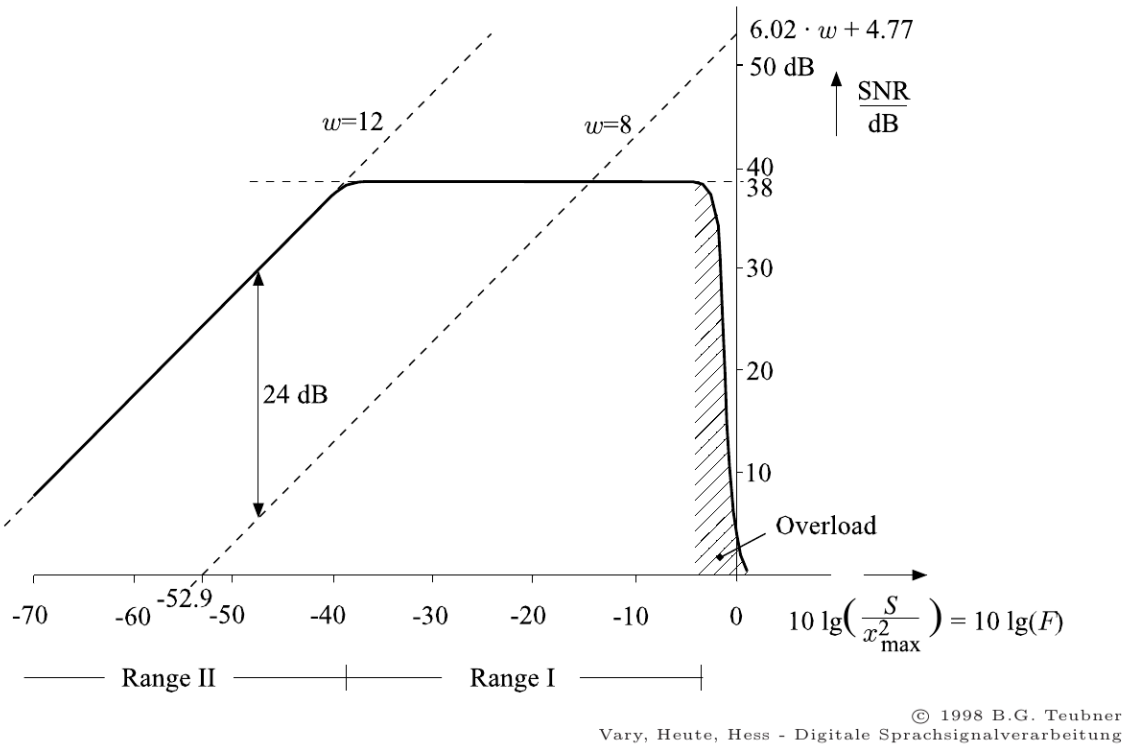


Figure 6.9: SNR as a function of wordlength after implementaion of a companding scheme. [7]

6.3 Adaptive Quantization

Non-uniform quantization applies a nonlinear compression to the amplitude domain of the signal, however this process is still uniform over time in that the same compression curve is applied to each time frame. We can begin to think of a compression curve that changes in time, that adapts to the range of the signal at each time frame. This process is known as *adaptive quantization*.

A simple way of implementing adaptive quantization would be to examine a signal at a certain time. If it is only scaled between -0.2 to 0.2, then the range of quantization would be adjusted accordingly. On the other hand, if its fully scaled between -1 to 1, the same number of quantization intervals would be used, however they would be larger. By adjusting the quantization characteristic curve over time with respect to the scaling of the signal, then the chosen $x_{\max}$ for each frames corresponds the boundaries of the signals PDF. As was show in the first section, this maximizes the SNR because there are no bits wasted on amplitudes that the signal does not contain.

Adaptive quantization generally utilizes a uniform quantizer that has k levels, but introduces a temporal adaptation of the step size Δx. Because the quantization noise power is $\frac{\Delta x^2}{12}$, the quantization noise power will also change over time if the step size Δx is a function of time. If a signal is loud, then the noise will also be loud, but the higher amplitudes help to mask this noise. When a softer signal is quantized within the same range as the loud signal, then the noise is not masked as well. Adaptive quantization overcomes this problem by adjusting the range of the soft signal so that the noise is lower and is masked more easily by the signal.

The adaptation step is commonly implemented with respect to the square-root of the signal power or the signal's standard deviation, σ. The step size can now be expressed as a function of time that is dependent upon the standard deviation and a chosen scalar.

$$\Delta x(n) = c\,\hat{\sigma_x}(n)$$

In order to quantize the signal, we would need to transmit some quantization index $Z(n)$ and quite possibly the quantization step size $\Delta x(n)$.

$$\hat{x}(n) = \text{sign}(x)\; Z(n)\; \frac{\Delta x}{2}$$

For a midrise characterization curve, $Z \in \{1, 3, 5, ...\}$.

There are different variants of this process, and they generally differ in how they estimate the standard deviation, σ_x. The two basic solutions are referred to as *adaptive quantization forward (AQF)* and *adaptive quantization backwards (AQB)*.

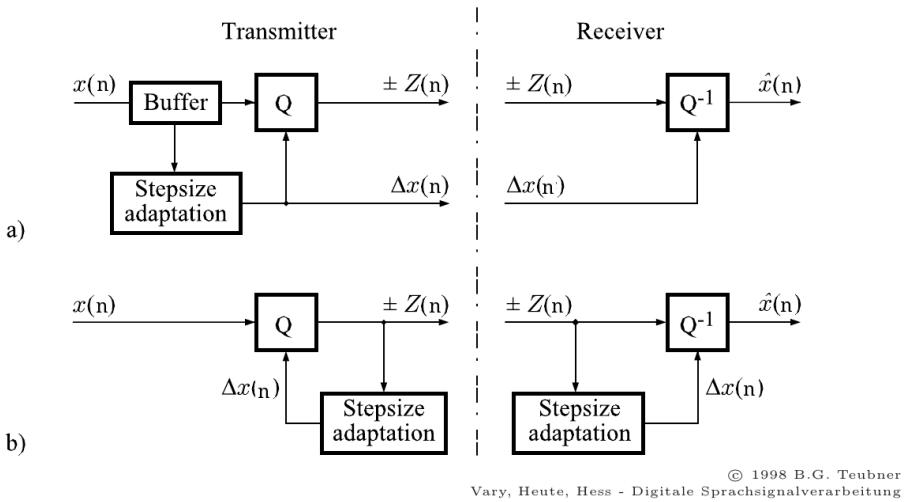


Figure 6.10: (Top) Adaptive quantization forward. (Bottom) Adaptive quantization backward. [7]

AQF is a method in which the variance is estimated by taking the mean of the squared non quantized signal samples over a sliding window of length $N-1$ meaning that $\hat{\sigma_x}$ is estimated on time blocks of length N samples.

$$\hat{\sigma_x^2}(n) = \frac{1}{N}\sum_{m=0}^{N-1} x^2(n+m)$$

Because Δx is a function of $\hat{\sigma_x}$ it is necessary to not only transmit the quantization index but also the chosen step size, Δx or the standard deviation, $\hat{\sigma_x}$. This can be seen in the upper left of Figure 6.10. The input signal, $x(n)$ is buffered for N samples and then the signal power and standard deviation is computed. This is used to determine the step size Δx which is then fed into the quantizer.

AQF improves sound quality by improving SNR during softer sections of the sound, however it is necessary to transmit the step size Δx and this requires more bits to be transmitted. One of the main goals of speech coding algorithms is to minimize this number of transmitted bits in order to minimize costs and increase efficiency. This is achieved by the use of adaptive quantization backward (AQB), a process in which the step size adaptation, a function of the signal power, is computed as a function of the quantized signal. Because the quantized signal is used, the same process can be performed on the receiver side which has access to current and past samples of the quantized signal. This has the very nice property in that it saves on bit rate as the step size does not have to be transmitted. Figure 6.11 displays the German word ”das” along with step sizes as a function of time computed by AQB and AQF methods. The AQF method can be seen to very coarsely track changes in the signals energy. The process is so slow because a change in the quantization step size requires the transmission of this information to the receiver. Because AQB adapts the quantization step size by using the quantized signal as opposed to the signal itself, it can more quickly track these changes of energy in the signal.

AQB utilizes recursive smoothing to estimate the signal power. This means that the standard deviation for the current sample is calculated by a weighted average of the previous standard

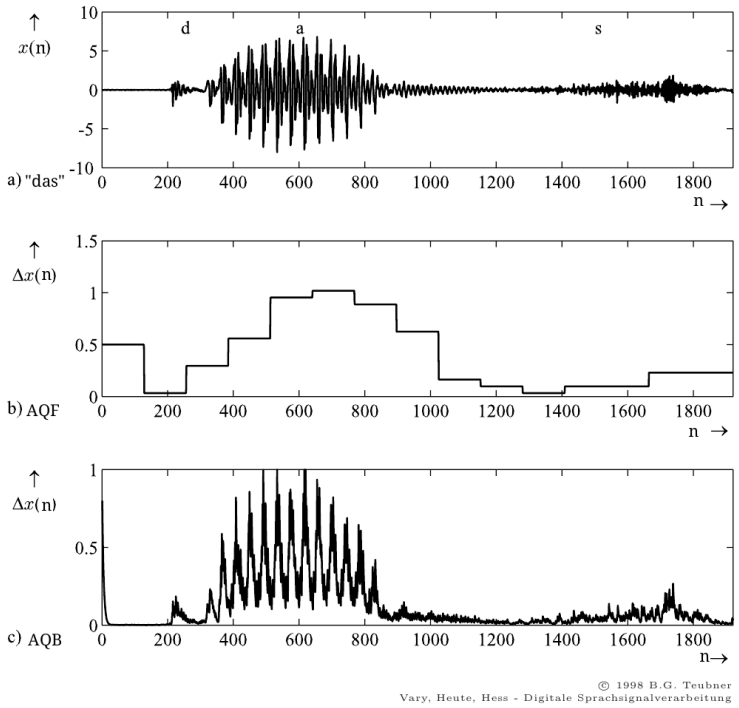


Figure 6.11: (Top) Time domain representation of the German word ”das”. (Middle) The quantized time domain signal using an adaptive quantization forward scheme. (Bottom) The quantized time domain signal using an adaptive quantization backward scheme. [7]

deviation and the power of the current signal. The average is weighted by using the smoothing factor α.

$$\widehat{\sigma}_{\hat{x}}^2(n) = \alpha\widehat{\sigma}_x^2(n-1) + (1-\alpha)\hat{x}^2(n-1), \quad 0 < \alpha < 1$$

If $\alpha \approx 1$, then there is a heavy smoothing in which the new estimation mostly relies upon the previous estimate while the current value does not have a strong influence on the signal. The opposite is true if $\alpha \approx 0$.

Figure 6.12 displays the local power of a speech signal as a function of time and the SNR as a function of time for different quantization methods. It can be seen that whenever the signal power is low, for instance in the word ”also”, then the SNR for the uniform quantization degrades. Because the quantization power is constant over time, the noise power becomes more audible as the signal power drops and therefore the SNR also degrades. This problem is not so present with the A and μ laws, however there is an effect on the SNR in parts where there is very little signal power. The bottom of Figure 6.12 reveals the benefits of utilizing an adaptive quantization in which there is a much higher SNR over the course of the entire speech sample.

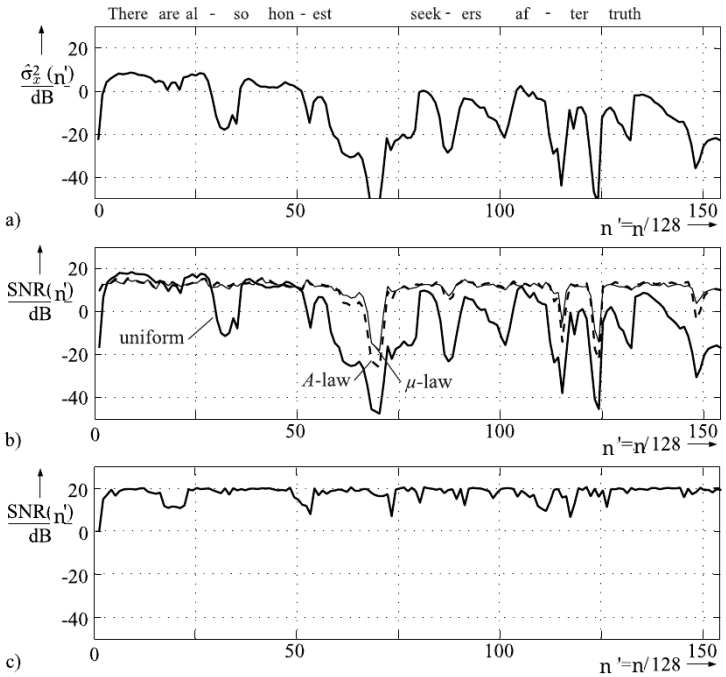


Figure 6.12: (Top) Power of a spoken speech sentence as a function of time. (Middle) Comparison of uniform / A-law / μ-law quantization methods. (Bottom) AQF. [7]

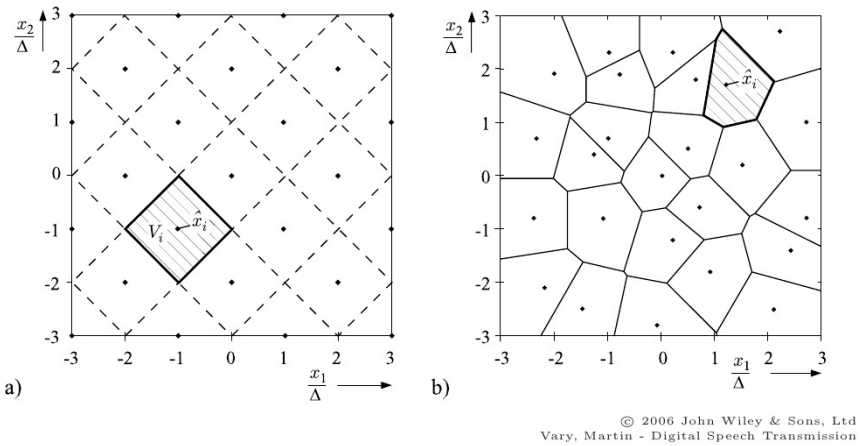


Figure 6.13: Vector quantization using 25 vectors with dimension = 2 and using both (left) uniform and (right) non uniform resolutions.[8]

6.4 Vector Quantization

All quantization methods that have been discussed thus far have dealt with quantizing each time sample independently. For an input signal, $x(n)$, this means that we have only dealt with quantizing $x(n_0)$ for a given n_0. Vector quantization deals with simultaneously quantizing multiple random variables (RV). This is used in the transmission of LPC coefficients where there are generally ten coefficients per time frame. In this case, all coefficients would need to be simultaneously coded and transmitted as a ten dimensional quantized vector. The applications also extend beyond speech coding to areas such as vector classification in general and unsupervised learning.

We first define a vector, $\hat{x}$, that is comprised of L random variables.

$$\hat{x} = [x_1\ x_2\ ...\ x_L]^T$$

$\hat{x}$ could represent a block of time domain samples, LPC coefficients, or even variants of it such as cepstral coefficients. These vectors correspond to centroids stored in a codebook to which both the sender and the receiver have access. This process saves on bit rate because it is only necessary to transmit the codebook index. In a sense, bit rate is exchanged with memory, because the receiver needs to store all possible codebooks to which the index corresponds. We will see that the process exploits the correlation between the successive samples to achieve an improved quantization and higher signal quality.

Figure 6.13 shows a two dimensional example of the simultaneous quantization of two random

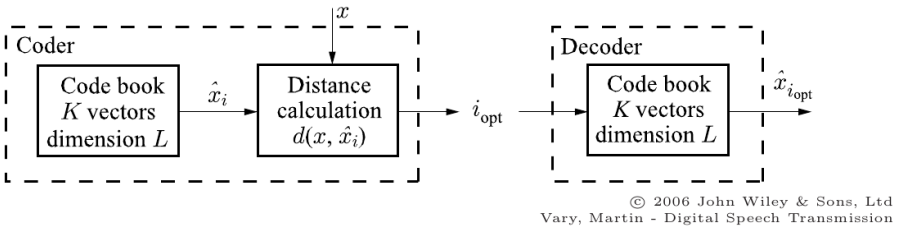


Figure 6.14: Flow chart depicting the process of vector quantization. [8]

variables, x_1 and x_2. Input values of the two continuous random variables correspond to a specific location on the plot. This location falls within an area, known as a Voronoi cell, that is bounded by predefined quantization boundaries. The center point of these areas are referred to as centroids, and these points represent the quantization of the multi-dimensional signal. In this two dimensional case, the quantization levels that were introduced for the scalar case are visualized as the areas of these cells. In the three dimensional case (a vector of three random variables), these levels would be visualized as volumes. Vector quantization generally employs higher numbers of dimensions, which make the process difficult to visualize, but the principle is still the same. The equivalent areas of the cells in the left of Figure 6.13 are called a uniform resolution and are analogous to the uniform quantization introduced for the scalar case. However, the cells can have arbitrary shapes such that the shapes are optimized to the signal statistics. This is referred to as non-uniform resolution and is beneficial when successive samples of the vector are correlated such as in speech.

Figure 6.14 displays a flow diagram of the vector quantization process in which both sender and receiver have copies of codebooks that correspond to the centroids of the chosen quantization scheme. By utilizing this method, the number of bits spent is minimized because the sender must only transmit the index of the centroid corresponding to the current quantized vector. This is how vector quantization trades memory for bits spent, however there is another trade-off associated with the process. The sender must first find the correct cell for an input vector, and this must be done by performing a search, which requires computational time. One method of doing this is by choosing some distance measure and searching for the centroid that has the smallest distance to the input vector. The optimal codebook index, i_{opt} would then be found by minimizing a distance measure, i.e. Euclidean, between the input vector, $\hat{x}$, and the codebook vectors corresponding to centroid locations, $\hat{x}_i$.

$$i_{\text{opt}} = \arg\min_i \; d(\hat{x}, \hat{x}_i)$$

$$\text{Euclidean distance:} \quad d(\hat{x}, \hat{x}_i) = \sqrt{\sum_{m=1}^{L} (x_m - x_{i_m})^2}$$

A codebook can be defined as a set of quantized centroid vectors. Each vector is then assigned a binary index. This is shown in Figure 6.15 which chooses arbitrary values for the centroid locations.

i	$w = 2$	$(x_{i,1}, x_{i,2})$
1	00	(0.1, 0.9)
2	01	(0.2, 0.8)
3	10	(0.3, 0.7)
4	11	(0.4, 0.6)

Figure 6.15: Digitization of codebook indicies corresponding to centroid locations.

The number of centroids that can be represented is then dependent upon the number of bits. We can now compute the number of bits required to quantize an L element vector with a quantization scheme that utilizes k centroids.

$$w = \log_2 k \text{ bits}$$

$$\hat{w} = \frac{1}{L} \log_2 k \, \frac{\text{bits}}{\text{vector element}}$$

As we saw with non-uniform quantization, codebooks can be trained for optimal performance based upon the type of signal for which they will be used. Figure 6.16 is a codebook created for a speech signal in which two successive samples, $s(n)$ and $s(n+1)$, are quantized using a two dimensional vector quantizer. Looking only at the one dimensional distribution of $s(n)$ on the x-axis reveals a high concentration of centroids about the origin which is to be expected knowing that speech is not uniformly distributed, but highly distributed about the origin. It can also be observed that a linear curve fit of the centroid distribution would be approximately a $45°$ positively sloped line. This means that there is a high correlation between two successive samples of speech. Training the vector quantizer exploits these correlations to optimize the quantization levels and provide a higher sound quality.

The codebook is trained by means of iterative algorithms such as the Linde-Buzo-Gray or the K-means. The first step of the process is to decide on the number of centroids and therefore the number of bits that will be used for the quantization process. Based upon the application, training data is then obtained from a database or created artificially in the form of predetermined number of training vectors. The algorithm must then be initialized by deciding the pre-training distribution of the cells. They could, for example, be uniformly or randomly distributed. The chosen distance measure is then employed to assign each training vector to the closest corresponding centroid, also known as the nearest neighbor. For all the vectors falling within a cell, the center of gravity is computed and this center of gravity replaces the old centroid. The result is a shifting of the centroid and a reorganization of the quantization cells. For a new repetition of the process, two training vectors that fell within the same cell on the first iteration, may now fall within different cells. This is the process that is then repeated iteratively until some criteria are fulfilled, such as the average distance to the centroids being less than some predetermined value.

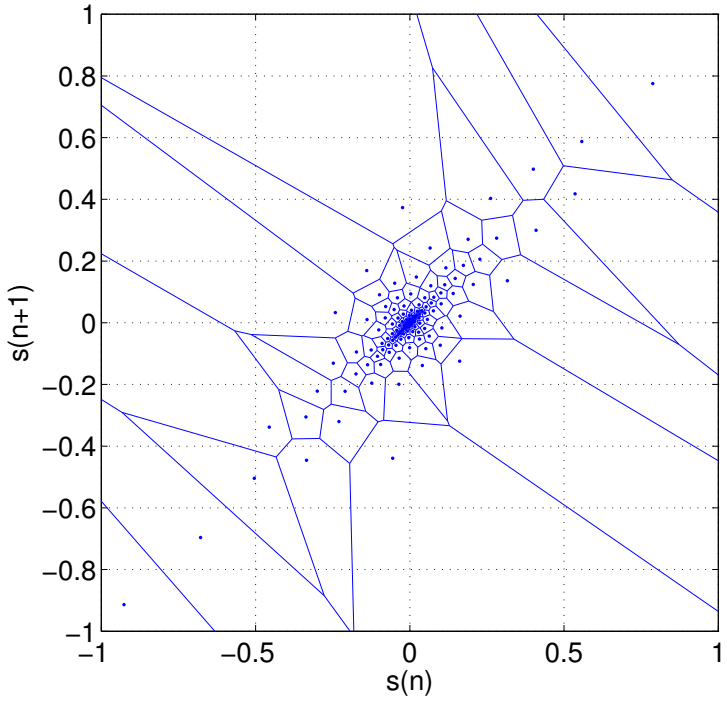


Figure 6.16: $2^7 = 128$ Voronoi cells for neighboring speech samples after K-means clustering

7 — Speech Coding

Learning objectives

- Codecs
- Waveform coding
- Parametric coding
- Hybrid coding
- Perceptual coding

7.1 Codecs

One of the main goals of quantization is the transmission of data between a sender and a receiver. For cost and quality reasons, it is often beneficial to minimize the amount of data that is transmitted by inventing new methods by which the signal can be coded. This is done by finding algorithms called *codecs* that compress the data in order to enable faster transmission while still maintaining intelligibility of the decompressed signal after it is received. As will be shown, there are different methods of performing this task that often involve exploiting a priori knowledge of the signal in order to minimize redundancy.

Different algorithms correspond to different data transmission rates, measured in kilobits per second, kbps. Text has a very low bit rate of about $50\,\mathrm{bps}$ whereas speech signals can be in the range of $2\,\mathrm{kbps}$ to $256\,\mathrm{kbps}$. The multitude of different sounds in music requires a considerably larger bit rate between $32\,\mathrm{kbps}$ to $1152\,\mathrm{kbps}$. For instance, for an audio CD, a sampling rate of $44.1\,\mathrm{kHz}$ and 16 bit per sample are used, resulting in $705.6\,\mathrm{kbps}$ per channel. In studios often a sampling rate of $48\,\mathrm{kHz}$ and 24 bit per sample are used resulting in $1152\,\mathrm{kbps}$ per channel. Using perceptual coders, like mp3 and its successors, allows to reduce the bitrate roughly by a factor of ten. In perceptual coding, a priori knowledge about the perceptual aspects of hearing are exploited to adapt the quantization to mask the quantization noise. This model of perception is employed because the types of signals in music have a high variability, meaning that a source based compression, like in speech coding, is difficult. The majority of this chapter will deal with speech where we already have quite a bit of a priori knowledge of the signal. These are the two ways that a priori knowledge can be used to compress data and streamline audio codecs. Either we exploit how a sound is perceived (i.e. mp3 coding), or we can exploit how it is produced (e.g. LPC coding).

There are mainly three different kinds of coders implemented for speech coding and these are represented in Figure 7.1. Waveform coding involves the direct quantization of the time domain waveform or some filtered version of it. The transmitted signal is then re-synthesized at the receiver by applying an inverse filtering in order to recover the signal. In contrast, a parametric coder transforms the signal from the time domain into another domain in which it can be represented by corresponding parameters. A typical example of the parameterization would be an LPC analysis, whereas the synthesis would be analogous to Dudley's Voder. Hybrid coders combine elements of both waveform and parametric coding in an attempt to achieve the low bit rate of parametric coders, but at the same time, produce the higher quality speech associated with waveform coders. There typically based on a concept called *analysis-by-synthesis*. A common problem with parametric coding is that the artificially created excitation signal can produce artifacts that produce speech sounding mechanic or robotic. Hybrid coders attempt to solve this problem by performing a LPC analysis in addition to a waveform coding of the residual signal.

Figure 7.2 shows several standardized codecs along with the bit rate and sound quality that

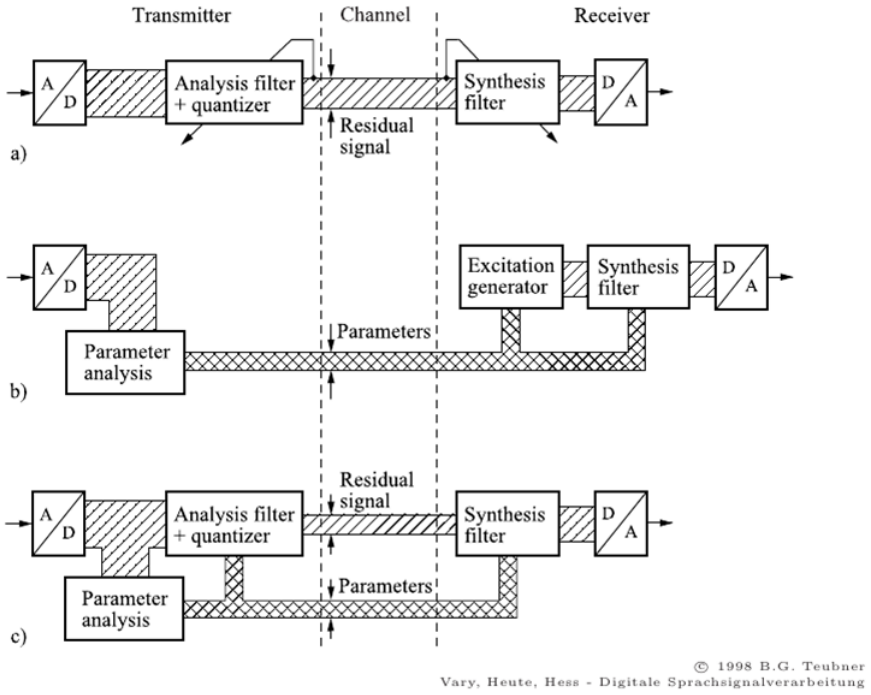


Figure 7.1: Flow charts depicting the processes of (top) waveform coding, (middle) parametric coding, (bottom) hybrid coding.[7]

corresponds to each one. As the bit rate along the x-axis is increased, so do costs associated with storage and transmission. The y-axis displays the results of MOS subjective sound quality tests meaning that the goal for a codec would be to lie in top-left corner of the plot. The standard PCM coding scheme, typically implemented on audio CDs, falls in the region of highest bit rate and highest sound quality. Next to it falls the log PCM method which is a companding scheme analogous to the A or μ law. The last of the waveform coders is an adaptive quantization implementation of PCM, referred to as ADPCM and having similar, however a bit lower, quality. The hybrid coders lie in the middle of the chart as a compromise between signal quality and bit rate. This is the reason that the GSM hybrid coder is the codec that is employed in cellular telephony. The parametric coders offer the lowest bit rate, however compromise sound quality.

Traditional telephony speech exhibits an audio bandwidth limited between $300\,\mathrm{Hz}$ to $3400\,\mathrm{Hz}$ for historic reasons and to save datarate. Because this standard has been used for a very long time, people have gotten used to the poor quality of speech and even though it is no longer necessary, this very limited bandwidth is still frequently used. Voice over IP (VoIP) is an example of a low bit rate speech coder that has a much higher quality than traditional telephony, as the bandwidth available for speech coding is dependent on the internet connection and not traditional values. Also on mobile telephones, HDvoice is available nowadays, where the frequency range is essentially doubled. However, first, this requires both parties to have a carrier and telephone that supports HDvoice, and second, the quality of the call is reduced adaptively, e.g. when the network is busy or the connection is poor.

ISDN is a standardized codec that is implemented in digital telephony and allows for the transmission of voice and data. The sampling frequency is $8\,\mathrm{kHz}$ and the bandwidth is somewhat

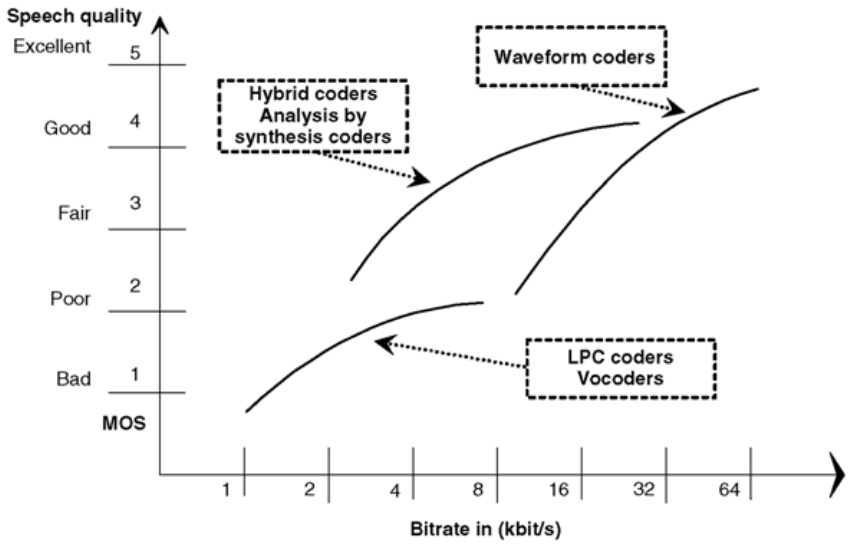


Figure 7.2: MOS of different coding algorithms plotted as a function of bit rate.

the same as in historic analog telephony. As opposed to a standard PCM coding scheme (uniform sampling and uniform quantization), the uniform quantization is applied in the compressed domain (e.g. A and μ law) which allows to only use $8\,\mathrm{bit}$ per sample for quantization. This, coupled with $f_s = 8\,\mathrm{kHz}$, results in a $64\,\mathrm{kbps}$ bit rate. For mobile telephony even lower bitrates are employed. This means that codecs even more efficient than the A and μ companding schemes are used so that quality is not sacrificed too much. Codecs for wideband telephony typically have a sampling rate of $16kHz$ and greater. Therefore, it is theoretically possible to restore audio frequencies up to $8\,\mathrm{kHz}$, however the usage of an anti-aliasing filter puts this value at about $7kHz$. This allows the user to distinguish between sounds like [s] and [f], a task that is not possible with traditional narrowband telephone speech.

The difference in speech quality can be measured using the mean opinion score (MOS) test. Listening experiments are conducted in which people are asked to rate sound quality of a codec on a scale between 0 and 5. The MOS results for different codecs can be seen in Figure 7.2 in which are a 5 on the y-axis corresponds to excellent quality sound reproduction and a 0 is bad quality. The experiments make use of standardized schemes to give subjects a basis of comparison. The Mushra test asks subjects to compare the codec to an anchor and a reference.

Complexity is another important characteristic of speech codecs. Modern computational abilities have become so powerful that our hand held phones easily outperform computers from ten years ago. Although it is possible to employ more and more complex algorithms, manufacturers are still extremely interested in keeping the complexity as low as possible. Higher complexity corresponds to increased chip area and higher power consumption. These are two big factors in a market driven by product size and battery life. Therefore, it is always desirable to come up with efficient solutions.

Processing delay is a very crucial characteristic of an algorithm that can affect the ability to communicate using the codec. This delay must definitely be kept under $500\,\mathrm{ms}$, otherwise the conversation is severely disturbed and changes the way by which we communicate (e.g. walkie talkies). This delay often becomes evident when using VoIP clients where the processing delay

is dependent on the transmission line.

Speech coders must also be robust with respect to disturbances, and therefore it is important to look at sensitivity when choosing a particular codec. Different algorithms handle environmental noise and transmission errors differently. LPC analysis is quite susceptible to external noise, and these disturbances result in artifacts in the resynthesized speech. The perceived quality of the codec could be greatly reduced depending on how these artifacts factor into the final SNR.

In order for some devices to communicate over the same network, it becomes more efficient and often necessary to use the same codec for data exchange. Therefore, many codecs are standardized and there are organizations that work with industry to further develop and set these standards. The telecommunication section of the International Telecommunication Union (ITU-T) is one example of an international agency that standardizes speech codecs; another is the European Telecommunication Standard Institute (ETSI).

7.2 Waveform Coding

In its simplest form, waveform coding would be the application of some quantizer to the original time varying signal. Efficient quantization methods such as non-uniform and adaptive quantization can be employed to reduce the required number of bits per sample. However because waveform coding does not employ a parametric description of the signal to be quantized, it remains the coding scheme requiring the highest bit rate.

Methods, such as differential pulse code modulation (DPCM), have been developed to limit dynamic range of the signal in an effort to produce more efficient methods of waveform coding. DPCM decorrelates the signal by subtracting weighted successive time samples from each other, e.g. $x(k) - ax(k-1)$ in the simplest case. Figure 7.3 shows a hypothetical input signal, $x(k)$, and the resulting waveform from subtracting successive weighted samples. It can be seen, that the dynamic range of the differential signal is smaller than for the original signal, which allows for using less bits/sample for quantization. The differential output signal can also be thought of as the output of an order one filter (one coefficient, a), which is similar to an LPC analysis with only one coefficient. However, similar to LPC analysis also multiple coefficients can be used and stacked to a vector $\mathbf{a}$. Like in LPC analysis, successive speech samples are decorrelated thus whitening the signal and reducing dynamic range that must be quantized to produce an intelligible SNR. The main difference between DPCM and LPC analysis is that the excitation signal in LPC analysis is parametrized, whereas DPCM quantizes and transmits the time varying post filtering residual. The savings in bitrate is also less than in the LPC coding, while the quality is higher. DPCM is a frequently used coding scheme, not only in audio, but also in video coding.

For this process to function, the receiver must have access to these coefficients $\mathbf{a}$ so that the coded signal can be inverse filtered in order to produce an intelligible output signal. When these

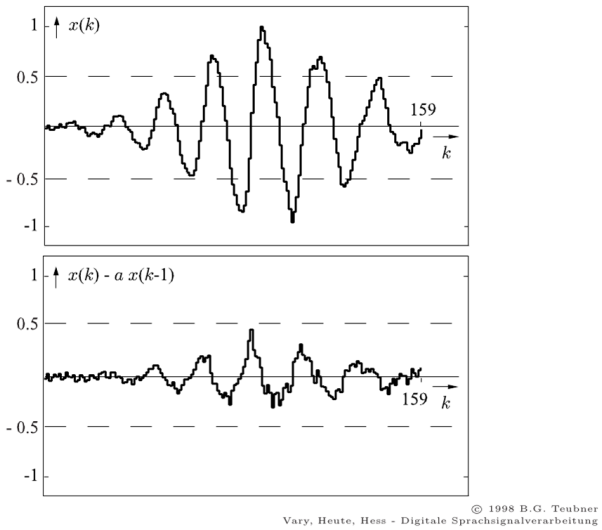


Figure 7.3: A windowed sinusoidal signal is plotted before (top) an implementation differential pulse code modulation and after (bottom) The process creates a new waveform of successive weighted sample differences. [7]

coefficients are adapted to successive windowed time frames, they must then be transmitted along with the quantized residual. This process is called an open loop prediction scheme. In the closed-loop setting in the bottom of Figure 7.4, the filter coefficients are computed from the quantized signal meaning that the coefficients computed by the receiver will be the same as those computed by the sender (assuming no transmission errors). Because the coefficients do not need to be transmitted, more coefficients can be used to further limit the dynamics of the signal and, ideally, completely whiten the signal. This concept is implemented in cordless phones to substantially reduce the bit rate to $32\,\mathrm{kbps}$, which is $\%50$ of the $64\,\mathrm{kbps}$ used in the log PCM scheme.

Even thought the closed loop system is preferential to the open loop system, because it does not require transmission of the filter coefficients, there are still some advantages to an open loop

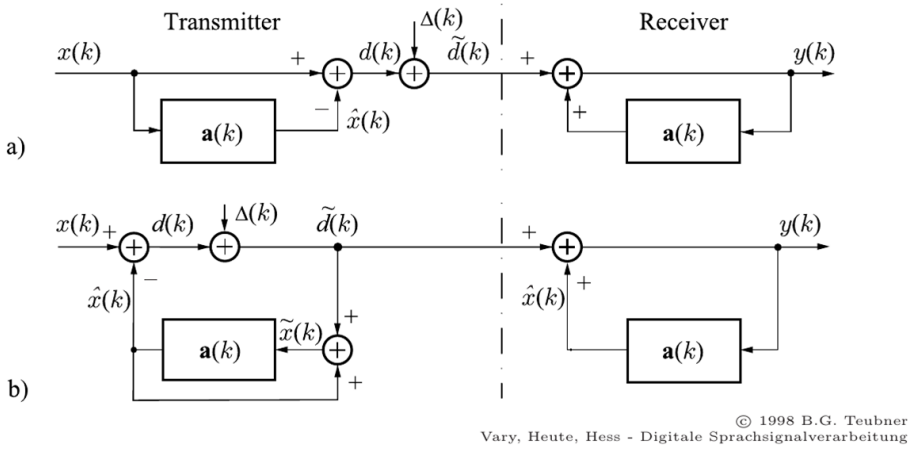


Figure 7.4: Flow charts comparing the (top) open and closed (bottom) loop prediction schemes. [7]

system that can be seen by performing an analysis in the z-domain. Looking at the open loop coding scheme in the top of Figure 7.4, the quantized difference signal, $\widetilde{d}(k)$, can be formulated in the z-domain as the sum of the difference signal, $d(k)$, and some quantization noise, $\Delta(k)$.

$$\widetilde{D}(z) = \Delta(z) + D(z)$$

We would like to express the difference signal $D(z)$ in terms of the input signal, $X(z)$, which is simply filtered and then subtracted from itself. This filtering corresponds to a multiplication in the z-domain. Upon substitution, the expression can then be further simplified into an equation representing the output of the transmitter and the input of the receiver.

$$\begin{aligned} \widetilde{D}(z) &= \Delta(z) + X(z) - X(z)A(z) \\ &= \Delta(z) + X(z)(1 - A(z)) \end{aligned} \tag{7.1}$$

The output of the receiver, $y(k)$, can be expressed as the sum of a filtered version of itself and the receivers input.

$$\begin{aligned} Y(z) &= \widetilde{D}(z) + A(z)Y(z) \\ &= \frac{\widetilde{D}(z)}{(1 - A(z))} \end{aligned} \tag{7.2}$$

We can now substitute the expression for $\widetilde{D}(z)$ and simplify in order to achieve this interesting result.

$$Y(z) = \frac{\Delta(z)}{1 - A(z)} + X(z)$$

As expected, the output of the receiver is the original signal plus some quantization noise. The interesting result is that the quantization noise is filtered, a property that is beneficial to the sound quality of the output signal. Because $A(z)$ was computed from the speech signal itself, it shapes the quantization noise $\Delta(z)$ with the spectral envelope of the speech signal. The noise and the speech now have the same spectral form, meaning that the quantization noise will be well masked by speech. This process is called noise shaping and results in the quantization noise becoming less disturbing resulting in a better quality output.

In order to compare the two systems, we now derive the output of the closed loop coding scheme in the bottom of Figure 7.4. The closed loop differs from the open loop in that the signal is filtered after quantization. Therefore, $\widetilde{d}(z)$ consists of the influence of the difference between $x(k)$ and its filtered version, $\hat{x}(k)$, and the quantization noise, $\Delta(k)$. The system can be mathematically reduced to:

$$\widetilde{D}(z) = X(z)(1 - A(z)) + \Delta(z)(1 - A(z))$$

The expression for the output, Y, is the same as for the open loop because the receivers for both the open and closed loop systems have the same structure.

$$\begin{aligned} Y(z) &= \widetilde{D}(z) + A(z)Y(z) \\ &= \frac{\widetilde{D}(z)}{(1 - A(z))} \end{aligned} \tag{7.3}$$

Substituting in the expression for the closed loop transmitted signal, $\widetilde{D}(z)$, yields the result:

$$Y(z) = \Delta(z) + X(z)$$

By using the closed looped method of coding, we see that, in contrast to the open-loop implementation, the quantization noise is re-synthesized perfectly in the output signal. However, closed loop coding is still the standard that is used for cordless land line telephones by the digital European cordless telephone (DECT) standard. So here is a case in which a lower bit rate and the consequential power consumption savings outweigh the loss of quality. It also reveals that straightforward mathematical derivations applied to engineering problems can save money and resources in the real world.

7.3 Parametric Coding

In a parametric coding scheme, a speech production model is employed and described by few parameters, such as coefficients describing the vocal tract filter, voiced/unvoiced information, fundamental frequency, and power. The parameters of this model are adapted for successive speech frames, coded, and transmitted. An example of a speech production model would be the LPC vocoder shown in Figure 7.5. An impulse generator creates impulses with a distance of the fundamental period length in order to model voiced speech, whereas unvoiced speech is modeled by using a noise generator. The LPC-10 model must transmit one bit that chooses between the two types of speech while the mixed excitation linear prediction (MELP) method creates a mixed excitation by means of a weighted average of the noise and impulse train. A gain parameter then controls the amount of energy that is in the signal. The adjusted excitation signal is then filtered through a variable vocal tract filter that is represented by filter coefficients transmitted for the current windowed speech segment.

The LPC-10 gets its name from the fact that it transmits ten LPC coefficients. This method also parameterizes windowed speech segment with a block length of of $22.5\,\mathrm{ms}$ at a sampling rate of $8\,\mathrm{kHz}$ resulting in a even number of $180\,\mathrm{samples}$. This was standardized after experimenting with different frame lengths and deciding which value yielded the best trade off with respect to

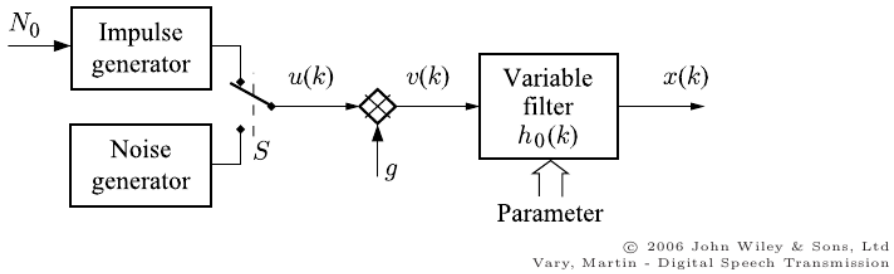


Figure 7.5: Flow chart depicting a LPC vocoder. [7]

quality and bit rate. Sometimes, the bit rate is the limiting factor, and we are forced to design a codec accordingly. An example is GSM telephony, in which data is transmitted over a channel with a predefined bit rate. It now becomes necessary to design a codec that has optimal quality and reaches this specific bit rate by adjusting the initial values that have been given the parameters of the model. This could be the block length, or even the number of LPC coefficients that are transmitted.

The LPC coefficients produced from a typical LPC analysis are autoregressive (AR) filter coefficients. A problem that comes with transmitting LPC coefficients is that the dynamic range is rather large and also not limited. This makes a quantization of LPC-coefficients difficult. However, we know that LPC coefficients can also be represented in different domains such as the reflection coefficients of the tube model. These are directly related to the LPC coefficients and can be computed one from each other using a simple algorithm. In fact, the Levinson Durbin recursion, used to compute the AR coefficients automatically, computes the reflection coefficients at the same time. The benefit of quantizing the reflection coefficients as opposed to the AR coefficients is that they, by definition, fall within an interval between-1 to 1.

$$\text{Reflection coefficient:} \quad r_i = \frac{A_{i+1} - A_i}{A_{i+1} + A_i} \quad , r_i \in [-1, 1]$$

The $-x_{max}$ and x_{max} of the quantization characteristic curve can then be simply set to -1 and 1, thus having the benefit an efficient quantization. However, standard practice is to use log area ratios which can be easily computed from the reflection coefficients by taking the logarithm of the ratio between the areas of successive tube segments in the simplified model of speech production. The logarithm produces a compression resulting in a companding scheme associated with a non-uniform quantization. The motivation for this is that reflection coefficients corresponding to large values of r are quantized more accurately which has been shown experimentally to be have better perceptual benefits.

$$\text{Log area ratio:} \quad L_i = \log \frac{A_{i+1}}{A_i} = \log \frac{1 + r_i}{1 - r_i}$$

In order to achieve the optimal bit rates associated with parametric coding, we see that it is not always enough to employ parametric models for speech production. In order to minimize the bit rate as much as possible, it becomes necessary to also to examine different methods of transforming the LPC coefficients to a domain in which they are more easily quantized. This

same principle also holds for the gain parameter where it also beneficial to take a logarithmic transform prior to quantization.

Typically, LPC-10 analysis yields a set of ten log area coefficients in which the first two would be quantized using $5\,\mathrm{bit}$ each by a non-uniform quantization scheme, and the remaining are quantized using $2\,\mathrm{bit}$ to $5\,\mathrm{bit}$ by a uniform quantization scheme. This results in a maximum of $41\,\mathrm{bit}$ spent for the log area ratios. The total number of bits spent for the LPC-10 coding scheme are summed up in Figure 7.6.

Parameter	bits, w
LPC coefficients	$41\,\mathrm{bit}$
Fundamental frequency	$7\,\mathrm{bit}$
Gain	$5\,\mathrm{bit}$
Synchronization bit	$1\,\mathrm{bit}$
TOTAL	**54 bit**

Figure 7.6: Bits spent using the LPC-10 parametric coding scheme.

The bit rate can then be computed using the total number of bits spent for each frame and the frame advance.

$$\text{Bit rate} = \frac{54\,\mathrm{bit}}{\mathrm{frame}}\frac{1\ \mathrm{frame}}{22.5\,\mathrm{ms}} = 2.4\,\mathrm{kbps}$$

This is a savings in bit rate of more than a factor of ten compared to the $32\,\mathrm{kbps}$ DECT waveform coding scheme introduced in the previous section. This is, of course, extremely low, the price is a drop in quality and an increased sensitivity to noise and artifacts.

7.4 Hybrid Coding

The hybrid coding scheme is a compromise between the parametric and waveform coding schemes in an to effort to achieve the low bit rate of the parametric coder and the better quality of a waveform coder. Like the parametric coder, a hybrid coder performs an adaptive LPC analysis of successive speech frames, however some processing is performed on the time domain residual and this is transmitted as is done in a waveform coding scheme. It is somewhat similar to the open loop ADPCM codec introduced in the section on waveform coders, however the residual signal in a hybrid coding scheme is approximated only rather roughly. This is done with a low pass filtering and sub-sampling in the residual excited linear prediction (RELP) scheme or a nearest neighbor codebook index in the codebook excited linear prediction scheme (CELP). By

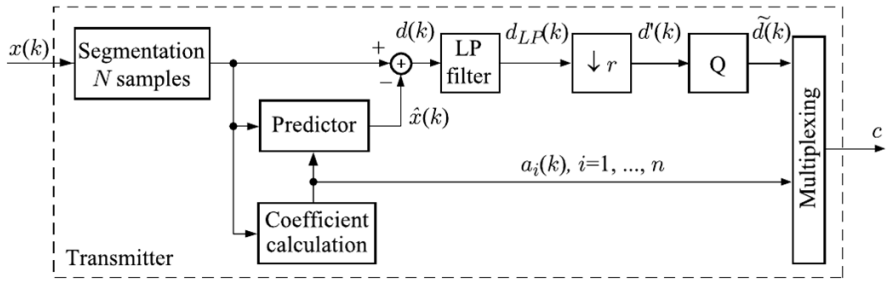


Figure 7.7: Residual excited linear prediction. [7]

transmitting the LPC coefficients and performing a waveform coding of the residual, it is possible to increase the quality of the coded speech and overcome the robotic sounding output of the LPC-10 codec.

Figure 7.7 depicts a flow chart of the residual excited linear prediction (RELP) coding scheme. Like all hybrid coding schemes, RELP initially performs an LPC analysis of a speech frame. In the standardized scheme, a vector quantizer is used to code the LPC coefficients so that only a codebook entry is transmitted. This not only reduces the number of bits that must be transmitted (at the expense of memory to store the codebooks), it was previously shown that vector quantization has the added benefit of capturing the correlation between the coefficients. The process that differentiates the RELP scheme is the heavy low pass filtering and sub sampling that is performed before the coding of the residual signal. This allows for the transmission of only a fraction of the bandwidth contained within the original excitation signal and therefore reduces the bit rate. For synthesis, replicas of the excitation signal are used which will certainly result in errors. Still, the overall result has a quality that is preferable to standard parametric coders.

Figure 7.8 depicts the initial stage of RELP processing of the excitiation signal. The spectrum is first low pass filtered and then then sub sampled to $\frac{f_s}{r}$.

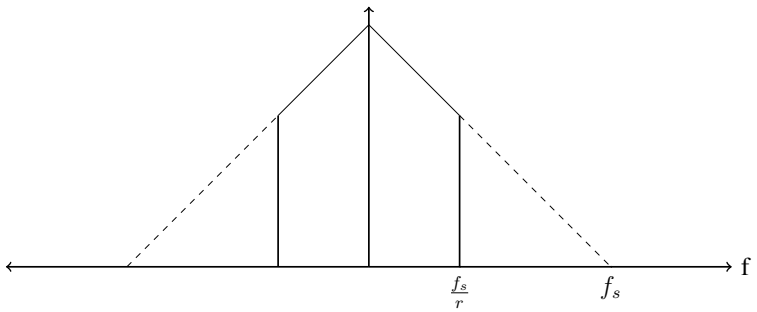


Figure 7.8: The first lowpass filtering stage in RELP coding shown in the frequency domain.

If $r = 2$, then the sub sampling step is equivalent to taking every second sample. This is analogous to multiplying the time domain excitation signal with a Dirac comb with a pulse width of $\frac{r}{f_s}$, $\text{Ш}_{\frac{1}{r/f_s}}$. Of course, this multiplication in the time domain corresponds to a convolution in the frequency domain with a Dirac comb of pulse width of $\frac{f_s}{r}$, $\text{Ш}_{\frac{1}{f_s/r}}$. This results in replicas of the low pass filtered spectrum shown in Figure 7.9.

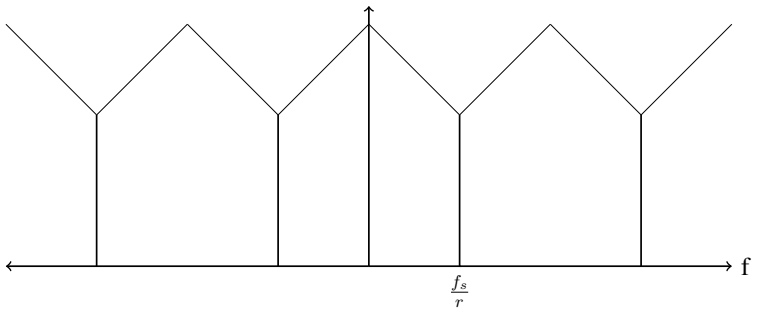


Figure 7.9: The second downsampling stage in RELP coding. The downsampling is a multiplication with a Dirac comb in the time domain. The resulting convolution in the frequency domain produces replicas of the lowpass filtered spectrum.

The reconstruction step involves an upsampling to the original sampling frequency by simply adding zeros between the successive samples (for $r = 2$). The result is an excitation signal with the original sampling rate, f_s, but with a new spectrum filled with the replicas of the low pass, sub sampled spectrum as shown in Figure 7.10.

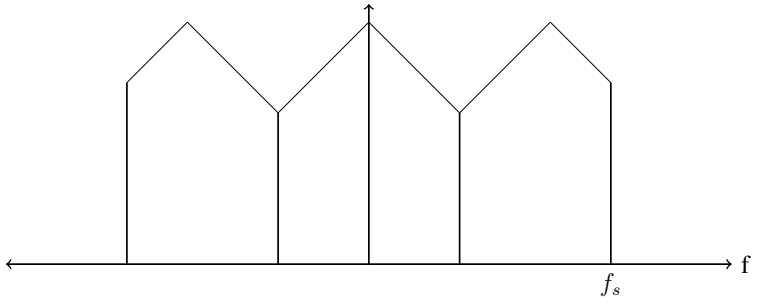


Figure 7.10: The final upsampling stage in RELP coding. The upsampling preserves the replicas of the spectrum, however the natural $\frac{1}{f}$ frequency roll-off introduced by the vocal tract, will damp these higher frequency distortions.

Of course, this reconstruction will not be completely correct because the harmonic structure at higher frequencies does not correspond to the structure of the original excitation signal. The result is a metallic sound for more high pitch voices and sounds, however voiced sounds have a natural $\frac{1}{f}$ decay in energy meaning that these higher frequency distortions will tend to be damped by the vocal tract filter function. Therefore, the effects are not as pronounced as they would seem, and the RELP codec produces a considerable increase in quality over the LPC-10 and other parametric coding schemes.

The codebook excited linear prediction (CELP) method performs a similar analysis and transmission of LPC coefficients by means of vector quantization as RELP, however CELP differs in that the excitation signal is also vector quantized and only a codebook index is transmitted to a receiver. The index then corresponds to a specific excitation signal in the receiver's codebook that is then used to synthesize the output signal. The codebook is generally trained in a similar manner as that described in the section on vector quantization. Common training algorithms,

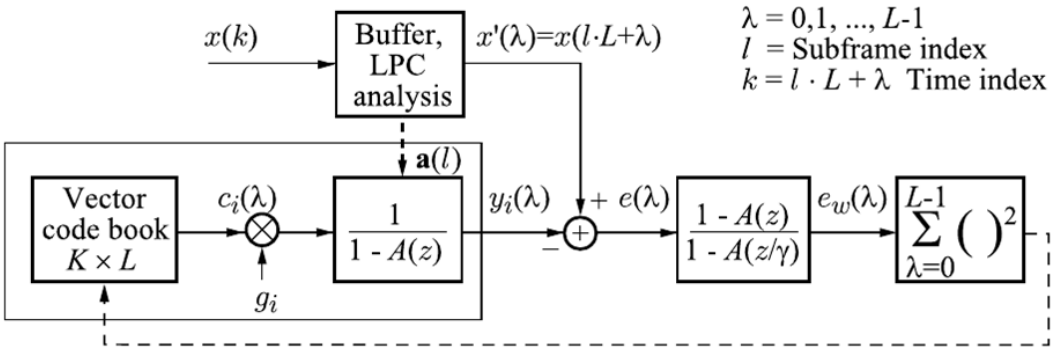


Figure 7.11: Codebook excited linear prediction. [7]

such as the Linde-Buzo-Gray or a k-means, are employed in order to obtain a codebook of typical excitation patterns that are associated with real speech.

The CELP process is depicted in Figure 7.11. The input signal, $x(n)$, is first windowed and separated into successive frames upon which an LPC analysis is performed. The sender then synthesizes multiple potential output signals by filtering all of the codebook entries corresponding to excitation signals with the current LPC coefficients. These potential output signals are then compared to the current speech frame and then the index is chosen that corresponds to the excitation signal producing the closest match. This best fit was previously introduced as a nearest neighbor approach in which the vector was chosen that corresponded to the smallest Euclidean distance. However, in this method, the perceptual quality of the coder is improved by first performing a perceptual weighting of the synthesized signal and then finding the nearest neighbor within this new perceptual domain. This entire process is referred to as *analysis-by-synthesis*.

The sender must then simply transmit an index corresponding to an excitation signal and an index corresponding to a vector of AR coefficients. CELP coding is employed in GSM meaning that it is the coding scheme that is implemented in our mobile telephones.

7.5 Perceptual Coding

The previous coding schemes that have been introduced employ models of speech production that utilize a priori knowledge in an effort to reduce the amount of bits required to code the input speech signal. Perceptual coding methods reduce the amount of information in the output signal by utilizing a priori knowledge of human sound perception. These are methods that are employed in MP3 or AAC compression.

Perceptual coders aim at ensuring that the quantization noise is inaudible while still using the smallest possible number of bits per sample for quantization. Frequency masking occurs when

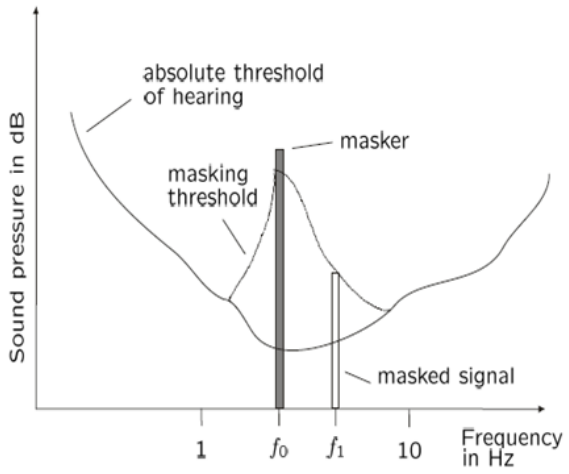


Figure 7.12: Human auditory filters result in masking of similar frequencies. This allows for perceptual coding.

one frequency cannot be perceived due to the presence of a similar frequency. Figure 7.12 depicts this process superimposed on a frequency tuning curve for a single auditory filter. The masker creates a masking threshold and frequencies within this tuning curve that fall under this threshold remain inaudible.

Quantization noise that falls within this masking area will also be inaudible thus allowing to adapt the quantization step size based on such masking effects. Thus, a frequency dependent adaptive quantizer is employed, which results in a significant savings in bitrate. This concept was responsible for the mp3 revolution that changed the way that music was purchased, stored, and played.

8 — Speech Enhancement

Learning objectives

- Spatial processing
- The Wiener Filter
- Statistics

Speech enhancement is a very important technique in digital signal processing for speech communications devices such as smartphones and hearings aids. The hearing impaired have difficulty with understanding speech in noisy environments. A hearing aid not only functions as an amplifier, but also applies a compression that results in soft sound being amplified more than loud sounds. Quite often, there exists the scenario when a speaker's voice is accompanied by a softer background noise such as the fan of a computer or an air conditioning system. In these situations, the compression will boost the noise power in contrast to the speech thus resulting in a reduced SNR. By employing speech enhancement techniques that will be discussed in this chapter, it is possible to greatly reduce these effects in an effort to aid the hearing impaired in living normal lives. These techniques can be divided into two approaches: spatial processing and statistical spectral processing.

8.1 Spatial processing

Spatial processing utilizes multiple microphones in an effort to exploit the spatial characteristics of the sound field. The human ears can be thought of as two microphones providing spatially separated signals upon which the human auditory system can perform similar processing. Using these signals, it is possible to perform a localization to distinguish speakers in a multi speaker environment and focus attention onto the desired source. Similar techniques can be performed in signal processing.

Figure 8.1 displays the concept behind the *delay and sum beamformer*. When the distance to a sound source is significantly longer than the wavelength emitted by the source, then one can assume a plane wave propagation in which successive wavefronts are parallel. The sound will arrive at the closer microphone before the other resulting in a measurable delay of arrival, τ. By shifting the delayed signal by τ and summing the two signals, the sound information from the direction corresponding to a specific delay, τ, will add up constructively. Sources originating from any other direction that do not correspond to the delay will therefore add up destructively. This basic principle can be implemented for any number of microphones.

There are also more complicated beamformers, such as the *minimum variance distortionless response* (MVDR) beamformer, that can utilize a priori knowledge of a noise source in order to optimize results. Figure 8.1 shows a beam pattern representing the gain applied to a signal as a function of spatial location. Ideally, the desired source would be undistorted, meaning that maximum gain would be applied to signals originating from the target location, whereas maximum attenuation would be applied to spatial locations containing the interfering sources. The beam pattern must be designed for different types of noise sources, such as a diffuse noise field containing noise sources from all directions. Because in such an environment, there is no distinct noise source, it is desirable to make the beam as narrow as possible and with maximum directivity towards the target source, also known as a *superdirective beamformer*.

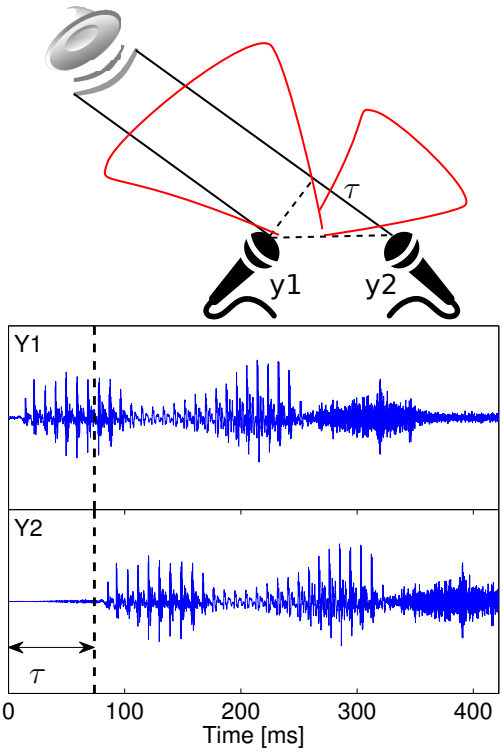


Figure 8.1: A simple two channel beamformer is displayed. The schematic depicts how spatially separated microphones result in a time delay of signals. This allows for the creation of the beamformer shown in red.

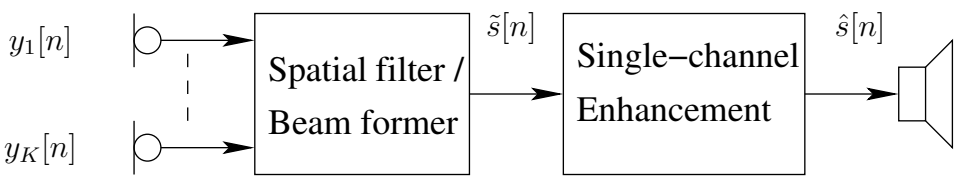


Figure 8.2: Flow chart of single channel speech enhancement using a beamformer/spatial filter.

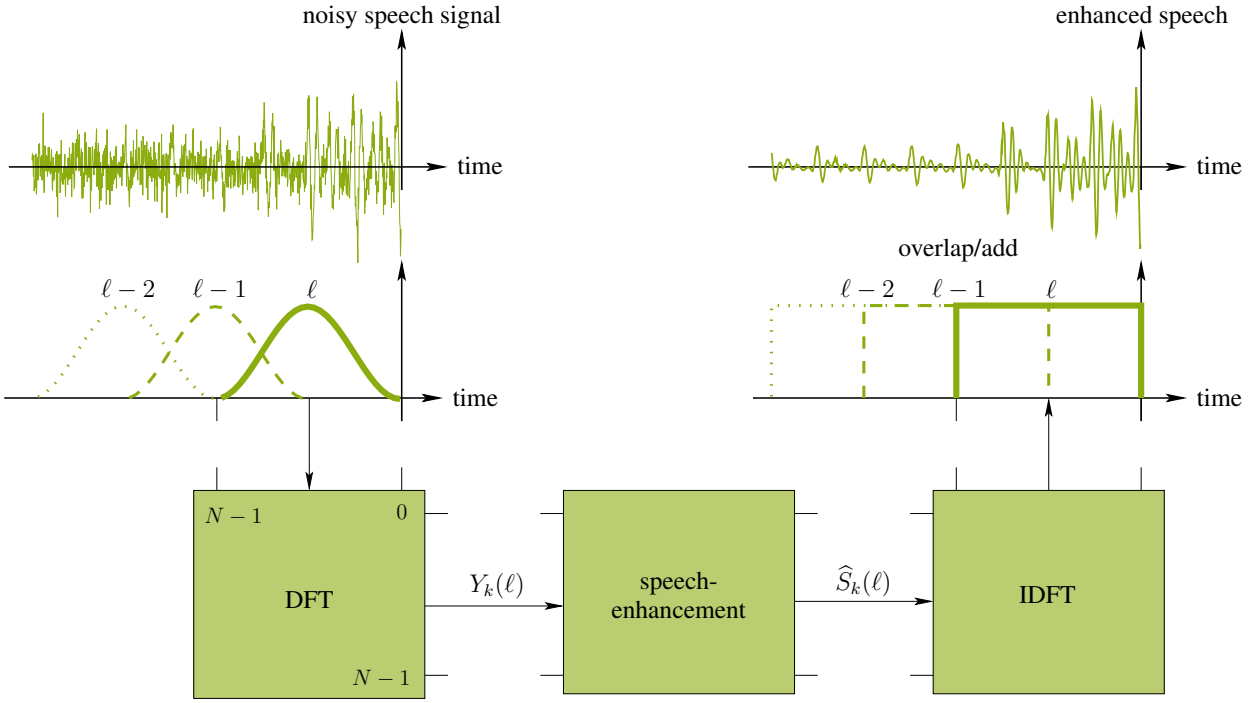


Figure 8.3: A block processing diagram of the single channel speech enhancement process.

Beamforming results in a combination of microphone signals with a single channel output as shown in Figure 8.2. It can be shown that the combination of a spatial beamformer with a single channel postfilter is optimal in the minimum-mean squared error sense under certain assumptions (Gaussian noise, rank-1 signal model). This result allows us to conveniently discuss multichannel and single-channel enhancement separately.

8.2 Computing the gain function

Single channel speech enhancement techniques are performed on the outputs of multiple microphone applications or in cases where practical limitations only allow for one microphone. It then becomes necessary to rely on the different statistical properties of the input signal and a priori knowledge of speech and noise, in an effort to produce an enhanced signal. This process is typically performed in the spectral domain by computing a gain function that is used to attenuate the undesired signal.

Figure 8.3 is a block processing diagram of the single channel speech enhancement process. Because speech can be shown to be quasi-stationary for window lengths of $32\,\mathrm{ms}$, the noisy signal is windowed. A tapered analysis window is generally used to reduce effects of spectral

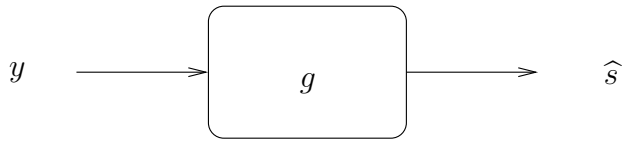


Figure 8.4: LTI model of the Wiener filtering of a noisy input signal.

leakage and to correspond to an appropriate synthesis window. The segmented and weighted blocks are then transformed into the Fourier domain by using a DFT as implemented by the FFT algorithm. We typically work in the frequency domain because speech is perceived in the spectral domain as a result of the tonotopic decomposition of the basilar membrane. However, because we are working with short windowed segments, the temporal structure of the original signal is still maintained. The speech enhancement is performed by deriving the corresponding filter to the current speech frame which results in an estimate of the clean speech. In order to produce the enhanced speech, the processed blocks must then be transformed back into the time domain by utilizing the overlap-add process introduced in a previous chapter.

8.2.1 Gain function: The Wiener filter

Figure 8.4 depicts the Wiener filtering of a noisy input signal as an LTI system in which a noisy signal, y, is processed by a system described by g to produce the estimated clean speech signal, $\widehat{s}$. For a linear system, g is not a function of the input y. Furthermore, typically a linear signal model is employed, i.e. the noisy signal $y = s + n$ is a superposition of the clean speech, s, and the added noise, n. The Wiener filter is the optimal linear filter that minimizes the mean squared error (MMSE) between the true clean speech, s, and the estimated clean speech, $\widehat{s}$.

In time domain, the Wiener filtering results in a convolution of the noisy input signal y and the filter's impulse response $g(n)$

$$\widehat{s}(n) = \sum_{\nu=-\infty}^{\infty} g(\nu) y(n-\nu)$$

The Wiener filter coefficients are given by those values of g that minimize the mean squared error between the true clean speech, s, and the estimated clean speech, $\widehat{s}$.

$$\min_{g} \left(\mathrm{E}\big((s(n) - \widehat{s}(n))^2 \big) \right)$$

The result is a function of both the autocorrelation of the noisy observation and the cross-correlation between the noisy and clean speech.

$$\sum_{\nu=-\infty}^{\infty} g(\nu) \varphi_{\mathrm{yy}}(n-\nu) = \varphi_{\mathrm{ys}}(n)$$

This set of equations can be easily solved in the spectral domain. We perform the transformation by using the known identity that the Fourier transform of the autocorrelation function is the power

spectral density (PSD) of the signal, and the fact that a convolution in time-domain results in a multiplication in spectral domain. We can then solve for the gain function $G(\mathrm{e}^{\mathrm{j}\Omega})$ per frequency Ω, as

$$G(\mathrm{e}^{\mathrm{j}\Omega}) = \frac{\Phi_{\mathrm{YS}}(\mathrm{e}^{\mathrm{j}\Omega})}{\Phi_{\mathrm{YY}}(\mathrm{e}^{\mathrm{j}\Omega})} = \frac{\Phi_{\mathrm{SS}}(\mathrm{e}^{\mathrm{j}\Omega})}{\Phi_{\mathrm{SS}}(\mathrm{e}^{\mathrm{j}\Omega}) + \Phi_{\mathrm{NN}}(\mathrm{e}^{\mathrm{j}\Omega})}$$

Here, we assumed that speech and noise are uncorrelated, which results in $\Phi_{\mathrm{YS}} = \Phi_{\mathrm{SS}}$ and $\Phi_{\mathrm{YY}} = \Phi_{\mathrm{SS}} + \Phi_{\mathrm{NN}}$.

In practice, Wiener filtering is not performed in the time domain as most practical applications, such as cell phones and hearing aids, apply the enhancement in the spectral domain. Because we are really interested in the optimal solution in the domain in which we are working, the Wiener filter will now be derived more intensely in the spectral domain directly.

Derivation in the STFT domain

Because the Fourier transform is linear, the same time domain signal model can be used in which the noisy observation is a superposition of the clean speech and the noise.

$$Y_k = S_k + N_k$$

The time domain convolution associated with the linear filter becomes a multiplication in the frequency domain with a gain function, $G_k(\ell)$. If it is assumed that the DFT block length is long enough to capture all correlations in the signal, then the resulting spectral coefficients will be decorrelated. This means that the optimal filter can be derived for each time frequency point, (k, ℓ).

$$\widehat{S}_k(\ell) = G_k(\ell)\, Y_k(\ell)$$

It is important to note that $G_k(\ell)$ is linearly constrained and therefore cannot be a function of the noisy observation, $Y_k(\ell)$.

Again, we use MMSE criterion for obtaining an optimal linear gain function $G_k(\ell)$, as

$$\min_{G} \left(\mathrm{E}\big(|S_k - G_k Y_k|^2\big) \right)$$

We now simplify this quadratic expression, however keeping in mind that the magnitude of a complex valued signal is the product of itself and its complex conjugate.

$$|S_k - G_k Y_k|^2 = |S_k|^2 + |G_k|^2 |Y_k|^2 - S_k G_k^* Y_k^* - S_k^* G_k Y_k$$

The cross terms are complex conjugates of each of other and the sum of a complex number and its conjugate is twice the real part.

$$\min_G \left(\mathrm{E}\big(|S_k|^2 + |G_k|^2|Y_k|^2 - 2\Re(G_k^* Y_k^* S_k)\big) \right)$$

The expected value operator is linear and can be distributed to each individual term of the expression. Because G_k is a fixed value that depends only upon the noisy observation, it is not a random variable and can be written outside of the expected value operator. The real part operator is also linear, and the expected value operator can be brought into the argument of the function.

$$\min_G \left((\mathrm{E}\big(|S_k|^2\big) + |G_k|^2 \mathrm{E}\big(|Y_k|^2\big) - 2\Re(G_k^* \mathrm{E}(Y_k^* S_k)) \right)$$

This is the expression that we would like to minimize with respect to, G_k, however because G_k is complex valued, the process slightly more difficult than for real-valued coefficients. However, looking at the expression, it can be seen that the middle term is only dependent upon the amplitude of G_k and that the only part dependent upon the phase of G_k is the last term containing the real part operator. Because this latter term is negative, the entire expression will be minimized when this term is maximized. By setting some arbitrary complex number $Z = \mathrm{E}(Y_k^* S_k)$, it is possible to closer examine when this condition occurs.

$$\Re(G_k^* Z) = |G_k||Z|\Re(\mathrm{e}^{j(\phi_Z - \phi_G)})$$

The real part of a complex exponential is the cosine.

$$= |G_k||Z|\cos(\phi_Z - \phi_G)$$

This expression is maximized when the value of the cosine is one which occurs when the two phases are the same.

$$\phi_{Gopt} = \phi_Z = \angle \mathrm{E}(Y_k^* S_k)$$

This optimal phase can now be substituted into the original expression and then it is possible to minimize with respect to $|G_k|$ as opposed to the complex valued G_k.

$$\min_{|G_k|} \left(\mathrm{E}\big(|S_k|^2\big) + |G_k|^2 \mathrm{E}\big(|Y_k|^2\big) - 2|G_k||\mathrm{E}(Y_k^* S_k)| \right)$$

To minimize a quadratic function with respect to a variable, one can simply find the point where the first derivative is equal to zero.

$$\frac{\partial}{\partial |G_k|} \left(\mathrm{E}\big(|S_k|^2\big) + |G_k|^2 \mathrm{E}\big(|Y_k|^2\big) - 2|G_k||\mathrm{E}(Y_k^* S_k)| \right) = 0$$

The derivative is a linear operator and can be applied to each term separately. The first term is not a function of $|G_k|$ and is therefore zero. For the remaining terms, a simple power rule can be applied.

$$\begin{aligned} 0 &= 2|G_k|\mathrm{E}\big(|Y_k|^2\big) - 2|\mathrm{E}(Y_k^* S_k)| \\ |G_k| &= \frac{|\mathrm{E}(Y_k^* S_k)|}{\mathrm{E}(|Y_k|^2)} \end{aligned}$$

The above expression must now be combined with the solution for the optimum phase. Because the denominator is real valued it has no effect on the phase of the expression and can be ignored. Looking closer at the numerator reveals that it is the same expression that was previously derived for the optimum phase. Because G_k and $\mathrm{E}(Y_k^* S_k)$ will always share the same phase, the optimal solutions is achieved by removing the absolute value operators. Now, the Wiener filter can be described as the correlation of the noisy signal with the speech signal, normalized by the magnitude squared of the noisy observation.

$$G_k = |G_k|\mathrm{e}^{j\phi_G} = \frac{\mathrm{E}(Y_k^* S_k)}{\mathrm{E}(|Y_k|^2)}$$

Let us know insert $Y_k = S_k + N_k$

$$= \frac{\mathrm{E}((S_k + N_k)^* S_k)}{\mathrm{E}((S_k + N_k)(S_k + N_k)^*)}$$

Again, assuming that speech and noise are uncorrelated things can be simplified. The above expression can then be expanded and all cross terms that correlate speech and noise go to zero, resulting in

$$G_k = \frac{\mathrm{E}(|S_k|^2)}{\mathrm{E}(|S_k|^2) + \mathrm{E}(|N_k|)^2)}$$

Thus, derived in the spectral domain, the Wiener filter, G_k is a function of the second order moments of the speech and noise signals. Because the speech and noise coefficients S and N can be considered to be zero mean signals, the second order moments correspond to the variances, i.e. $\mathrm{E}(|S_k|^2) = \sigma_{\mathrm{S},k}^2$ and $\mathrm{E}(|N_k|^2) = \sigma_{\mathrm{N},k}^2$, resulting in

$$G_k = \frac{\sigma_{\mathrm{S},k}^2(\ell)}{\sigma_{\mathrm{S},k}^2(\ell) + \sigma_{\mathrm{N},k}^2(\ell)}.$$

This form is the same as the form derived in time domain, when power spectral densities and variances are exchanged.

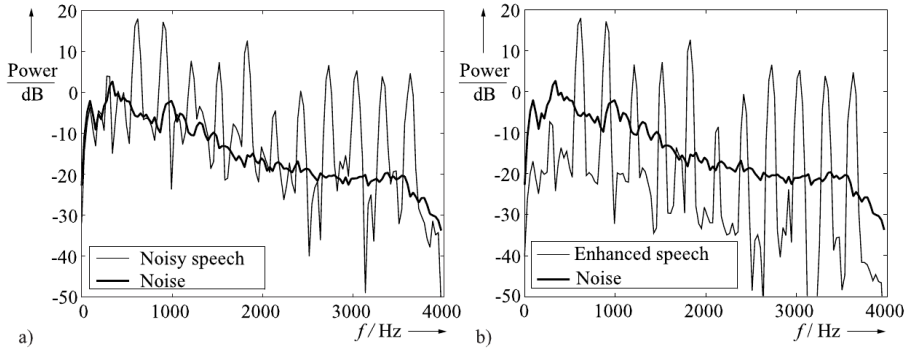


Figure 8.5: (Left) The spectrum of a noisy voiced speech segment is shown. Notice how the noise tends to fill in the gaps between the harmonics. (Right) The spectrum of an enhanced voiced speech segment is shown. The enhancement process has decreased the noise in between harmonics. [8]

Illustration of the Wiener filter

Figure 8.5 shows how the Wiener filter can be applied to a noisy voiced speech signal within a single time frame, $\ell = \ell_0$. It is possible to resolve the harmonics of the speech signal, however the gaps in between the peaks are filled with energy from the noise. The value of the linear gain function is computed using the local second order moments for each time-frequency bin, (k, ℓ) and then multiplied by the noisy observation, $Y_k(\ell)$, to produce the clean speech signal, $\widehat{S}_k(\ell)$.

$$\widehat{S}_k(\ell) = \frac{\sigma^2_{\mathrm{S},k}(\ell)}{\sigma^2_{\mathrm{S},k}(\ell) + \sigma^2_{\mathrm{N},k}(\ell)} Y_k(\ell)$$

The local powers of the noise (dark solid line) and the noisy observation (lighter line) can be compared in the left of Figure 8.5. For frequencies, such as the harmonics, when the $\sigma^2_{\mathrm{S},k}(\ell) \gg \sigma^2_{\mathrm{N},k}(\ell)$ the gain function approaches unity and the signal is left unattenuated. As the noise power increases in between harmonics, $\sigma^2_{\mathrm{S},k}(\ell) \ll \sigma^2_{\mathrm{N},k}(\ell)$, the gain function approaches zero, thus attenuating the signal. The noisy observation is therefore weighted by the local SNR to produce the clean speech signal displayed in the right of Figure 8.5. The spectral peaks are left unattenuated due to the almost unity gain function, and the noise in between the harmonics is greatly reduced. The Wiener filter thus performs a noise reduction by attenuating the noisy observation at time frequency points in which noise dominates and by remaining transparent at points in which speech dominates. Figure 8.5 shows this process for only one time frame, however the results of Wiener filtering over time can be seen in the spectrograms in Figure 8.6.

The Wiener filter was derived as being the linear constrained MMSE optimal estimator. In doing so,the resulting gain function, G, can be said to be the optimal solution for the constraints that were set.

$$G = \arg\min_{G} \Big(\mathrm{E}\big((S - G)^2\big) \Big)$$

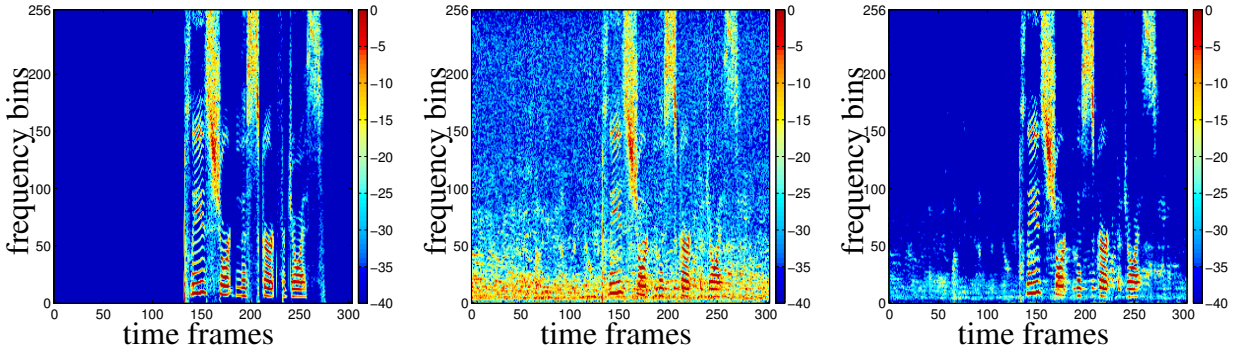


Figure 8.6: Spectrograms are shown of a speech signal for (left) clean speech, (middle) noisy speech, and (right) enhanced speech

8.2.2 Gain function: Spectral subtraction

It is also possible to obtain an estimate of the clean speech coefficients by simply subtracting the noise periodogram from the noisy signal periodogram in a technique called *spectral subtraction*. The *periodrgram* is simply the magnitude squared of a spectral quantity. however, because we are dealing with magnitudes squared quantities the cross-terms have to be considered,

$$\begin{aligned} |\widehat{S}|^2 &= |Y|^2 - |N|^2 \\ &= (S+N)(S+N)^* - |N|^2 \\ &= |S|^2 + |N|^2 + 2\Re(SN^*) - |N|^2 \\ &= |S|^2 + 2\Re(SN^*) \end{aligned}$$

Thus, even if the perfect noise periodogram was available, spectral subtraction would not yield a perfect clean speech estimate, but also includes some error term. Dependent upon S and N, this error term can generally not be ignored. However, by taking the expected value of $|\widehat{S}|^2$, we get

$$\begin{aligned} \mathrm{E}\Big(|\widehat{S}|^2\Big) &= \mathrm{E}\big(|S|^2 + 2\Re(SN^*)\big) \\ &= \mathrm{E}\big(|S|^2\big) + \mathrm{E}(2\Re(SN^*)). \end{aligned}$$

Once again making the assumption that the speech and noise are uncorrelated, the last term goes to zero and we obtain

$$\mathrm{E}\Big(|\widehat{S}|^2\Big) = \mathrm{E}\big(|S|^2\big) = \sigma_S^2$$

By rearranging some terms, we can also write spectral subtraction in form of a gain function.

$$\begin{aligned}
\mathrm{E}\Big(|\widehat{S}|^2\Big) &= \mathrm{E}\big(|Y|^2\big) - \mathrm{E}\big(|N|^2\big) \\
&= \mathrm{E}\big(|Y|^2\big)\Big(1 - \frac{\mathrm{E}(|N|^2)}{\mathrm{E}(|Y|^2)}\Big) \\
&= \mathrm{E}\big(|Y|^2\big)\Big(\frac{\mathrm{E}(|Y|^2) - \mathrm{E}(|N|^2)}{\mathrm{E}(|Y|^2)}\Big) \\
&= \mathrm{E}\big(|Y|^2\big)\Big(\frac{\mathrm{E}(|S|^2)}{\mathrm{E}(|Y|^2)}\Big)
\end{aligned}$$

$$G_{\text{spectralsubtraction}} = \frac{\sigma_S^2}{\sigma_Y^2} = \frac{\sigma_S^2}{\sigma_S^2 + \sigma_N^2}$$

This gain function obtained by spectral subtraction looks identical to the Wiener filter, however it is applied to the expected value of the noisy periodogram instead of the noisy signal. However, although they are both motivated quite differently, the term *spectral subtraction* is often used as a general term to describe single channel speech enhancement algorithms. Strictly speaking, spectral subtraction is the result of a heuristic approach whereas the Wiener filter is the result of a statistical optimal derivation.

Equation 8.1 shows how, in practice, spectral subtraction is applied differently by using smoothed estimates of the speech and noisy periodograms.

$$\widehat{|S|^2} = |Y|^2 \left(1 - \left(\frac{\overline{|N|^2}}{\overline{|Y|^2}}\right)^{\mu}\right)^{\nu} \tag{8.1}$$

The method is tuned heuristically through this smoothing and adjusting the parameters ν and μ. However, it is not possible to say under what constraints these parameters are optimal. In contrast, if we optimize for a certain result under a specific set of constraints, then we know that the result is the best that can be achieved under those constraints, and we avoid tuning parameters. This is true for the Wiener filter which is constrained to be linear and is optimal in the MMSE sense.

8.3 Power spectral density estimation

The previous section introduced two gain functions, the Wiener filter and spectral subtraction, that can be utilized in single channel speech enhancement. However both the Wiener filter and spectral subtraction require to know the noise and speech power spectral densities (PSDs). Because a common implementation of these filters allows access to only the noisy observation,

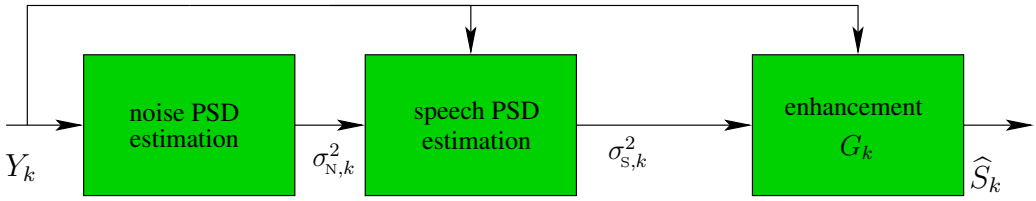


Figure 8.7: Flow chart describing the speech enhancement process in which the PSD of the noisy signal must now be estimated.

these quantities must be estimated. Figure 8.7 displays a block diagram of this process in which the noisy observation, Y, is input into the first block and is then used to estimate the noise PSD, σ_N^2. Based upon these results, the speech PSD, σ_S^2, is then estimated so that the gain function can be computed. The estimation of the PSD is crucial in providing robust speech enhancement and the present section will introduce several methods and compare their relative advantages and disadvantages.

8.3.1 $\widehat{\sigma_N^2}$ estimation based on voice activity detection

Figure 8.8 shows the time-domain plot of a noisy observation, $y(n)$, with the corrupting noise signal, $d(n)$. Note that the noisy observation consists of noise only when the speech signal is zero. It is then possible to estimate the noise variance in the areas of speech inactivity. $\widehat{\sigma_d^2}$ can then be formally defined as the average of the squared noisy observations, $|y(n)|^2$, over the set of samples in which there is no speech active. The purpose of taking the mean is to approximate the expected value in the same manner that a sample mean as an estimate of the expected value. If $\mathcal{N}$ is the set of samples in which speech is inactive, then we normalize over the cardinality of the set, $|\mathcal{N}|$.

$$\widehat{\sigma_n^2} = \frac{1}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} |y(n)|^2$$

$\mathcal{N}$: set of samples where speech is absent

This method of estimating the noise variance is probably the simplest and can also be implemented in the STFT domain, where the principle idea would be the same. Then, the spectral variance represents the power spectral density of the noise and is defined per frequency bin. However, there are several key problems with this approach. Most practical applications involve non-stationary noise in which noise power fluctuates as seen in the example of a car passing-by during a conversation. In this situation, the noise signal starts-off softly and then becomes loud, resulting in problems with tracking the change in the noise PSD. Because the noise PSD can only be calculated when speech is absent, then $\widehat{\sigma_N^2}$ cannot be updated during speech activity. The noise PSD would be therefore be underestimated and noise would bleed into the enhanced speech signal. Another drawback to the technique, is that it is difficult to define a speech activity threshold. If the choice is too conservative, then the noise PSD is updated less frequently, resulting in the same problems already stated. However, if this threshold is set too high, then speech energy will leak into the noise PSD estimate and will cause speech distortions.

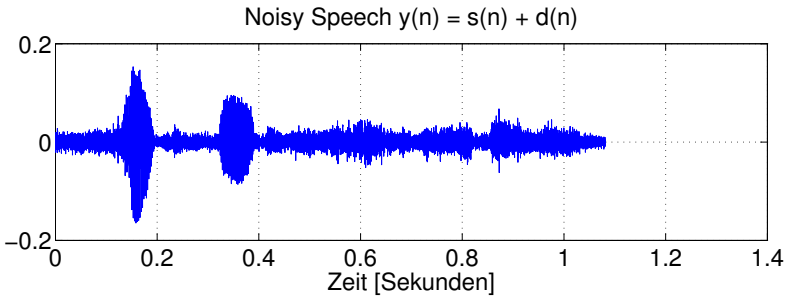


Figure 8.8: A noisy time-domain speech signal.

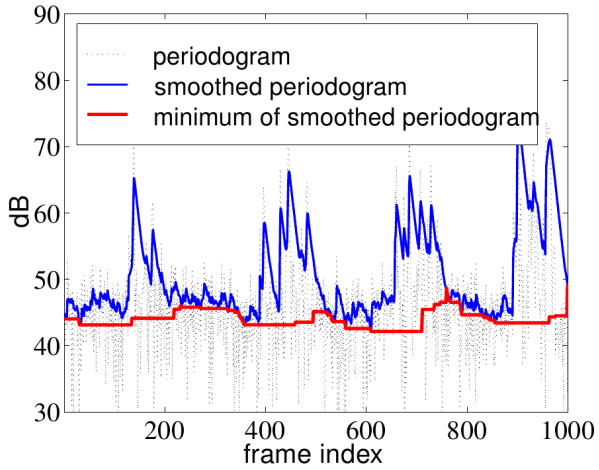


Figure 8.9: The minimum statistics approach attempts to estimate the noise PSD within frames of speech inactivity. Estimated minima lie below the mean of the smoothed periodogram allowing noise power to bleed into the enhanced signal thus requiring a bias compensation.

8.3.2 $\widehat{\sigma_N^2}$ estimation based on minimum statistics

Voice activity detection requires a detector that determines in which frames speech is active or inactive. The *minimum statistics* approach is based upon the observation that the noisy periodogram, $|Y(n)|^2$ does not goto zero, but only to a minimum value that is related to the noise PSD. Figure 8.9 displays the noisy periodogram, $|Y_{k_0}(n)|^2$, for frequency bin $k = k_0$. The minimum statistics approach estimates $\widehat{\sigma_N^2}$ by assuming that the minimum value of the noisy observation PSD in the areas of speech inactivity is related to the noise PSD. By updating this value over predetermined time frames, this approach is more efficient at tracking noise PSD changes than voice activity detection.

The first step of the process is to smooth the noisy periodogram. This smoothing can be done using a recursive smoothing approach.

$$\overline{|Y_k(l)|^2} = \alpha|Y_k(l-1)|^2 + (1-\alpha)|Y_k(l)|^2$$

The next, step is the search for the minimum of noisy periodogram over a predefined time window. A typical length of this search window is 1.5 s in which the assumption is made that there

will be at least one frame where speech is inactive.

We now assume that the found minimum is only influenced by the noise signal and does not contain speech. Then, a bias compensation is performed to relate the found minimum of the noise periodogram to the mean of the noise periodogram to obtain $\sigma_N^2 = \mathrm{E}(|N|^2)$. Without a bias correction, the found minimum will always be smaller than the noise PSD. This can be seen in Figure 8.9 in which the estimated minima lie below the mean of the smoothed periodogram. This would results in an underestimation of the noise PSD that result in a limited noise reduction performance. The bias compensation infers the noise PSD from the estimated minima in an effort to prevent this from occurring. However, the bias compensation process is generally prone to errors thus making the smoothing process an even more important step. Smoothing brings the found minima closer to the mean and reduces the errors that would be further expanded in the bias compensation step.

The main disadvantage with the minimum statistics technique is that it can only track changes in noise power that occur within the 1.5 s analysis window. Because noise powers can rise significantly within this time span, there is a risk of it being severely underestimated. If the window is made shorter, then there is the risk that there are no frames in which speech is inactive. The estimated minimum would then be based on the speech power and result in speech distortions in the enhanced signal.

8.3.3 Speech presence probability

Noise power estimation by using speech presence probability detection is one of the simplest algorithms to implement and offers better noise tracking over the previously introduced techniques. The estimator does this by employing the a posteriori speech presence probability that is constantly updated for each time and frequency bin. This is implemented by first defining two hypotheses.

Hypothesis $\mathcal{H}_1$: Speech is present in the time frequency bin under consideration. When this is true, the noisy observation consists of speech and noise.

$$\mathcal{H}_1 \implies Y_k(\ell) = S_k(\ell) + N_k(\ell)$$

Hypothesis $\mathcal{H}_0$: Speech is absent in the time frequency bin under consideration. When this is true, the noisy observation consists only of noise.

$$\mathcal{H}_0 \implies Y_k(\ell) = N_k(\ell)$$

The a posteriori probability that speech is present is defined as the probability of the hypothesis $\mathcal{H}_1$ given the noisy observation.

$$P\big(\mathcal{H}_1 \,\big|\, Y\big)$$

This will be done using Bayesian estimation theory in which certain conditional probability density functions (PDF), are defined and then used to estimate the a posteriori probability.

- $P(\mathcal{H}_1 \mid Y)$: *a posteriori probability*. This is the probability that $\mathcal{H}_1$ is true after observing Y. This is the probability that speech is present given the current noisy observation.
- $p(Y \mid \mathcal{H}_1)$: *likelihood*. This is the distribution of Y under the assumption that $\mathcal{H}_1$ is fulfilled.
- $P(\mathcal{H}_1)$: *prior probability*. This is the probability that $\mathcal{H}_1$ is fulfilled before anything has been observed.
- $p(Y)$: *evidence*. The PDF of p(Y). This is the distribution of the noisy observation.

We are looking to define the posterior probability of speech presence given the current noisy observation, $P\big(\mathcal{H}_1 \,\big|\, Y\big)$. This can be done by using *Bayes' theorem* in which a conditional probability can be expressed in terms of the joint probability normalized by the probability of the initial condition. Then, a conditioned probability can be represented in terms of a joint probability density and the evidence.

$$p(\mathcal{H}_1 \mid Y) = \frac{p(\mathcal{H}_1, Y)}{p(Y)}$$

Because it would be difficult to model a joint probability density for $p(\mathcal{H}_1, Y)$, Bayes' theorem can be applied again. In doing so, the a posteriori probability can now be expressed as a function of the likelihood, the prior, and the evidence.

$$\begin{aligned} p(\mathcal{H}_1 \mid Y) &= \frac{p(\mathcal{H}_1, Y)}{p(Y)} \\ &= \frac{p(\mathcal{H}_1)(Y \mid \mathcal{H}_1)}{p(Y)} \end{aligned}$$

Given the joint PDF of two random variables, $p(\mathcal{H}_1, Y)$, then the marginal probability density $p(Y)$ can always be obtained by integrating over all instances of the other random variable.

$$p(Y) = \int_{-\infty}^{\infty} p(\mathcal{H}_1, Y)$$

$$p(\mathcal{H}_1 \mid Y) = \frac{p(\mathcal{H}_1)(Y \mid \mathcal{H}_1)}{\int_{-\infty}^{\infty} p(\mathcal{H}_1, Y)}$$

In this case. there are only two possible scenarios for $\mathcal{H}$, namely $\mathcal{H}_1$ or $\mathcal{H}_0$. Therefore, $p(Y)$ can be represent as the summation of the joint PDFs of the noisy observation in speech presence and the noisy observation in speech absence.

$$p(Y) = p(Y, \mathcal{H}_1) + p(Y, \mathcal{H}_0)$$

Bayes' theorem can then again be used to represent the joint probability density functions in terms of the corresponding conditional probabilities.

$$p(\mathcal{H}_1 \mid Y) = \frac{p(\mathcal{H}_1)(Y \mid \mathcal{H}_1)}{p(\mathcal{H}_0)p(Y \mid \mathcal{H}_0) + p(\mathcal{H}_1)p(Y \mid \mathcal{H}_1)} \tag{8.2}$$

The posterior distribution is now represented in terms of only the priors for speech presence and speech absence and their corresponding likelihoods. This is helpful because they are often much easier to model. In the simplest case, the probability that speech is present or absent in the current time frequency bin when there is no observation can be modeled as the flipping of a coin. 50/50 is a very conservative choice, neither favoring speech presence or absence. It is possible to use different values, or even train the model based upon real speech, but 50/50 is a simple assumption that still produces adequate results.

$$p(\mathcal{H}_1) = p(\mathcal{H}_0) = \frac{1}{2}$$

It is then necessary to model the likelihoods meaning that we want to know the PDF of the noisy observation when we are given the fact that there is speech present or absent. We first look at the condition when speech is absent. It was stated at the beginning that when speech is absent, the noisy observation will consist only of the noise. Thus, under hypothesis $\mathcal{H}_0$ the likelihood is given by the PDF of the noise coefficients. This could be done by recording noise and plotting a histogram of the observation, but this can also be greatly simplified by assuming a Gaussian distribution, which is often a good assumption when dealing with random data. We thus model the noise as a zero mean Gaussian distribution with variance σ_N^2.

$$p(Y \mid \mathcal{H}_0) = \mathcal{N}(0; \sigma_N^2)$$

A more difficult task is to model the PDF under speech presence because it depends not only on the PDF of the speech signal, but also on the SNR. For example, if the speech is Laplacian distributed and the noise is Gaussian, then a higher SNR will result in the overall distribution tending toward the speech PDF. Otherwise, it will tend toward the noise. In general, it is a practically useful to assume that the speech is also Gaussian distributed. So we now model the speech signal as a zero mean, Gaussian distributed random signal with variance σ_Y^2.

$$p(Y \mid \mathcal{H}_1) = \mathcal{N}(0; \sigma_Y^2)$$

Because the model of the noisy observation is the superposition of uncorrelated noise and speech, σ_Y^2 can be expressed as $\sigma_S^2 + \sigma_N^2$. When it is assumed that the priors are equal, $p(\mathcal{H}_1) = p(\mathcal{H}_0) = \frac{1}{2}$, then they cancel out in (8.2). The structure is similar to the Wiener filter: if the likelihood of speech presence is much higher than that of speech absence, $p(Y \mid \mathcal{H}_1) \gg p(Y \mid \mathcal{H}_0)$, then the a posteriori probability of speech presence will goto one, whereas the expression tends towards zero for the opposite case.

$$p(\mathcal{H}_1 \mid Y) = \frac{p(Y \mid \mathcal{H}_1)}{p(Y \mid \mathcal{H}_0) + p(Y \mid \mathcal{H}_1)}$$

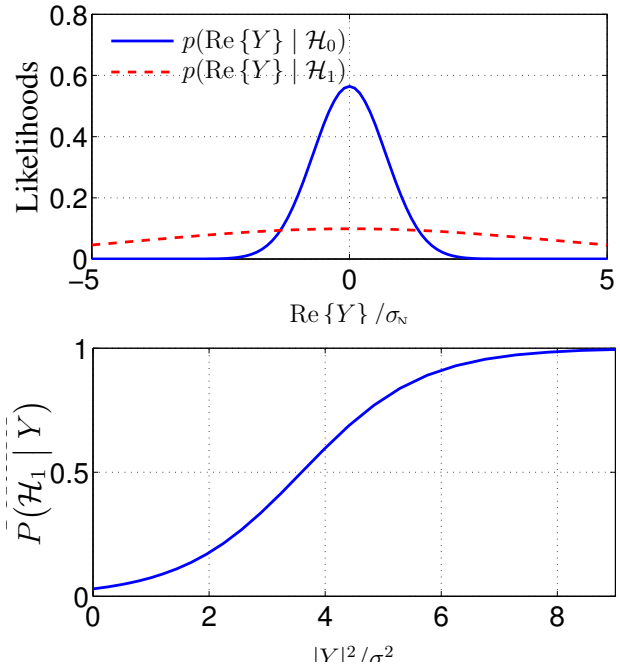


Figure 8.10: Probability density distribution plots of the likelihoods of speech absence and speech presence.

This process can be seen in Figure 8.10 that displays plots of the likelihoods of speech absence and speech presence. Both distributions are Gaussian, however the distribution representing the likelihood of speech presence is spread out among higher magnitudes than that of speech absence. This is measured by the standard deviation which is defined as the squared root of the variance, $\sigma = \sqrt{(\sigma^2)}$.

The model can be further simplified by assuming a specific SNR, $\xi_{\mathcal{H}_1} \approx 15\,\mathrm{dB}$, in speech presence. Substituting the Gaussian models for speech absence and presence likelihoods into Equation 8.2 produces a single exponential function for posteriori probability estimation. If the priors are assumed to be equal, $p(\mathcal{H}_1) = p(\mathcal{H}_0) = \frac{1}{2}$, then this simplified posteriori probability is a function of only the current noisy observation, Y and the noise power σ_N^2.

$$P(\mathcal{H}_1 \mid Y) = \left(1 + \frac{P(\mathcal{H}_0)}{P(\mathcal{H}_1)}(1+\xi_{\mathcal{H}_1})\exp\left(\frac{|Y|^2}{\sigma_N^2}\frac{\xi_{\mathcal{H}_1}}{1+\xi_{\mathcal{H}_1}}\right)\right)^{-1}$$

This posterior probability can then be used to estimate the probability of speech presence in each time and frequency bin in a speech enhancement framework. From that, an estimate of the noise periodigram can be obtained as

$$\widehat{|N|^2} = P(\mathcal{H}_0 \mid Y)\ |Y|^2 + P(\mathcal{H}_1 \mid Y)\ \widehat{\sigma_N^2}$$

A noise PSD estimate can now be obtained from a weighted average of the current estimated noise periodogram, $\widehat{|N(\ell)|^2}$, and the previously estimated noise power, $\widehat{\sigma_N^2}(\ell-1)$

$$\widehat{\sigma_N^2}(\ell) = \alpha\,\widehat{\sigma_N^2}(\ell-1) + (1-\alpha)\,\widehat{|N(\ell)|^2}$$

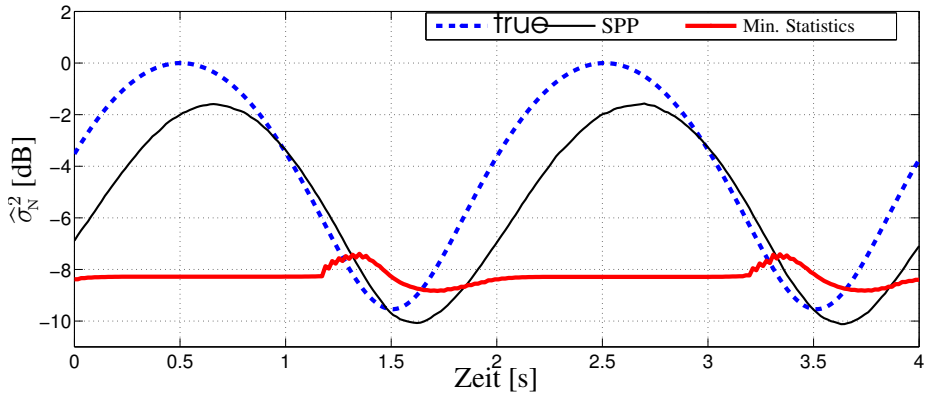


Figure 8.11: Comparison of noise tracking between the speech presence probability and minimum statistics approaches for a noise signal modulated at $0.5Hz$.

In practice, the weighting factor α is chose close to one, e.g. $\alpha \approx 0.8$, meaning that more emphasis is placed on past estimates of the noise power, σ_N^2.

The main advantage of using the speech presence probability for noise PSD estimation is its superior ability in tracking increasing noise levels when compared to the other methods. This can be seen in Figure 8.11 that compares noise tracking between the speech presence probability and minimum statistics approaches for a noise signal that is modulated at $0.5\,\mathrm{Hz}$. Because the latter can only search for the minimum within $1.5\,\mathrm{s}$ analysis windows, noise power bleeds into the enhanced speech due to a heavily underestimated the noise power. The speech presence probability technique more accurately tracks the noise resulting in less artifacts in the enhanced speech. This algorithm also exhibits a very low complexity making it advantageous in applications where battery power and computational speeds are limited.

8.3.4 Speech PSD estimation

One method of estimating the clean speech PSD is by using *maximum likelihood estimation* (MLE) which attempts to maximize the agreement of the model with the observed data. Let us assume we want to estimate the speech PSD if L succesive observations are given. Assuming that, given the speech PSD, these observations are independent the MLE problem can be written as

$$\sigma_{\mathrm{S}}^{2,\mathrm{ML}} \quad = \quad \arg\max_{\sigma_{\mathrm{S}}^2} \prod_{n=0}^{L-1} p(Y(\ell - n) \mid \sigma_{\mathrm{S}}^2)$$

Once again assuming the Gaussian model for the speech and noise PSD, the likelihood function is

$$p(Y|\sigma_S^2) = \frac{1}{\sqrt{2\pi(\sigma_S^2 + \sigma_N^2)}} \mathrm{e}^{\frac{|Y|^2}{2(\sigma_S^2+\sigma_N^2)}}$$

The goal now is to find the σ_{S}^2 that maximizes this function, and this can be done by taking the first derivative and setting it to zero.

$$\frac{\partial(Y|\sigma_s^2)}{\partial\sigma_s^2)} = 0$$

Solving the respective equations leads to

$$\begin{aligned}\sigma_{\mathrm{S}}^{2,\mathrm{ML}} &= \arg\max_{\sigma_{\mathrm{S}}^2} \prod_{n=0}^{L-1} p(Y(\ell-n) \mid \sigma_{\mathrm{S}}^2) \\ &= \frac{1}{L}\sum_{n=0}^{L-1} |Y(\ell-n)|^2 - \sigma_{\mathrm{N}}^2(\ell)\end{aligned}$$

If the operation is performed over only one frame, $L = 1$, then the MLE is the magnitude squared of the noisy observation minus the noise PSD. This is very similar to spectral subtraction that was introduced in the previous section meaning that under certain conditions, spectral subtraction can be interpreted as a MLE of the speech PSD. By averaging over multiple frames, the technique is made more robust by eliminating more outliers. However, in this form, the MLE is not typically used as it gives a very bad trade off with processing outliers and speech distortions.

What is more commonly used is the *decision directed* approach in which the current speech PSD is estimated from a weighted average of a single frame MLE, $L = 1$, and the magnitude-square of the previously estimated speech coefficients. Of course, in theory, the MLE term can never be less than zero, however it is constrained to be larger than zero to account for computational estimation errors.

$$\widehat{\sigma_{\mathrm{S}}^2}(\ell) = \alpha_{\mathrm{dd}}|\widehat{S}(\ell-1)|^2 + (1-\alpha_{\mathrm{dd}})\max\bigl(0, |Y(\ell)|^2 - \sigma_{\mathrm{N}}^2(\ell)\bigr)\,.$$

This is very similar to the MLE approach, however the averaging over successive frames is replaced by a non-linear and, in a sense, almost heuristic weighting. While being only one line of code, the decision directed approach is very successfully applied in practice.

8.4 Appendix: Statistical Signal Processing

In this section we recapitulate some basics of statistical signal processing.

The probability density function (PDF) is represented below. X is a random variable and x is some realization of this random variable. A uniform PDF would be a constant function of x meaning that all realizations of X are equally probable. Other common examples of PDFs are Gaussian and Laplacian.

$$\text{Probability density function:} \quad p_X(x)$$

Multidimensional PDFs can also be defined for functions of more than one variable. Below is the *joint probability density function* for two random variables, X, Y.

$$\text{Joint density function:} \quad p_{X,Y}(x, y)$$

There are rules regarding how these types of functions can be related. For instance, the *marginal density function* can be obtained from a joint density function by simply integrating over all realizations of one of the random variables.

$$\text{marginal density function:} \quad p_X(x) = \int_{-\infty}^{\infty} p_{X,Y}(x, y)\,\mathrm{d}y$$

It is now important to derive the concepts of expectation and moments. For this we employ the expected value operator which is defined below. It can be seen that if y is a function of x, then the expectation can be computed by two separate methods. If the PDF of X is known, then one can simply integrate over all realizations of the function of x weighted by their corresponding probability. If the PDF of Y is known, then the expectation is the integral over all realizations Y weighted by their corresponding probability. In the practice of deriving estimators, one PDF can be more difficult to obtain than the other and this principle becomes very useful.

$$\mathrm{E}(g(X)) = \int_{-\infty}^{\infty} g(x)\, p_X(x)\,\mathrm{d}x = \mathrm{E}(y) = \int_{-\infty}^{\infty} y\, p_Y(y)\,\mathrm{d}y$$

In practice, we often do not know the PDF of a random variable. By plotting a histogram of the signal samples, also the underlying PDF could be approximated. However, instead of estimating the histogram, it may be more efficient to directly estimate the moments using a sample mean. In doing so, the distribution of the signal is automatically incorporated into the mean as the samples are implicitly following the underlying PDF.

The moments of a random variable are typical expectations that characterize the statistical properties of a random variables.

$$n^{\text{th}} \text{ moment:} \quad \mathrm{E}(X^n)$$

The mean is the 1^{st} order moment of a random variable

$$\text{Mean, expectation of } X \text{ (first moment):} \quad \mathrm{E}(X) = \int_{-\infty}^{\infty} x\, p_X(x)\,\mathrm{d}x = m_{\mathrm{X}}$$

The centralized second order moment of a random signal is defined as the variance. The moment is called centralized when its mean is subtracted. Because speech and noise signals are typically zero-mean, the variance is equivalent to the power of the signal $\mathrm{E}(|X|^2)$. Because of the linearity of the mean operator and the Fourier transform, this same property holds in the spectral domain.

$$\text{Variance:} \quad \mathrm{E}\big((X - m_\mathrm{X})^2\big) = \mathrm{E}\big(X^2\big) - m_\mathrm{X}^2 = \mathrm{var}(X) = \sigma_X^2$$

$$\text{Standard deviation:} \quad \sigma_X = \sqrt{\mathrm{var}(X)}$$

The correlation of two real valued random variables can be described as the expectation of their product. For complex valued signals, the complex conjugate is taken of one of the signals. The cross-correlation is defined as

$$\text{Correlation:} \quad \mathrm{E}(XY^*) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xy^*\, p_{X,Y}(x,y)\,\mathrm{d}x\mathrm{d}y$$

The covariance, similar to the variance, is a centralized moment, meaning the mean is subtracted prior to multiplying x and y. A somewhat misleading point is that random variables are called uncorrelated when their covariance is zero. Of course, when we are dealing with zero mean signals, such as speech, then covariance and correlation are the same. However, when dealing with non-zero-mean signals, it becomes important to realize that two signals are uncorrelated when their centralized second order moment is zero, not the correlation. Thus, RVs with $\mathrm{cov}(X,Y) = 0$ are called *uncorrelated*.

$$\text{Covariance:} \quad \mathrm{cov}(X,Y) = \mathrm{E}((X - m_\mathrm{X})(Y - m_Y)^*)$$

9 — Speech Recognition

Learning objectives

- Introduction
- Feature Extraction
- Classification

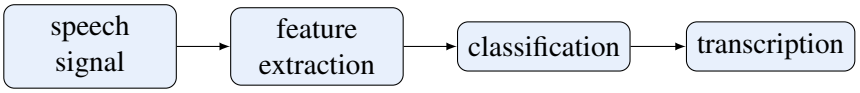


Figure 9.1: Processing chain of an automatic speech recognition algorithm

9.1 Introduction

The tools that humans have developed to improve our lives have been historically limited by the control and manipulation of their hands. Technology has a natural evolution towards replacing this hand control with more precise automated processes that are safer and more efficient. However, no matter how complex and long the processing chain, a human must still ”tell” the processor what to do. Whether this is performed by means of programming with a keyboard, moving a giant crane with a joystick, or texting with your thumbs, there is still hand control involved. A hands-free human-machine interface is thus desirable in which a machine can ”understand” verbal commands and perform the appropriate task. This is known as automatic speech recognition (ASR).

Although ASR represents a conceptually and computationally challenging topic in digital speech processing, it has made great strides in the past decade. Modern smart phones are now commonly equipped with ASR allowing for hands free control of basic tasks. Dictation using ASR algorithms has had successful applications in car communication systems. A dictation engine can transcribe human speech to text which is not only a more efficient method of transcription, but also safer than attempting to type while driving. Speech dialog systems are now commonly implemented by companies to reduce the required man power for customer phone calls regarding frequently asked questions. Another application that is of increasing interest is content analysis; the concept of big data. Bigger companies generally collect enormous amounts of data, however this data is useless if its content is unknown. The process of having humans analyze and label this data is very time consuming and often expensive, therefore a tool that can do it automatically is of very high value. For media data containing speech, audio cues can be used to label corresponding datasets. This can then even be used to train the recognizers themselves resulting in a constant improvement of the implemented ASR. Of course, surveillance is another application of ASR that also begins to show implications in security versus privacy issues. The technology allows for the monitoring of millions of cell phones conversations in which key words (e.g. “bomb”) can be flagged for further investigation.

Figure 9.1 details the ASR process. The first step is an extraction of the features that contain the most relevant information to discriminate between different phonemes. This is then followed by the classification step in which the closest match is found between features in the training database and the test case. This process returns a transcription which is the final chosen feature predicted by the model.

A simplified processing chain of an ASR algorithm is the comparison of test data to to a set of training data. For example, a recognizer could attempt to recognize the numbers from zero to nine. Representative training data would be then be recorded for each of the ten digits and the ASR algorithm would compare a recorded test utterance to the training data by using some user defined measure. This is similar to the process of vector quantization discussed in section 6.4 in which the Euclidean distance was used to find the nearest neighbor. Using this measure, the recognized digit would simply be the training dataset having the smallest Euclidean distance to the test data. This process is simple, and reasonably well to detect digits, however it has its limitations which will later become more apparent.

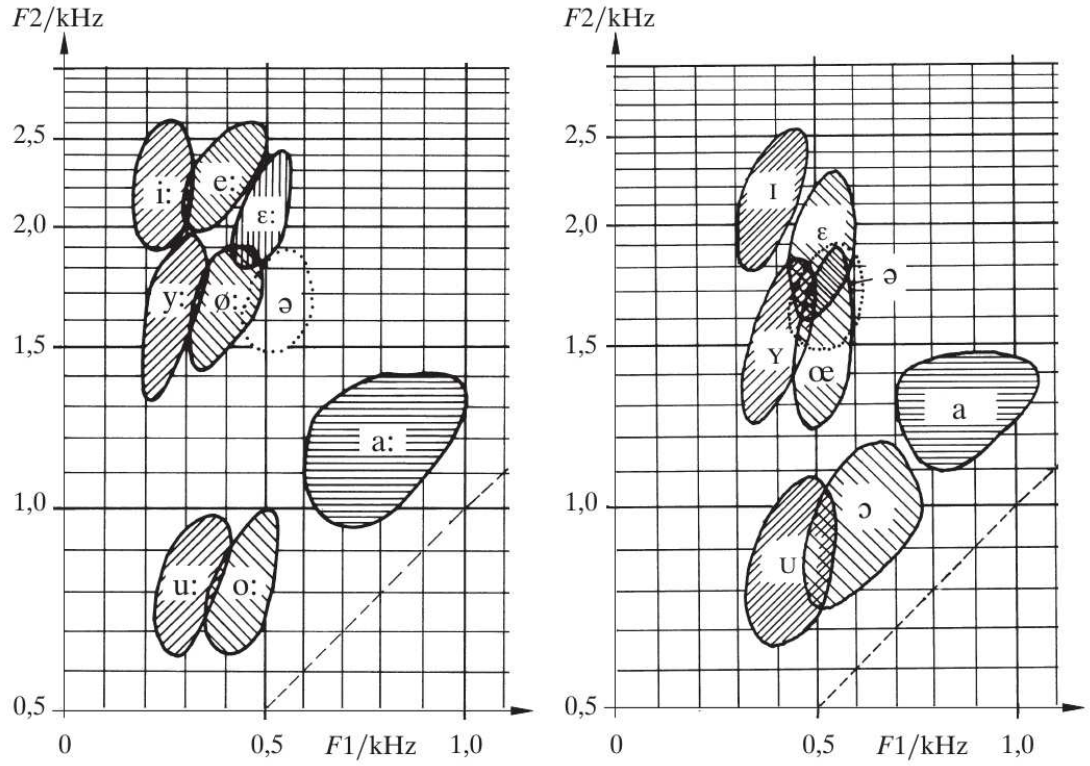
Modern recognizers work a bit differently by employing statistical models to create a statistical representation of words as opposed to the deterministic recording employed above. These recognizers are ”softer” because they build a variance around some mean value by training with many different versions of the same phonemes. In a statistical model, the measure between a test and training utterance is the likelihood that the two match. The training dataset that produces the highest likelihood would therefore be the recognized digit.

The first question we need to answer is: ”Upon what are we computing these comparison measures, i.e. what are adequate features?”. Different time domain recordings of the same word have such a huge variability that they prove to be inadequate. Speech spectrograms of the same utterance are different for different age groups and different genders. This variability is even further extended for a single user in which mood can produce fluctuations in tempo and level. If a general speech recognizer is to account for all of these variabilities, the ASR must be presented with adequate features that are robust with respect to all the differences that are present in speech. At the same time, irrelevant information should be discarded because it could affect the robustness of the recognizer. Furthermore, these features must also be robust in the noisy and reverberant environments in which ASR applications are commonly employed. The previous chapters have introduced the basic properties of speech and the speech production process. It was shown that the vocal tract filter coefficients contain the information necessary to distinguish between phonemes. This entails that our features should contain some representation of the spectral envelope and the formant positions.

Despite the advances made in ASR, we all know that the technology is still far from perfect. There are many major challenges that still persist. One major problem that is difficult to solve is that speech often contains ambiguities in which two sentences sound almost the same but can mean completely different things. This is exemplified by the two sentences ”How to wreck a nice beach” and ”How to recognize speech”. Continuity is also a major problem as there is often no clear separation between words in fluid speech.

The challenges caused by variabilities can be divided into to groups. Extrinsic variabilities include all external and environmental effects on the received speech signal. A recognizer that is trained in a normal room will have major difficulties with speech recognition in a church where the reverberation time is much different. This same principle holds for noisy environments such as a train or a cafeteria. It is therefore important to ensure that the features used in the recognizer are robust to these extrinsic variabilities.

The second group of challenges are intrinsic variabilities that exist for different (and even the same) speaker. The vocal tract of humans vary in shape and size and this results in spectral en-



Quelle: Vary, Heute, Hess (1998): Digitale Sprachsignalverarbeitung, Teubner, Stuttgart

Figure 9.2: Second formant frequencies of phonemes plotted as a function of first formant frequencies. The areas represent the variabilities that exist among different speakers. [7]

velopes thats differ among speakers for the same phoneme. The formants can differ in energy and position resulting in the different areas plotted in Figure 9.2. Pronunciation also differs among different speakers meaning that accents and dialects can seriously degrade ASR performance. Intrinsic variabilities are even further confounded because they exist for the same speaker. The emotional state of the speaker can introduce variabilities in the level (whispering vs. screaming) and timing (prosody) of the same utterance that can have detrimental effects on a recognizer. Solutions to the timing issue will be discussed in a forthcoming section.

9.2 Feature Extraction

The previous section introduced the intrinsic and extrinsic challenges that are faced by an ASR system. To minimize their effects it is critical, to choose features that are robust and sensitive to discriminating different phonemes. Because the formant map in Figure 9.2 shows very little overlap for different phonemes, it is reasonable to assume that F1 and F2 are good features to

Figure 9.3: Processing chain for the computation of Mel frequency cepstral coefficients.

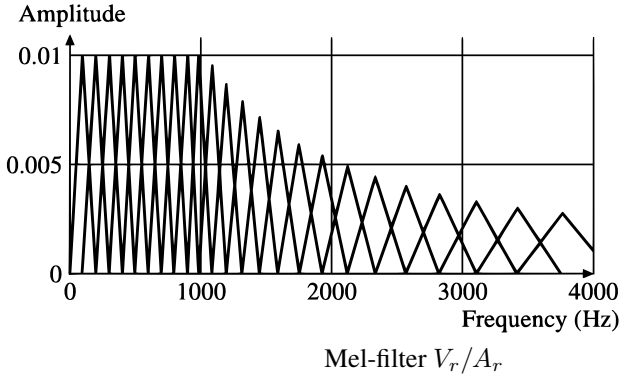


Figure 9.4: Spectral domain representation of a Mel-frequency filter bank.

input to the ASR classifier. However, it is extremely difficult in practice to robustly estimate the first and second formants. The LPC analysis method that was introduced in Chapter 4, turns out to be not very robust with respect to noise. A tool that has proven to be much more powerful is the Mel-frequency cepstral coefficients (MFCC).

Figure 9.3 details the process for extracting the MFCC. A STFT is first produced by transforming each of the windowed time domain signal segments into the spectral domain. Using a common FFT size results in 257 coefficients for each time segment, upon which a periodogram is then computed. The signal is then filtered by multiplying the spectrum of each consecutive frame by the Mel-filter bank shown in Figure 9.4. A typical implementation averages the periodogram coefficients to obtain 24 mel spectral coefficients. The cepstrum is then computed by taking the inverse discrete cosine transform (IDCT) of the logarithm of each segment's spectrum. After a cepstral liftering process, the number of coefficients is even further reduced to a typical number like 13. Although there are other motivations behind each step that will be discussed below, the reduction of features is a nice benefit that allows for a compact representation of the current frame.

One motivation behind the usage of this particular filter bank is that it groups the signal into bands that are similar to the filters that are employed in auditory perception. The basilar membrane has a tonotopic frequency representation that produces a frequency dependent frequency resolution, resulting in a low frequency resolution (broad filters) in high frequencies and comparatively high frequency resolution (narrow filters) at low frequencies. This is quantitatively represented by the ERB and the Mel scales. The Mel scale is empirically constructed from experiments with human pitch perception, but it is also essentially related to more loudness based scales (e.g. Bark scale), due to the non linear representation on the basilar membrane. The Mel filter bank in Figure 9.4 attempts to model this behavior. The signal-processing perspective is that the Mel filter bank will tend to average out harmonic components. This is good, because phonemes are the same regardless of pitch, a good speech recognizer should thus be be pitch invariant.

The output of the Mel filter bank is usually transformed into the cepstral domain by using the inverse discrete cosine transform (IDCT) as opposed to the IDFT with which we are familiar.

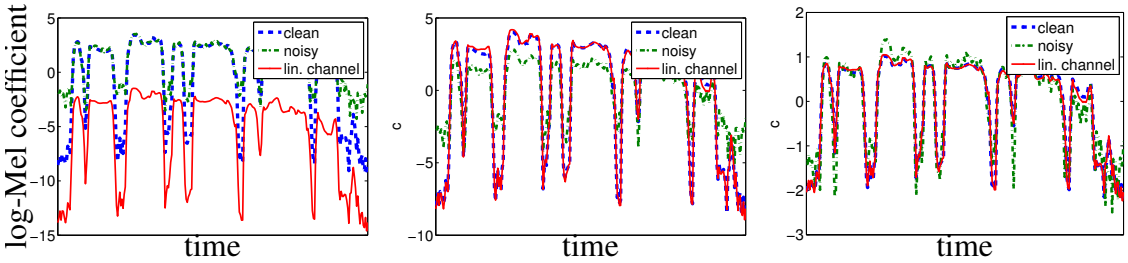


Figure 9.5: (Left) Channel effects on the time evolution of a MFCC coefficient of clean and noisy speech. (Middle) Effects after mean normalization. (Right) Effects after mean and variance normalization.

One of the motivating reasons behind this is that the log-periodogram is real-valued and even, meaning that all information can be captured by correlating with cosines. This results in a reduction in computational complexity allowing for increased speed in recognition tasks. One of the motivations for using a spectral transfrom such as the IDCT is that it decorrelates signal coefficients which allows for a simplified statistical representation. Time domain speech signals have a high degree of correlation that is greatly reduced upon transformation into the spectral domain. This correlation is even further reduced upon taking the IDCT, such that the cepstral covariance matrix can be approximated by a simple diagonal covariance matrix. This allows for significant savings in computational complexity when employing a statistical multivariate prediction model.

A further motivation from going from the mel-spectral domain to the MFCC domain is that the remaining pitch information can be reduced further by keeping only the lowest 13 MFCCs. This process is called cepstral liftering. The result produces a smoothed spectral envelope with features containing the necessary information for the ASR process. A third motivation behind using the cepstral coefficients is that they are level independent (except for the zeroth coefficient). The zeroth cepstral coefficient captures the mean value of logarithmic mel spectral coefficients. Therefore, any scaling of the signal does not affect the remaining coefficients that are used in the classification step. These feature of MFCCs are very helpful in dealing with the extrinsic and intrinsic variabilities that were previously discussed.

MFCC performance can be boosted further by creating some representation of the temporal structure of the utterance. For this, often the difference between two successive feature vectors (ΔMFCCs) are used as additional features. Also $\Delta\Delta$MFCCs are typically incorporated, i.e. the difference between successive ΔMFCCs.

There are even more ”tricks” that we can implement to further increase the robustness of the recognizer. Noise is an extrinsic problem that can be limited with the speech enhancement algorithms discussed in Chapter 8. Reducing noise before performing feature extraction can boost the performance, but there are other, more commonly implemented methods that are simple and more efficient. Multi-condition training, for example, can be employed to combat intrinsic and extrinsic variabilities. With this method, training data can include mixtures of clean speech, noisy speech, reverberant speech, different dialects, etc. A statistical model can then be employed that quantifies the variabilities of the training data, thus producing features with a higher level of robustness.

Another commonly implemented method is *cepstral mean and variance normalization*. Any channel through which speech is processed can be modeled as a filter with a corresponding

impulse response and transfer function. Speech processed through such a channel undergoes a time domain convolution or frequency domain multiplication. In the log domain, this results in an addition of a constant channel offset that should be accounted for to increase the robustness of the recognizer.

Figure 9.5 (Left) shows the time evolution of a log-mel spectral coefficient (a certain frequency band). In practice, mean and variance normalization is performed on MFCCs, but a log-mel spectral coefficient was chosen to simply the interpretation. Larger values reflect speech activity while lower values reflect speech inactivity. It can also be seen that noisy speech tends to fill in the gaps of speech activity, thus reducing the dynamic range of the signal. This can have adverse effects on the representation of the features.

To combat channel offsets, a mean normalization can be implemented in which a global empirical mean is taken over the entire utterance and subtracted from the features. This method is quite simple and works well as long as the length of the channel impulse response is shorter than the segment length. The effects can be seen in Figure 9.5 (Middle) which also shows that the dynamic range of the noisy speech is still reduced. This is solved by implementing a *variance normalization* in which the mean normalized features are also divided by the global empirical variance thus resulting in features that are more similar over time. The ASR algorithm would then be trained on features that were mean and variance normalized resulting in a recognizer that is much more robust.

9.3 Classification

In the previous section, MFCCs were introduced as a method for extracting features from a given input signal. Further methods were also discussed for making these features robust against the extrinsic and intrinsic variabilities that plague an ASR system. The next step in the process is *classification* in which these features are compared against a training database. In the simplest case, the test data would be compared with all of the reference models, and the one with the closest match would correspond to the recognized digit.

In the classification step, it is important to distinguish between a *local* and *global* match. A local match is a match between the feature vector computed from the observation of a single frame of test and training data. This could be one feature vector of thirteen MFCC coefficients matched with a training feature observation by using a statistical model or possibly a simple Euclidean distance. A global match is the overall distance between a set of feature vectors and the training data. This addresses the problem that extrinsic and intrinsic variabilities produce test and training utterances of different lengths. This problem can be handled e.g. by using *dynamic time warping* (DTW) or *hidden Markov models* (HMM).

Dynamic Time Warping

The idea behind dynamic time warping is to temporally warp the test utterance against each training utterance in an effort to force them to be the same length. As the warping is performed, a Euclidean distance measures the local distance between all the test and training data segments. The utterance yielding the smallest global distance is chosen as the match. So we begin with a set of feature vectors of Mel cepstral coefficients that represent a word in our training database in which l is the segment index and n is the index of an MFCC.

$$x_n(l)$$

A word is then recorded by the recognizer. It is windowed and a feature extraction is performed on each segment to obtain

$$y_n(l)$$

.

The task at hand is to find the reference utterance that is most similar to the observed sequence by using some quantitative measure. The simplest solution would be to compute the Euclidean distance between corresponding frames of the test and training feature vectors.

$$d(\hat{x}, \hat{y}) = \sqrt{\sum_n (x_n - y_n)^2}$$

However, this method fails because different speaking tempos will result in different lengths, and thus a different number of feature vectors for test and training data. In addition, because the test utterance must be compared against the entire training database, it will be compared against utterances that are naturally of different length. The solution is to apply a warping of the signals to make them the same length. The simplest implementation would be a linear time warping in which one of the signals is stretched to be of the same length as the other. However, the lengths of phonemes within each utterance are also quite variable which causes this idea to also fail.

Instead, a dynamic time warping is implemented in which a distance measure is computed between each of the test segments and all of the reference segments to produce a huge matrix that indexes this similarity. The optimal path through this matrix of distances yields a global difference measure that can be calculated for all reference utterances in the training set.

The method of determining the optimal global path will be introduced by using the analogy of computing the cheapest route from Oldenburg to New York City. Because it is not possible to go directly from Oldenburg to NYC, there are a set of different combinations that can be used with each corresponding to a different cost. For speech recognition, the lowest cost is all that is of interest because this corresponds to best temporal fit between the test and reference data. In this context, it is not necessary to find the optimal path, or the route associated with the lowest cost, however we will still examine this to analyze how the signals must be warped against each other to match them temporally.

Figure 9.6 introduces an example with different possibilities of traveling between Oldenburg and NYC. There is a possibility to take a train to Bremen or to Hanover. From there, a flight can be

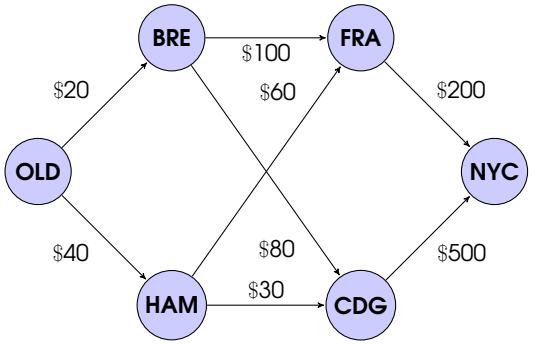


Figure 9.6: There are a set of different combinations that can be used to travel from one place to the other. In ASR, it is desirable to choose the combination that minimizes the overall cost.

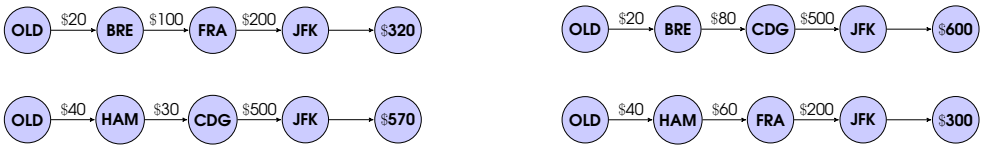


Figure 9.7: Brute force method for determining the optimal path and minimal cost.

taken to NYC with a stop over in either Frankfurt or Paris. In each of these tracks is assigned a certain cost. In the speech recognition context, this cost corresponds to the Euclidean distance between the feature vectors for all segments in the test and the reference utterance.

The simplest method of finding the minimal cost is to implement a brute force solution that computes the price of all possible combinations (Figure 9.7). This process always works in practice, but it requires a lot of computation. In a speech recognition example, the number of segments in each utterance is quite high and this results in many more combinations than are shown in Figure 9.7. The result is a computational complexity that is inefficient and difficult to realize in practice.

Instead, a technique called *dynamic programming* is implemented. Using the traveling example in Figure 9.6, we must compute the cheapest route from the predecessor and the corresponding cumulative price. The first step is simple as there is only one way to get from Oldenburg to Bremen or Hanover which costs $20 and $40, respectively. However, getting to Frankfurt involves two possibilities: Frankfurt over Hanover with a cumulative price of $100 and Frankfurt over Bremen with a cumulative price of $120. At this point, we only record the cheapest route and its respective cumulative cost (Figure 9.8). This same step is also repeated for Paris. To get to

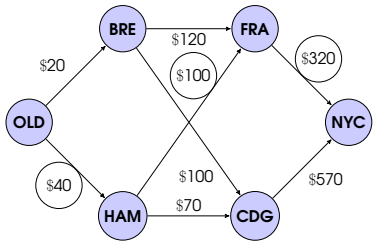


Figure 9.8: Dynamic programming method for determining the optimal path and minimal cost.

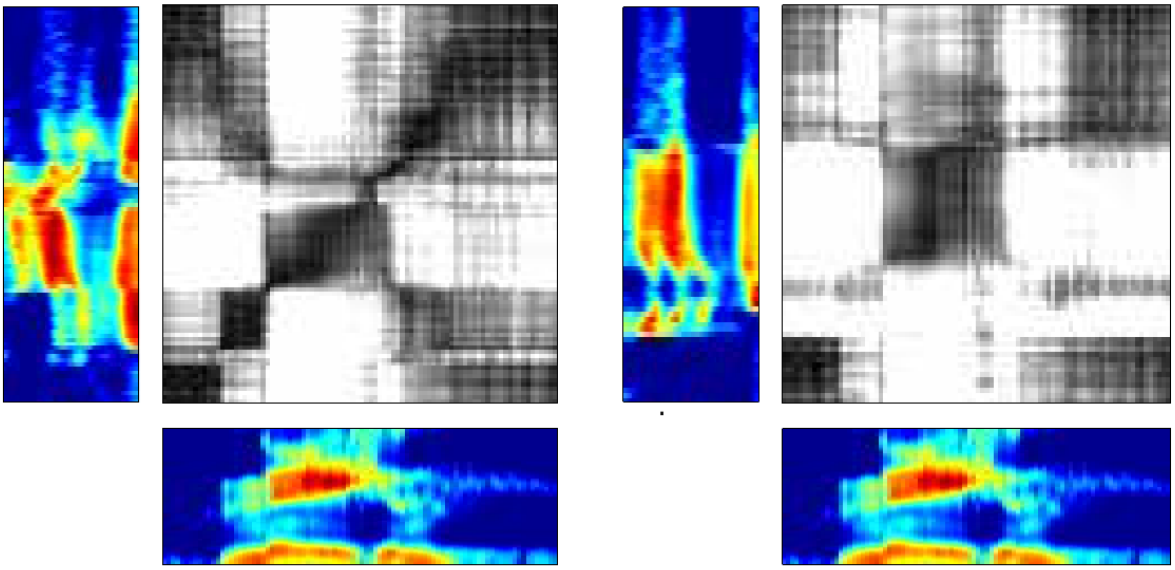


Figure 9.9: Example of the distance matrix between a test utterance (X-axis) and two different reference utterances (Y-axis).

NYC, there are only two possibilities.

Although it is cheaper to travel from Oldenburg to Bremen, the route from Oldenburg to Hamburg minimizes the overall global cost. In order to find the optimal route, it is therefore necessary to backtrack. Therefore, if it is desired to know the optimal path, it is also necessary to record how one got to each location.

This analogy can now be used to develop the dynamic time warping algorithm in the speech processing context. The first step is to compute the Euclidean distance between all feature vectors of the test and reference data. These feature vectors could consist of the thirteen MFCC coefficients calculated for each frame index l of the test utterance and each frame index λ of the reference utterance.

$$d(l,\lambda) = ||\hat{x(l)} - \hat{y(\lambda)}|| = \sqrt{\sum_n (x_n(l) - y_n(\lambda))^2}$$

The Euclidean distance produces a single value for each combination of frames of the test and reference utterances. Each value is then used to create the distance matrix that is shown in Figure 9.9. The goal now is to compute the minimal cumulative cost required to get from the beginning of the test utterance to the end. Dynamic time warping is the process of finding the optimal path through this matrix and this will determine how to stretch the test utterance to best match to the reference utterance. This process must then be performed for each of the reference entries in the database and the lowest cumulative cost will correspond to the best match and the utterance recognized by the ASR system.

In order to compute the cumulative price as was done in the traveling example, the optimal path must be found by navigating through the distance matrix. If N is the length of the test utterance and M is the length of the reference utterance, then this path will begin at $d(1,1)$ and end at

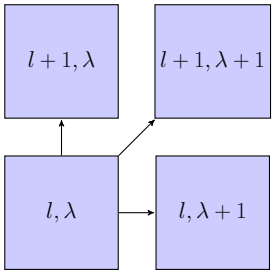


Figure 9.10: Steps permitted in navigating through the distance matrix.

$d(N, M)$. It is first necessary to constrain the steps that can be made through the distance matrix. Because speech progresses forward in time, we cannot travel backwards through the distance matrix. Therefore there are only three different steps shown in Figure 9.10 that can be made. In this example, a horizontal movement entails that the reference utterance is slower than the test utterance at that particular point. The opposite is true for vertical movements.

To compute the global cumulative cost, it is necessary to compute the cost of arriving at each position and then recording the path required to get there. This is done by introducing a cost matrix, c, that stores the cumulative costs for the optimal path to each location in the distance matrix, d. The cost matrix is built from the distance matrix by initializing with:

$$c(1,1) = d(1,1)$$

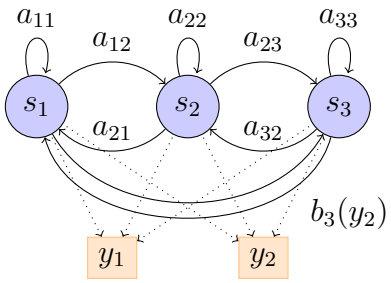
Because there is only one way of making a horizontal or vertical shift, the values of the cost matrix at these positions are

$$\begin{aligned} c(2,1) &= c(1,1) + d(1,2) \\ c(1,2) &= c(1,1) + d(2,1) \end{aligned}$$

Diagonal shifts are a bit more complicated. There are several possibilities of getting there: diagonally, vertically, or horizontally. As in the airport example, only the path with the lowest cost will be recorded in the cost matrix. Although it is not necessary for the ASR system, the chosen path can also be recorded so that the optimal path through the distance matrix can be recreated by backtracking. A heuristic penalty is applied to diagonal steps so that the diagonal transition is not always chosen as the optimal route. This value can be tuned to optimize the algorithm, but two is a reasonable choice that works quite well.

$$\begin{aligned} c(2,2) &= c(2,1) + d(2,2) \\ c(2,2) &= c(1,2) + d(2,2) \\ c(2,2) &= c(1,1) + 2d(2,2) \end{aligned}$$

These steps can be used to write an algorithm that processes the entire distance matrix and creates a cost matrix in which each value represents the lowest cost required to reach the corresponding



Example: Weather

States:
s_1: cloudy
s_2: rainy
s_3: sunny

Observations:
y_1: man with umbrella
y_2: man without umbrella

Figure 9.11: Example of a hidden Markov model used in simplified weather prediction.

position. In addition, by saving the path, backtracking can also be used to find the optimal warping of the two signals. Using this information, the spectrograms can be reshaped to analyze the differences in the test and reference utterances.

However, for the recognition example, the important value is located at $c(N, M)$ which represents the cumulative costs of the total optimal path. This value is also referred to as the dynamic time warping distance (DTW distance). It is computed for all reference utterances in the database and the one corresponding to the lowest DTW is chosen by the ASR system as the recognized word.

Hidden Markov models

Commonly ASR processing schemes implement more statistical approaches like hidden Markov models (HMMs) and deep neural networks (DNNs) to describe the variable flow of speech. For HMMs, probabilities are used to describe the transition from one phoneme to the next. A Markov chain is a chain of states in which the probability at some future state is only dependent on the previous state. In this model, it is not necessary to look into all of the past transitions that could be made, but only consider the last state of the system. These states are hidden because they can not be directly observed. Instead, we are given some observations that allow for a prediction of the current state.

Figure 9.11 shows an HMM example in which there are three states with transition probabilities from one state to the next. The probability of advancing from s_1 to s_2 is given by a_{12}. The probability of returning to s_1 from s_2 is given by a_{21}. It is also possible to remain in s_1 with a probability of a_{11}. We are given some observations y_1 and y_2. For each state, there is a certain likelihood to generate a particular observation. This is also referred to as a *generative model*.

The HMM depicted in Figure 9.11 considers forecasting the weather based upon one of two possible observations. In this example, there are only three states: cloudy, rainy, sunny. However, perhaps you are in an office and cannot directly observe the state of the weather. Instead, you must determine the probability of a particular weather condition based upon the observation of your co-workers carrying an umbrella. Probabilities can be established between each possible observation and all states. For instance, the probability that we observe a co-worker without an umbrella when it is sunny is quite high while the probability that we observe a man with an umbrella when its sunny is rather low. This can also be done for transitions between states. For

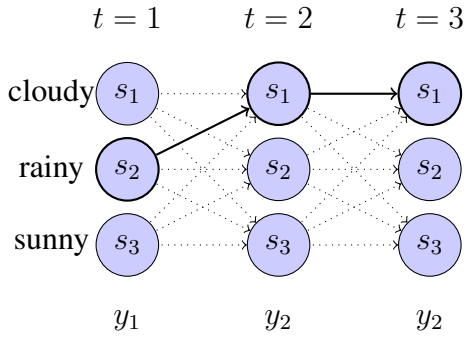


Figure 9.12: A trellis diagram representation of Figure 9.11

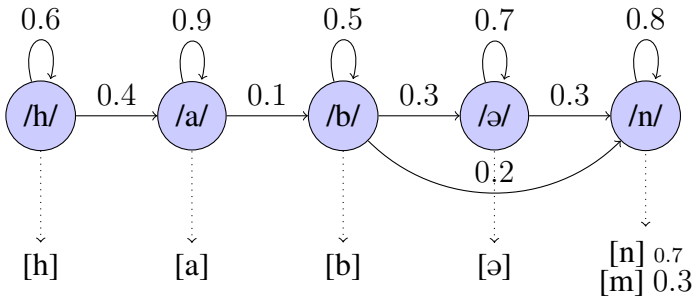


Figure 9.13: HMM implementation in speech recognition.

example, if it is sunny, then there is a low probability that it will be directly rainy. It is more likely that it will be cloudy or that it will stay sunny. All of these probabilities are established in a training process that uses a large amount of data. Figure 9.11 can also be represented by the *trellis diagram* depicted in Figure 9.12. In this representation, the temporal progression of state transitions is shown.

The application of HMM to speech recognition is now rather intuitive. Each of the states could correspond to a phoneme and a large volume of labeled training data would allow for the computation of the transition probabilities. Similarly as for DTW, we assume that we can only stay in the same state or move forward to the next state. Figure 9.13 shows an example of an HMM used to model the word ˊhabenˊ. This model shows that there is a higher probability of staying in the /a/ state then moving on to the /b/ state. There is also the ability to move directly from the /b/ state to the /n/ meaning that the model can account for the shortened ˊhabnˊ. This ability allows the model to handle variabilities that cannot be dealt with using the dynamic time warping algorithm.

A training dataset would include an HMM for each of the words in the lexicon of the recognizer. Once again the test utterance must be compared with each of the models in the reference dataset. The model makes a prediction by producing a probability that the observation fits to each reference and then choosing the most likely match.

List of Figures

Figure Sources

[7] P. Vary, U. Heute, and W. Hess, *Digitale Sprachsignalverarbeitung*. Teubner, Stuttgart, 1998.

[8] P. Vary and R. Martin, *Digital speech transmission*. Wiley, 2006.

[9] Wikimedia Commons. (2007). Places of articulation. created by Ishwar. Permission: GNU FDL, CC-BY-SA-2.5, [Online]. Available: `https://commons.wikimedia.org/wiki/File:Places_of_articulation.svg`.

[10] ——, (1918). Saggital mouth. From Gray's Anatomy 1918 edition. Permission: Public Domain, [Online]. Available: `https://commons.wikimedia.org/wiki/File:Sagittalmouth.png`.

[11] International Phonetic Association. (2005). The international phonetic alphabet. Permission: Creative Commons Attribution-Sharealike 3.0 Unported License (CC-BY-SA), [Online]. Available: `https://www.internationalphoneticassociation.org/content/full-ipa-chart`.

[12] Wikimedia Commons. (2009). Anatomy of the human ear. created by Chittka L, Brockmann. Permission: Creative Commons Attribution 2.5 Generic license., [Online]. Available: `https://commons.wikimedia.org/wiki/File:Anatomy_of_the_Human_Ear_en.svg`.

[13] ——, (2010). Cochlea-crosssection. created by Oarih. Permission: Creative Commons Attribution-Share Alike 3.0 Unported license., [Online]. Available: `https://commons.wikimedia.org/wiki/File:Cochlea-crosssection.svg`.

[14] ——, (2006). Audible. created by Booby. Permission: Public domain., [Online]. Available: `https://commons.wikimedia.org/wiki/File:Audible.JPG`.

[15] ——, (2008). Uncoiled cochlea with basilar membrane. created by Kern A, Heid C, Steeb W-H, Stoop N, Stoop R. Permission: Creative Commons Attribution 2.5 Generic license., [Online]. Available: `https://commons.wikimedia.org/wiki/File:Uncoiled_cochlea_with_basilar_membrane.png`.

[16] A. d. Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," vol. 111, no. 4, pp. 1917–1930, Apr. 2002.

References

[1] A. d. Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," vol. 111, no. 4, pp. 1917–1930, Apr. 2002.

[2] P. Vary, U. Heute, and W. Hess, *Digitale Sprachsignalverarbeitung*. Teubner, Stuttgart, 1998.

[3] P. Vary and R. Martin, *Digital speech transmission*. Wiley, 2006.

[4] J. Deller, J. Hansen, and J. Proakis, *Discrete-time Processing of Speech Signals*. IEEE Press, 2000.

[5] R. Hendriks, T. Gerkmann, and J. Jensen, *DFT-Domain Based Single-Microphone Noise Reduction for Speech Enhancement - A Survey of the State of the Art*. Morgan & Claypool, 2013.

[6] W. B. Snow, "Change of pitch with loudness at low frequencies.," *Journal of the Acoustical Society of America*, vol. 8(1), pp. 14–9, 1936.