# Introduction to Speech Processing

**Tom Bäckström, Okko Räsänen, Abraham Zewoudie, Pablo Pérez**

**Apr 28, 2022**

# CONTENTS

**2nd Edition**

This is an open access and creative commons book of speech processing, intended as pedagogical material for engineering students.



# Contributing

We welcome all type of useful contributions! Content contributors will, unless author requests to remain anonymous, be automatically included on the list of authors and for bugfixers we will create an "also featuring" list. The best way of contributing however depends on type of contribution:

- For small bugfixes
  - Post an issue on gitlab.com, or
  - Modify the code directly and push a merge request.
- For clearly missing or incomplete content
  - Create the content and push a merge request.
- Extensions of topic area
  - Discuss it first with other by posting an issue on gitlab.com.

The source material is stored at https://gitlab.com/speech-interaction-technology-aalto-university/itsp.

See also Instructions for developers.

## Referencing

> Tom Bäckström, Okko Räsänen, Abraham Zewoudie, Pablo Pérez Zarazaga, Liisa Koivusalo and Sneha Das, "*Introduction to Speech Processing*", 2nd Edition, 2022. URL: https://speechprocessingbook.aalto.fi

Bibtex format:

```
@book{itsp2022,
   title = {Introduction to Speech Processing},
   edition = 2,
   year = 2022,
   author = {Tom Bäckström and Okko Räsänen and Abraham Zewoudie and Pablo Pérez␣
 ↪Zarazaga and Liisa Koivusalo and Sneha Das},
   url = {https://speechprocessingbook.aalto.fi},
   }
```

CHAPTER

ONE

# PREFACE

This is a collection of pedagogical material within the topic of speech and language technology. The idea is to provide

- teachers material for their courses, where they can pick and choose material which is appropriate for their own courses.
- self-study material on-line for anyone interested.

By licensing the material under creative commons (share alike), we want to encourage people to contribute improvements and additions to the content.

## 1.1 Design philosophy of this document

- Target audience
  - Primary target:
    - Master's level students with some background in digital signal processing (=signals and systems), machine learning, linear algebra and stochastic processes
  - Secondary targets:
    - Researchers in related areas who want to expand their expertise, for example, researchers in machine learning, signal processing, audio processing, linguistics, human-computer interfaces etc
- Keep a pedagogical approach
  - Explain why this tool is needed and how it solves a problem.
  - Give an example of how it is used in practice, including demonstrations and pictures.
  - Favour tools which everyone are using, rather than inventions of your own team
- Small steps
  - Better to have something than nothing. Though it is a good direction, reaching perfection is not a requirement.
  - Fix errors immediately when you find one.

## 1.2 Foreword to the First Edition

Foreword by Tom Bäckström

As I was teaching the course "Speech processing" at Aalto University, I was always looking for good teaching material. I was not really content with what I found. Some good books were available but they were expensive. I was not comfortable with demanding the students to pay hundreds of Euros for a book they'd use once. I was also not comfortable in illegally copying content. The alternatives were then to accept lower-quality material or write my own.

Part of the issue I have with expensive books is that the money does not go to the authors themselves, but to middle-men. Moreover, in the Internet-era, paper books seem so last-century. Why print a book on paper when we can make it a web document? Why put it behind a pay-wall? I mean, I really would not receive any significant part of my income from such a book. Putting it on the web then seems like the only sane solution.

Besides, once you're free from the constraints of a conventional book, you can do all kinds of fun stuff. Like why would I limit access to modifying content and why not something more wikipedia-like? I'm paid by the government, so it seems also obvious that I should put my work out in the public domain. No, more accurately, I'm putting this out with a Creative Commons licence (attribution & share-alike). Perhaps it's vanity, but I would like to receive credit for this work, if there is any credit due.

The desired consequence of Creative commons licensing is that the material would find multiple contributors, to improve the content. To follow the old, worn but accurate adage; to stand on the shoulders of giants, and so forth. By collaboration we can do better.

The way I intend to use this in my own teaching is that the on-line version of the document follows its own natural grouping of topics. Start with basics and progress to more complex topics and applications. For my own, course, however, I want to have exercises in parallel with the course. The problem is then that the most basic chapters do not lend themselves to exercises which are useful for my teaching goals. So I design exercises to match my teaching goals and organize lecture material to give sufficient background to the exercises. In this web-based document this is no problem. I'll just create a new table of contents, where the ordering of chapters and sections is reorganized.

## 1.3 Foreword to the 2nd Edition

Foreword by Tom Bäckström

I have been positively encouraged and surprised by the feedback I have received for the first edition. So many people have spontaneously given feedback. I find it safe to assume that many more have used this material than those who have contacted me. I therefore conclude that the impact of this material has been much larger than I anticipated. Great! This encourages me to continue putting effort into the document, to make it better and expand it.

For the second edition, I wanted to address the following issues:

- Platform:
  - Integration with JupyterLab and other similar platforms for interactive coding examples and visualizations are not easily possible on the original platform. Still, in my own teaching I have found such tools mmensly effective and popular among the students.
  - Though the material was published as Open access with a Creative Commons license, the original platform did not allow for easy porting to other formats. Especially mathematical notation and equations required extra effort when porting. This is a clear contradiction to our open access intentions and desires.

  Clearly both arguments lead to the conclusion that we have to switch platforms. Delaying the switch further will make it only harder. Currently the dominant way of sharing evovling community projects is git and consequently, that is the obvious choice. Additional benefits from git-based platforms is that they have many practical tools integrated, like merge-requests, discussion boards etc.

- Content:
  - Machine learning in speech processing was not well-enough represented. Have to add and improve the content in that area.
  - I am developing and have recently particpated in generation of additional content both for a Bachelor-level introductory course as well as a course about design of speech interaction technology. Those should be added here.
  - There are also many other areas which would benefit from additions, like speech recognition and NLP. I hope we find someone to contribute material also there.

CHAPTER

TWO

# INTRODUCTION

1. *Why speech processing?*
2. *Speech production and acoustic properties*
3. Speech perception (Wikipedia)
4. *Linguistic structure of speech*
5. Speech-Language pathology (Wikipedia)
6. Applications and systems structures
7. Social and cognitive processes involved in human communication (external)

## 2.1 Why speech processing?

### 2.1.1 Why speech?

Speech is our primary mode of communication; When you want to communicate something important, you say it face-to-face. Think about your first "I love you", your last job interview and a nice evening with friends. Everything *important* is communicated in a spoken form.

Speech is *about* communication. A characteristic trait of humans in comparison to other animals is our refined abilities to communicate. To work efficiently as a group, we need to communicate. To learn from our mistakes, we need to communicate. Where hand waving and smoke signals can be used to communicate, speech remains as our best way to communicate abstract thoughts.

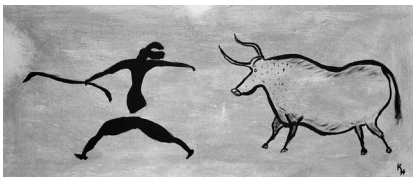
However, a common idiom is "*a picture is worth a thousand words*". It is also the reason why this document has pictures on the side. They help in capturing the essential information. An important difference between speech and images is however that where pictures excel in transmission of information, speech excels in interaction. The game "Pictionary" is fun because interaction through a picture is difficult.

Speech interaction is part of a large research in its own right (see e.g. the book Message processing).

### 2.1.2 Early communications technology

The expressive power of speech is tremendous, it is a powerful tool for interaction, but in early human cultures, it was difficult to *store* information. Story-telling was a way to memorize history, but our capability to accurately reproduce stories is limited.

Cave paintings was an early way to store information more permanently, and this technology later evolved to stone tablets, papyrus and paper letters. Such ancient documents provide the most accurate information we have about our past. We would not know about Socrates, without the writings by Plato. Innovations in communications technology, such as cave paintings, book printing and the Internet, have been so important that they characterize historical eras.



### 2.1.3 Evolution of speech technology

Telecommunications was another milestone in human history. Though the telegraph was an effective way for communicating, it also required specialized training. The invention of the telephone in 1849 was therefore a great invention because it was the first technology to provide instantaneous telecommunication without specialized training.

The first wireless (radio) transmission of speech came 50 years later in 1900, quickly to become an important broadcast media. Again, while newspapers had an important role in broadcasting news, the radio was faster and more accessible (does not require the ability to read).

Another important step was the introduction of mobile phones in the 1990's. The importance of its impact is easily demonstrated by the changes in our behaviour which are a consequence of the new technology:

- Before, we would agree on a specific time to talk on the phone - "I'll call you at home around 18 o'clock.". The other party would then know to stay at home waiting for the phone call. Today we just say "I'll talk to you later". There is no need to know where the other person is and we also do not agree on a specific time.
- Before, we would agree on a specific time and place where to meet - "I'll meet you at the main building at 12:15." Both people would then adjust the timing of their arrival to match the agreed time. Today, we can just say "I'll call you when I'm nearby."

In both cases, we are more flexible in our scheduling, making for more efficient use of time.

### 2.1.4 Further development

We have thus determined that speech is an important and powerful mode of communication for humans. For improving technology, this gives two prominent opportunities;

- If we can make communication with speech easier using technology, it can be very useful. For example, if telecommunication, such as telephony, teleconferences, and voice-over-IP, can be improved, then that would allow people to use speech more efficiently.
- We can use to our advantage the people's preference of speech communication. For example, interactions with devices and computers could be improved by allowing spoken interaction with them. In particular, typing on a keyboard and other tactile interfaces are difficult for children, the elderly and handicapped people, whereas a majority of people (but not all) can speak. Similarly, user interfaces based on visual information is often based on accessing information and services through menus. Using natural language can be more intuitive and simple to use; we could just say to the washing machine "Wash this small amount of dirty curtains. " instead of searching for washing options from a menu.

The devices and services which use speech and language are extremely wide-spread. By now, a majority of people in the world has access to a mobile phone and there are almost 8 billion active mobile-phone subscriptions. If we can improve the technology used by those 8 billion people, by say, reducing energy consumption, then the impact of such improvements would be majestic.

conda install -c conda-forge jupyter_contrib_nbextensions # Speech production and acoustic properties

## 2.2 Physiological speech production

### 2.2.1 Overview

When a person has the urge or intention to speak, her or his brain forms a sentence with the intended meaning and maps the sequence of words into physiological movements required to produce the corresponding sequence of speech sounds. The neural part of speech production is not discussed further here.

The physical activity begins by contracting the lungs, pushing out air from the lungs, through the throat, oral and nasal cavities. Airflow in itself is not audible as a sound - sound is an oscillation in air pressure. To obtain a sound, we therefore need to obstruct airflow to obtain an oscillation or turbulence. Oscillations are primarily produced when the *vocal folds* are tensioned appropriately. This produces *voiced sounds* and is perhaps the most characteristic property of speech signals. Oscillations can also be produced by other parts of the speech production organs, such as letting the tongue oscillate against the teeth in a rolling /r/, or by letting the uvula oscillate in the airflow, known as the uvular trill (viz. something like a guttural /r/). Such trills, both with the tongue and the uvula, should however not be confused with voiced sounds, which are always generated by oscillations in the vocal folds. Sounds without oscillations in the vocal folds are known as *unvoiced sounds.*

Most typical unvoiced sounds are caused by turbulences produced by static constrictions of airflow in any part of the air spaces above the vocal folds (viz. larynx, pharynx and oral or nasal cavities). For example, by letting the tongue rest close to the teeth, we obtain the consonant /s/, and by stopping and releasing airflow by closing and opening the lips, we

obtain the consonant /p/. A further particular class of phonemes are nasal consonants, where airflow through the mouth is stopped entirely or partially, such that a majority of the air flows through the nose.

### 2.2.2 The vocal folds

The *vocal folds,* also known as vocal cords, are located in the throat and oscillate to produce voiced sounds. The opening between the vocal folds (the empty space between the vocal folds) is known as the *glottis*. Correspondingly, the airspace between the vocal folds and the lungs is known as the *subglottal* area.

When the pressure below the glottis, known as the *subglottal pressure* increases, it pushes open the vocal folds. When open, air rushes through the vocal folds. The return movement, again closing the vocal folds is mainly caused by the Venturi effect, which causes a drop in air pressure between the vocal folds when air is flowing through them. As the vocal folds are closing, they will eventually clash together. This sudden stop of airflow is the largest acoustic event in the vocal folds and is known as the *glottal excitation*.

In terms of airflow, the effect is that during the *closed phase* (when the vocal folds are closed), there is no airflow. At the beginning of the *open phase* (when the vocal fold are open), air starts to flow through the glottis and obviously, with the closing of the vocal folds also air flow is decreasing. However, due to the momentum of air itself, the movement of air occurs slightly after the vocal folds. In other words, there is a phase-difference between vocal folds movement and glottal airflow waveform.

The frequency of vocal folds oscillation is dependent on three main components; amount of lengthwise tension in the vocal folds, pressure differential above and below the vocal folds, as well as length and mass of the vocal folds. Pressure and tension can be intentionally changed to cause a change in frequency. The length and mass of the vocal folds are in turn correlated with overall body size of the speaker, which explains the fact that children and females have on average a higher pitch than male speakers.

Note that the frequency of the vocal folds refers to the actual physical phenomenon, whereas *pitch* refers to the perception of frequency. There are many cases where these two may differ, for example, resonances in the vocal tract can emphasise harmonics of the fundamental frequency such that the harmonics are louder than the fundamental, and such that we perceive one of the harmonics as the fundamental. The perceived pitch is then the frequency of the harmonic instead of the fundamental.

### 2.2.3 The vocal tract

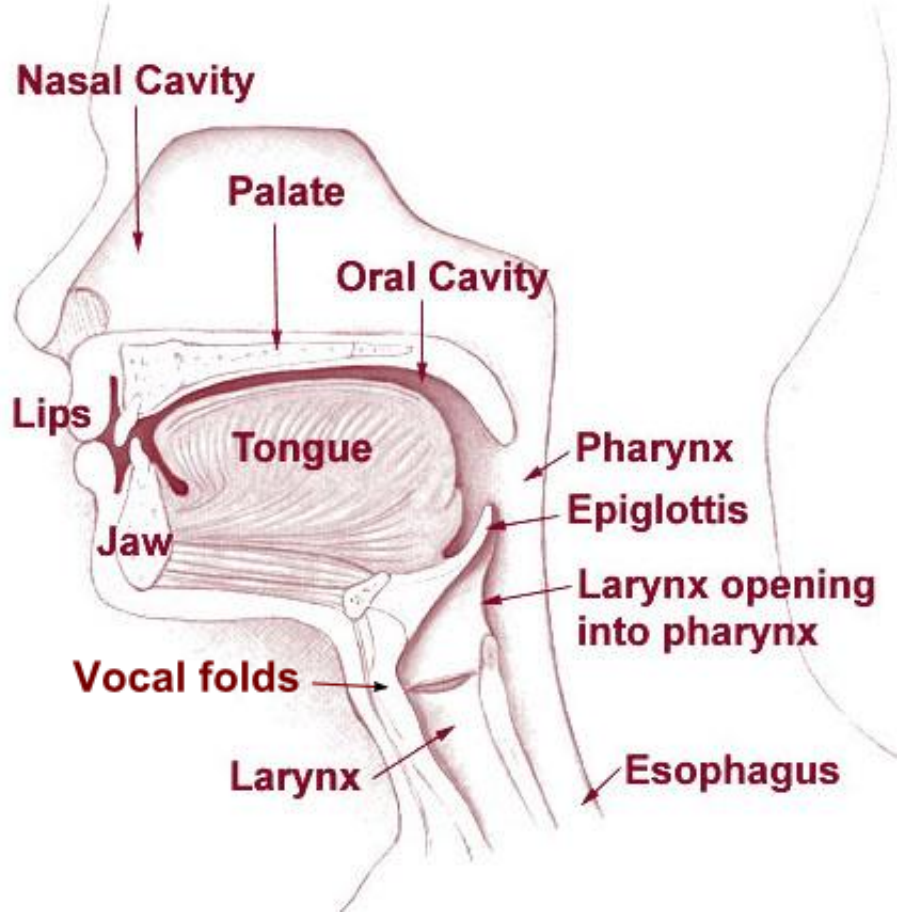
The vocal tract, including the larynx, pharynx and oral cavities, have a great effect on the timbre of the sound. Namely, the shape of the vocal tract determines the resonances and anti-resonances of the acoustic space, which boost and attenuate different frequencies of the sound. The shape is determined by a multitude of components, in particular by the position of the jaw, lips and tongue. The resonances are easily modified by the speaker and perceived by the listener, and they can thus be used in communication to convey information. Specifically, the acoustic features which differentiate *vowels* from each other are the frequencies of the resonances in the vocal tract, corresponding to specific *places* of articulation primarily in terms of tongue position. Since the air can flow relatively unobstructed, vowel sounds tend to have high energy and loudness compared to *consonants*.

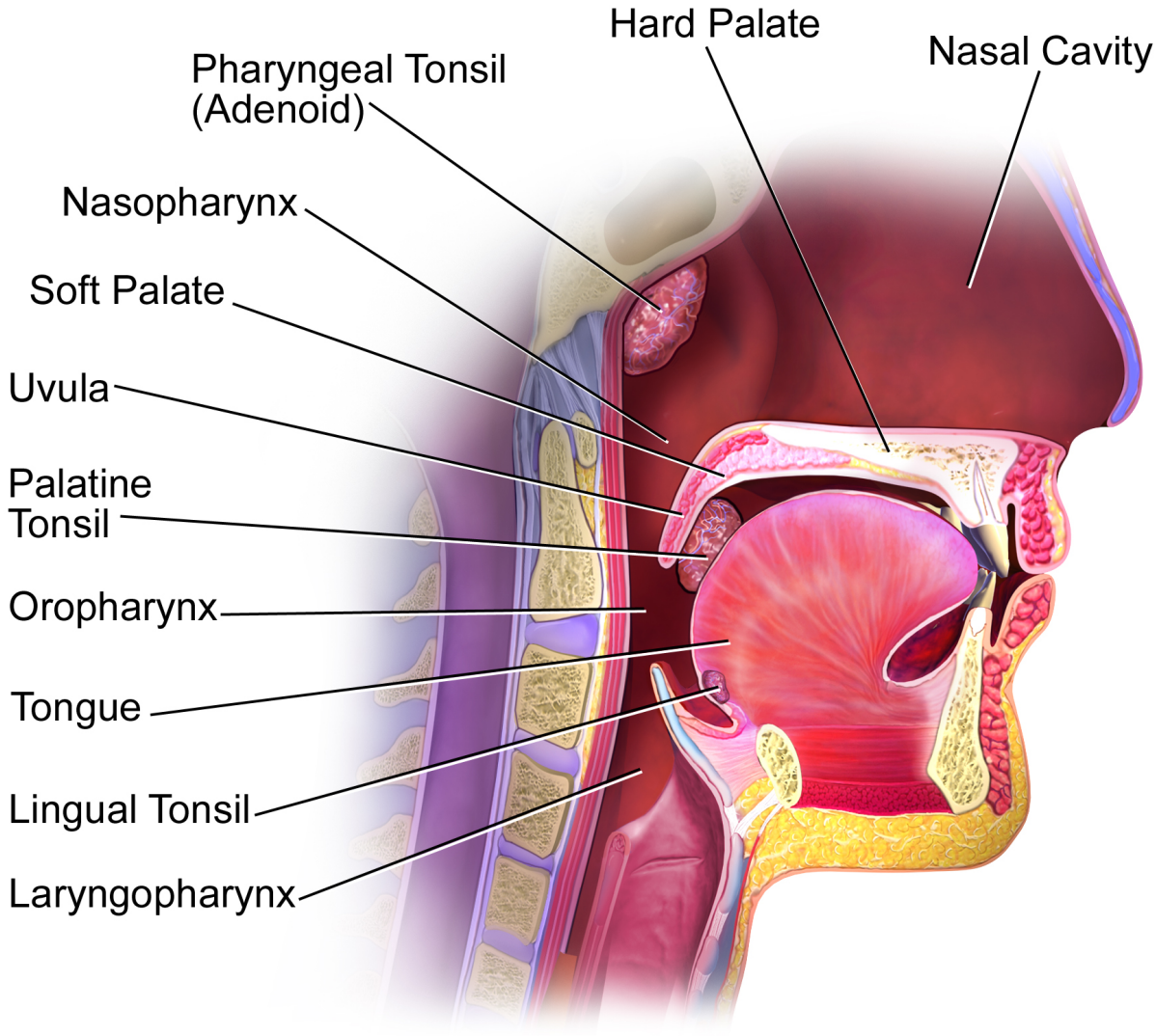
In *consonant* sounds, there is a partial or full obstruction at some part of the vocal tract. For instance, *fricative consonants* are characterized by a narrow gap between the tongue and front/top of the mouth, leading to hiss-like turbulent air flow. In plosives, the airflow in the vocal tract is fully temporarily obstructed. As an example, *bilabial plosives* are characterized by temporary closure of the lips, which leads to accumulation of air pressure in the vocal tract due to sustained lung pressure. When the lips are opened, the accumulated air is released together with a short *burst* sound (plosion) that has impulse- and noise-like characteristics. Similarly to vowels, the *place* of the obstruction in the mouth (i.e., place of articulation) will affect the acoustic characteristics of the consonant sound by modifying the acoustic characteristics of the vocal tract. In addition, *manner* of articulation is used to characterize different consonant sounds, as there are several ways to produce speech while the position of the primary obstruction can remain the same (e.g., short *taps* and *flaps*, repeated *trills,* or already mentioned narrow constrictions for *fricatives*).

In terms of vocal tract shape, a special class of consonants are the *nasals*, which are produced with velum (a soft structure at the back top of the oral cavity) open, thereby allowing air to flow to the *nasal cavity*. When the velum is open, the vocal tract can be viewed as a shared tube from the larynx to the back of the mouth, after which the tract is divided into two parallel branches consisting of the oral and nasal cavities. Coupling of the nasal cavity to the vocal tract has a pronounced impact on the resonances and anti-resonance of the tract. This is commonly perceived as *nasalization* of speech sounds by listeners.

---

Side-view of the speech production organs.



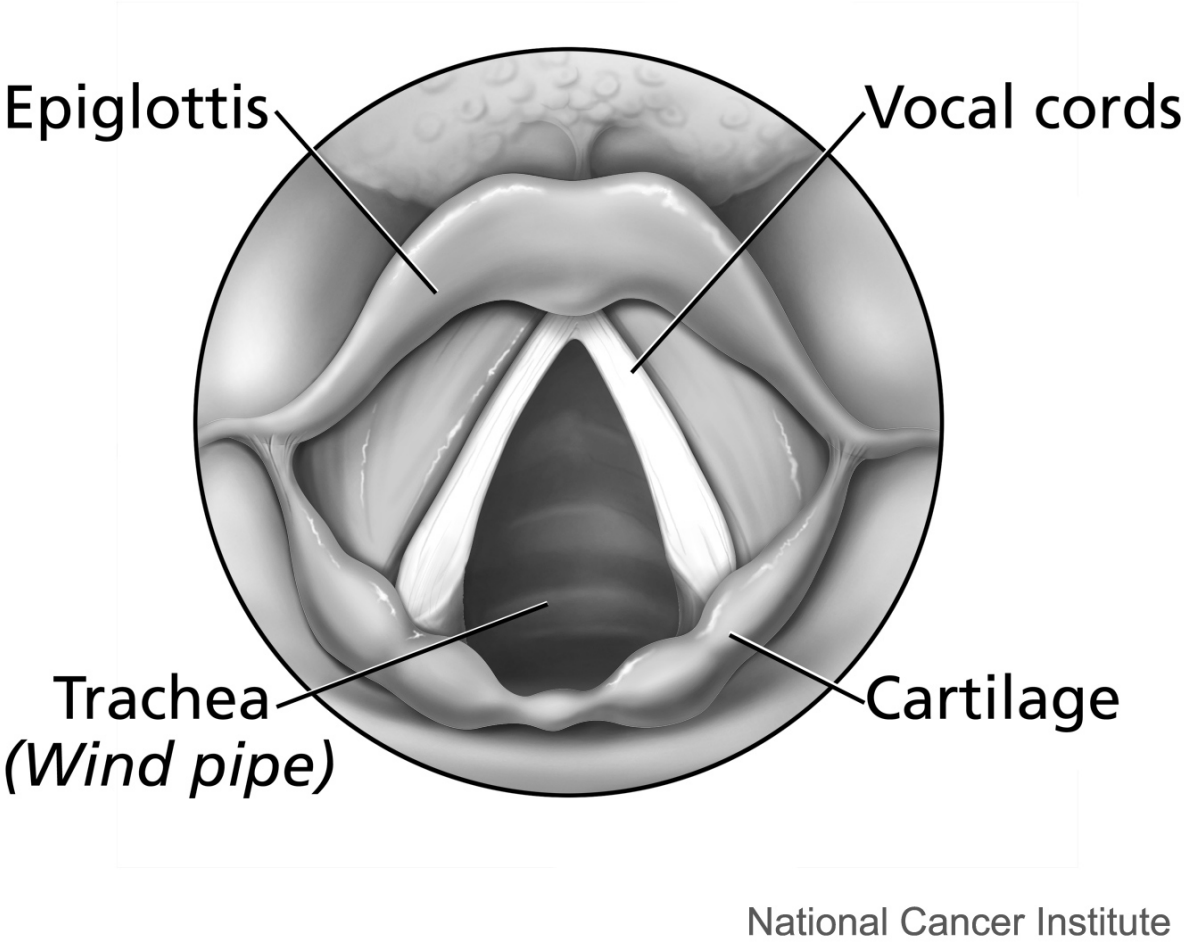



Tonsils and Throat

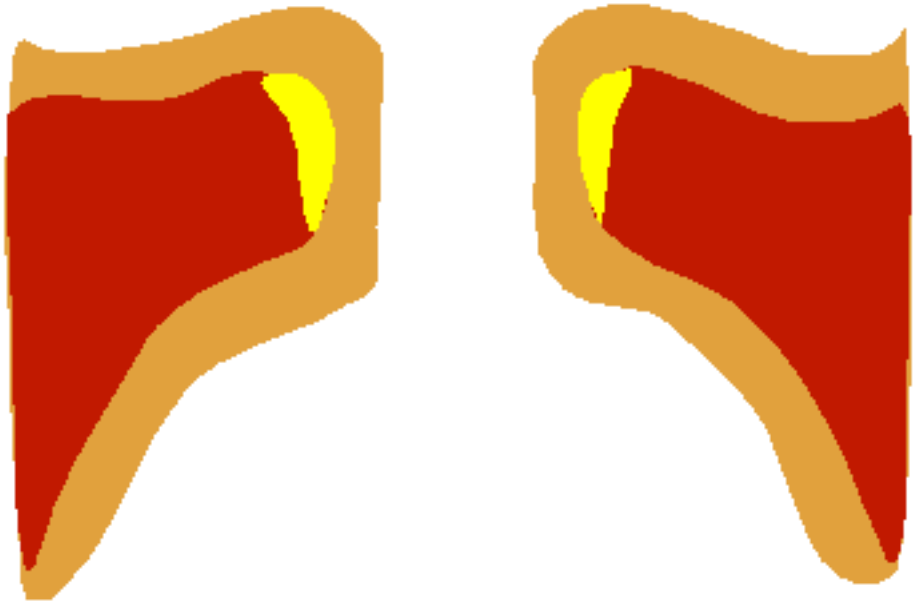
By BruceBlaus. When using this image in external sources it can be cited as:Blausen.com staff (2014). "Medical gallery of Blausen Medical 2014". WikiJournal of Medicine 1 (2). DOI:10.15347/wjm/2014.010. ISSN 2002-4436. - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=29294598

---

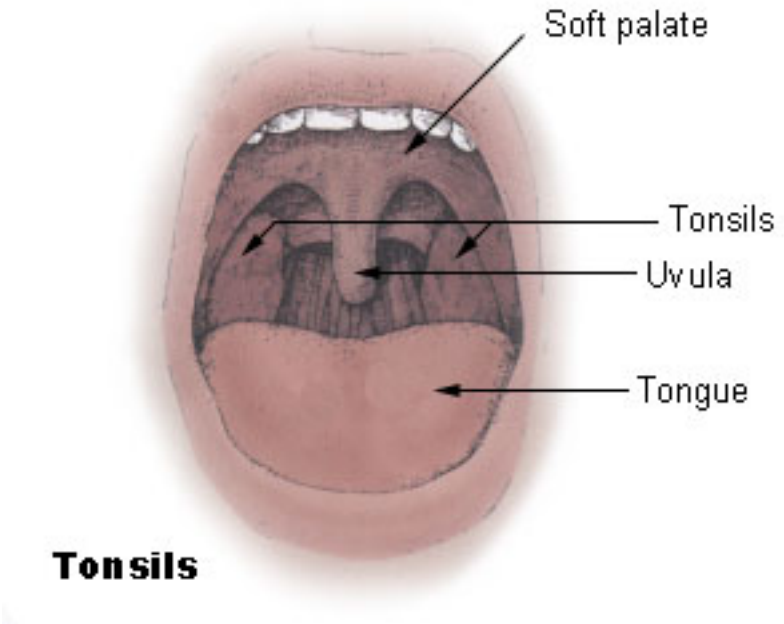
Vocal folds as seen from above.




The motion of vocal folds seen from the front (or back).



Organs in the mouth.



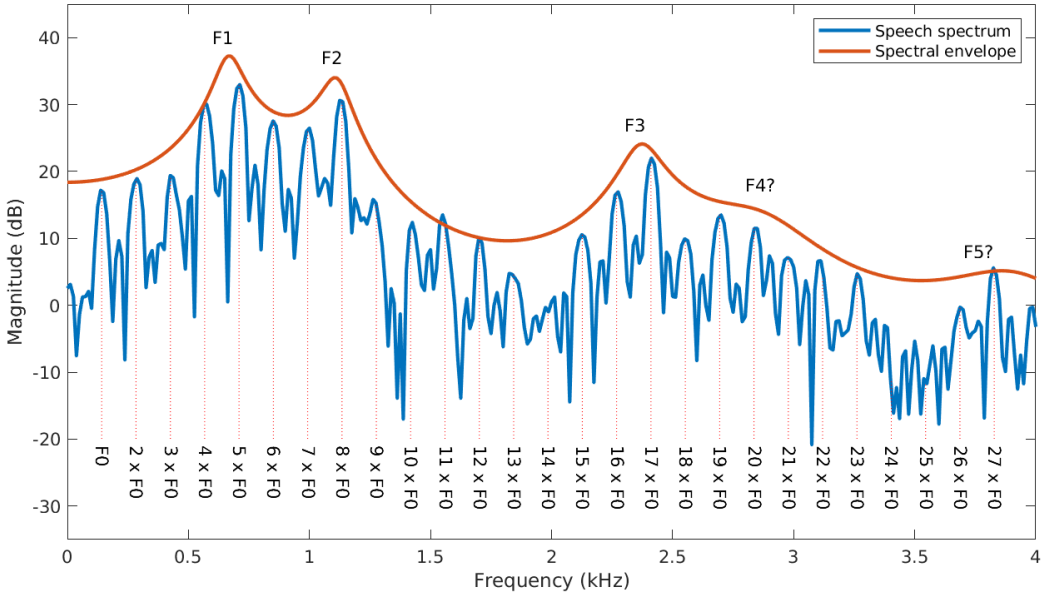

The four images above are from Wikipedia.

## 2.3 Acoustic properties of speech signals

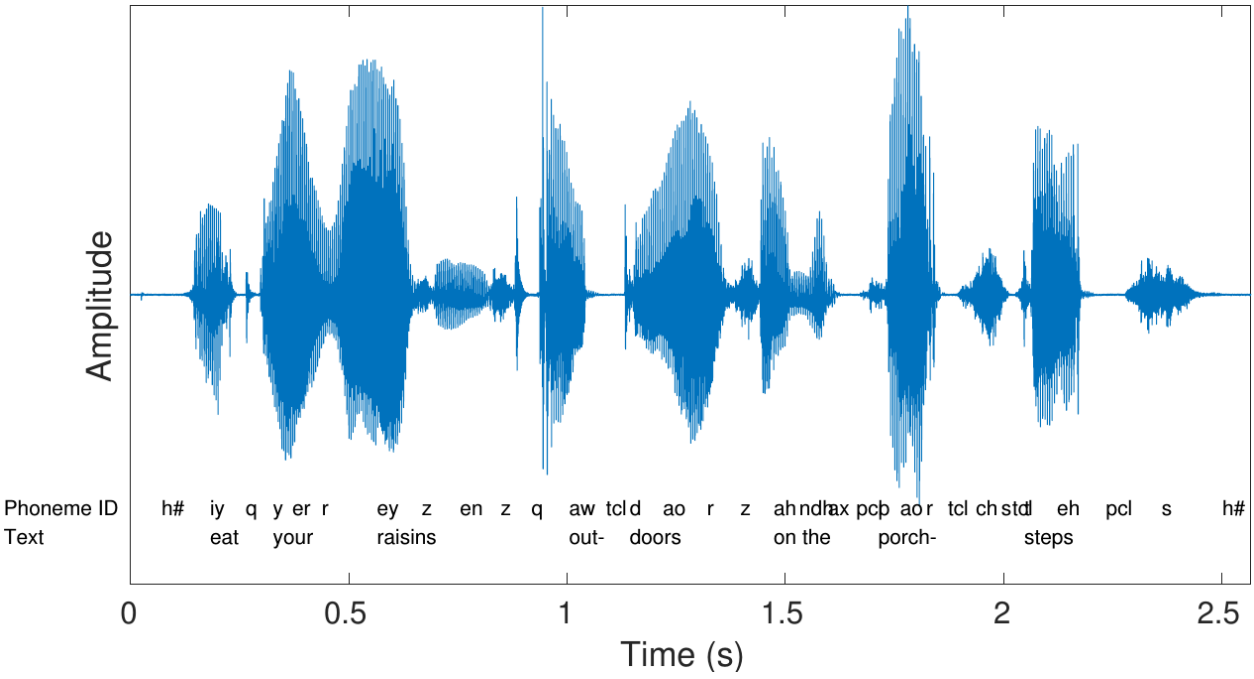
The most important acoustic features of a speech signal are (roughly speaking)

- The *resonance of the vocal tract*, especially the two lowest resonances, known as the *formants* F1 and F2 (see figure below). The resonance structure can be easily examined by drawing an "*envelope*" above the spectrum, that is, to draw a smooth line which goes just above the spectrum, as seen on the figure below. We thus obtain the *spectral envelope*, which characterizes the macro-shape of the spectrum of a speech signal, and which is often used to model speech signals.
- The *fundamental frequency* of a speech signal or its absence carries a lot of information. Per definition, voiced and unvoiced phonemes, respectively, are those with or without an oscillation in the vocal folds. Due to its prominence, we categorize phonemes according to whether they are voiced or unvoiced.The airflow which passes through the oscillating vocal folds will generally have a waveform which resembles a half-wave rectified sinusoid. That is, airflow is zero when the vocal folds are closed (closed phase) and during the open time (open phase) the waveform resembles (somewhat) the shape of the upper part of a sinusoid. The spectrum of this waveform will therefore have the structure of a harmonic signal, that is, the spectrum will have peaks at the fundamental frequency and its integer multiples (see figure below).In most languages, pitch does not differentiate between phonemes. However, in languages that are known as *tonal* languages, the shape of the pitch contour over time does bear semantic meaning (see Wikipedia:Tone (linguistics) for a nice sound sample). Pitch contours are however often used to encode *emphasis* in a sentence. Roughly speaking, exerting more physical effort on a phoneme raises its pitch and intensity, and that is usually interpreted as emphasis, that is, the word (or phoneme) with emphasis is more important than other words (or phonemes) in a sentence.
- Signal *amplitude* or *intensity* over time is another important characteristic and in its most crude form can be the difference between speech and silence (see also Voice activity detection (VAD)). Furthermore, there are phonemes characterized by their temporal structure; in particular, *stop* and *plosive-consonants*, where airflow is stopped and subsequently released (e.g. /p/, /t/ and /k/). While the stop-part is not prominently audible, it is the contrast of a silence before a burst of energy which characterizes these consonants.

The spectrum of a speech segment annotated with its formants $F_k$ (for $k \geq 1$) as well as the fundamental frequency $F_0$ and its integer multiples $kF_0$. Note that it is not always clear where the formants are; here formants F4 and F5 are not prominent and therefore difficult to locate.




The waveform of a sentence of speech, illustrating variations in amplitude and intensity.




## 2.4 Physiological modelling

### 2.4.1 Vocal tract

Vowels are central to spoken communication, and vowels are determined by the shape of the vocal tract. Modelling the vocal tract is therefore of particular interest.

#### Simple models

The vocal tract is essentially a tube of varying length. It has a 90-degree bend, where the throat turns into the mouth, but the acoustic effect of that bend is minor and can be ignored in simple models. The tube has two pathways, through the oral and nasal cavities. The acoustic effect of the oral cavity dominates the output signal such that, roughly speaking, the oral cavity generates resonances to the output sound, while the nasal cavities contributes mainly anti-resonances (dips or valleys) to the spectral envelope. Presence of energy is perceptually more important than absence of energy and anti-resonances can therefore be ignored in simple models.

A very simple model is thus a straight cylindrical tube sub-divided into constant radius segments of equal length (see illustration below). If we further assume that the tube-segments are lossless, then this tube is analytically equivalent with a linear predictor. This is a fantastic simplification in the sense that from a physiologically motivated model we obtain a analytically reasonable model whose parameters we can readily estimate from observed signals. In fact, the temporal correlation of speech signals can be very efficiently modelled with linear predictors. It offers a very attractive connection between physiological and signal modelling. Unfortunately, it is not entirely accurate.

Though speech signals are very efficiently modelled by linear predictors, and linear predictors are analytically equivalent with tube-models, *linear predictors estimated from sound signals need not correspond to the tube which generated the sound*. The mismatch in the shape of estimated and real tubes is due to two primary reasons;

1. Estimation of linear predictive coefficients assumes that the excitation, viz. the glottal excitation, is uncorrelated (white noise). This is certainly an incorrect assumption. Though the periodic structure of the glottal excitation does not much bias linear predictors, glottal excitations are also dominated by low-frequency components which will bias the linear predictor. The linear predictor cannot make a distinction between features of the glottal excitation and

contributions of the vocal tract, but model both indiscriminately. We also do not know the precise contribution of the glottal excitation such that it is hard to compensate for it.

2. The analytical relationship between coefficients of the linear predictor and the radii of the tube-model segments is highly non-linear and sensitive to estimation errors. Small errors in predictor parameters can have large consequences in the shape of the tube model.

Still, since linear predictors are efficient for modelling speech, they are useful in speech modelling even if the connection to tube-modelling is sensitive to errors. Linear prediction is particularly attractive because it gives computationally efficient algorithms.

### Advanced models

When more accurate modelling of the vocal tract is required, we have to re-evaluate our assumptions. With digital waveguides we can readily formulate models which incorporate a second pathway corresponding to the nasal tract. A starting point for such models is linear prediction, written as a delay-line with reflections corresponding to the interfaces between tube-segments. The nasal tract can then be introduced by adding a second delay line. Such models are computationally efficient in synthesis of sounds, but estimating their parameters from real sounds can be difficult.

Stepping up the accuracy, we then already go into full-blown physical modelling such as finite-element methods (FEM). Here, for example, the air-volume of the vocal tract can be split into small interacting elements governed by fluid dynamics. The more dense the mesh of the elements is, the more accurately the model corresponds to physical reality. Measuring and modelling the vocal tract with this method is involved and an art form of its own.

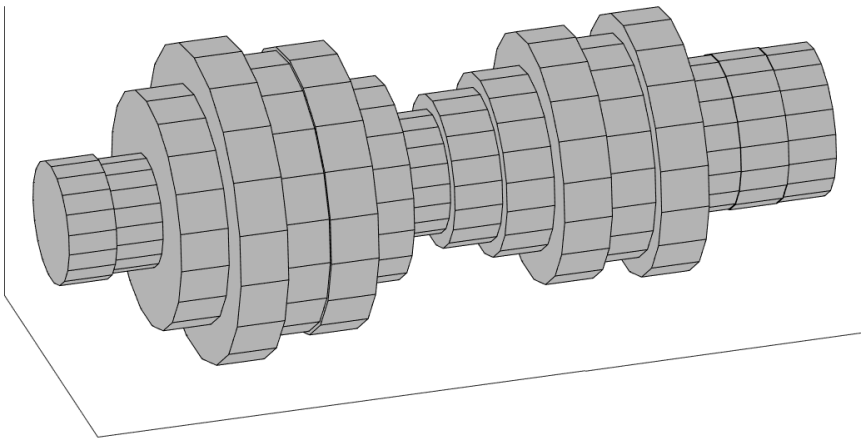


Illustration of a vocal-tract tube-model consisting of piece-wise constant-radius tube-segments.

### 2.4.2 Glottal activity

As characterization of the glottal flow, we define events of a single glottal period as follows (illustrated in the figure below):

- *Opening and closing time* (or instant), are the points in time where respectively, glottal folds open and close, and where glottal flow starts and ends.
- *Open and closed phase*, are the periods during which the glottis is open and closed, respectively.
- The length of time when glottis is open and closed are, respectively, known as *open time (OT)* and *closed time (CT)*. Consequently, the period length is $T = OT + CT$.
- *Opening and closing phases* are the portions of the open phase, when the glottis is opening and closing, respectively.
- The steepness of the closing phase is related to the "agressiveness" of the pulse, that is, it relates to the tension of glottal folds and is characterized by the (negative) *peak of the glottal flow derivative*.
- All parameters describing a length in time are often further normalized by the period length $t$.

Like modelling of the vocal tract, also in modelling glottal activity, there is a range of models of different complexity:

- *Maximum-phase linear prediction*; The most significant event in a single glottal flow pulse is its closing instant; the preceding waveform is smooth but the closing event is abrupt. The waveform can thus be interpreted as the impulse response of an IIR filter but turned backwards, which also known as the impulse response of a maximum-phase linear predictor (the figure on the right was generated with this method). The beauty of this method is that it is similar to vocal tract modelling with linear prediction, such that we are already familiar with the method and computational complexity is simple. Observe, however, that maximum-phase filters are by definition unstable (not realizable), but we have to always process the signal backwards, which complicates systems design.
- The *Liljencrantz-Fant (LF) -model* is a classical model of the glottal flow, the original form of which is a function of four parameters (defined in article). It is very useful and influential because it parametrizes the flow with a low number of easily understandable parameters. The compromise is that the parameters are not easily estimated from real signals and that it is based on anecdotal evidence of glottal flow shapes and if it were presented today, to be widely accepted, we would require more evidence to support it.
- *Mass-spring systems*; the opposing glottal folds can be modelled as simple point-masses connected with damped springs to fixed points. When subjected to the Venturi-forces generated by the airflow, these masses can be brought to oscillate like the vocal folds. Such models are attractive because, again, their parameters have physical interpretations, but since their parameters are difficult to estimate from real-world data and they oscillate only a limited range of the parameters, their usefulness in practical applications is limited.
- *Finite-element methods (FEM)* are again the ultimate method for accurate analysis, suitable for example in medical analysis, yet the computational complexity is prohibitively large for consumer applications.

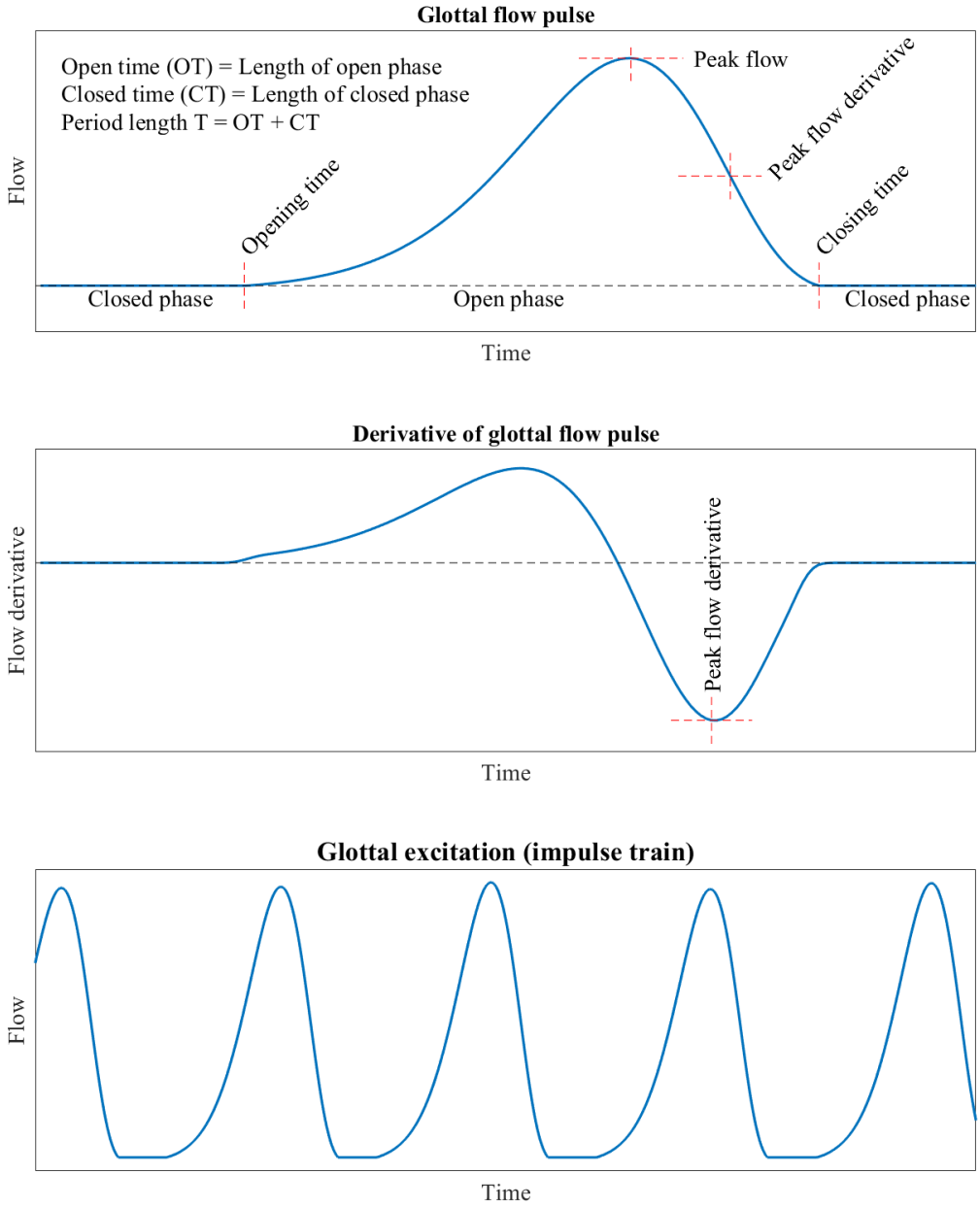



Illustration of a glottal flow pulse, its derivative and a sequence of glottal flow pulses (corresponding sound below).

```
<IPython.lib.display.Audio object>
```

### 2.4.3 Lip radiation

Having travelled through the vocal tract, air exits primarily through the mouth and in some extent through the nose. In leaving this tube, it enters the free field where airflow in has little effect. Recall that sounds are, instead, variations in air pressure. At the transition from the tube to the free field, variations in air flow become variations in air pressure.

The physics of this phenomenon are governed by fluid dynamics, an advanced topic, but heuristically we can imagine that variations air pressure is related to variations in airflow. Thus if we take the derivative of the airflow, we get an approximation of its effect on air pressure $sound(t) \approx \frac{d}{dt} flow(t), where t$ is time.

Often we deal with signals sampled at time indices $n$, where the derivative can be further approximated by the first difference $sound(n) \approx g\left[flow(n) - flow(n-1)\right], where g>0$ is a scalar gain coefficient.

## 2.5 Linguistic structure of speech

### 2.5.1 Overview

Besides their acoustic characteristics, speech signals can be characterized in terms of their *linguistic structure*. Linguistic structure refers to the recurring regularities in spoken language as described by linguistic theories, such as what are the basic building blocks of speech and how they organized. Linguistic descriptions provide a means for systematic interpretation, conceptualization, and communication of speech-related phenomena.

Many linguistic properties of speech, such as syllables and words, also have their analogs in written language. However, it is important to distinguish the two from each other: While written language consists of discrete categorical elements (sequences of letters, whitespaces, and punctuation marks), speech signals are always continous and non-categorical. This is due to the motor behavior in speech production, which operates in real-time and finite speed under physical and neuro-physiological constraints. Therefore, the resulting speech signal also flows in continuous time and frequency. In addition, where speech carries extra information in terms of *how* things are said and what are the *characteristics of the speaker*, written language is impoverished of these aspects. In contrast, written language uses lexical and syntactic means and special characters (and more recently, emoticons) to differentiate more fine-grained meanings, such as conveying emotional content or differentiating questions from statements. Finally, only few languages have clear one-to-one relationship between how words are pronounced and how they are written. Therefore, a separate system is needed to describe structure of spoken language, and one should not equate units of written language to that of a spoken one by default. Naturally, the two have a systematic relationship in order to enable reading and writing. However, this relationship also varies from language to another. **

Within the broad field of linguistics, *phonetics* is its branch that focuses on understanding the physical basis of speech production, speech signals, and speech perception. In contrast, *phonology* is a branch of linguistics that studies sound systems of languages (both spoken and signed). While both attempt to describe how spoken language is organized, phonetic description attempts to be faithful to the acoustic, articulatory and auditory aspects of the speech signal. Phonetics are therefore strongly grounded to the measurable phenomena in the physical world. In contrast, phonological description consists of abstract speech units that allow more convenient study of how sounds of a language combine to create meaningful messages. For this, phonological description usually gets rid of signal variation that does not impact meaning of the speech. To give an example, different speakers of the same language may use different pronunciations of the same word, still resulting in the same phonological form. In contrast, accurate phonetic description would reflect the pronunciation-dependent differences, allowing documentation and study of such differences. Since speech processing is primarily concerned with physical (digitized) speech signals and how to deal with them, we will use phonetics as the primary language of description. However, the basic relationship between phonetic and phonological units is also briefly discussed below.

### 2.5.2 Elementary units of spoken language

In terms of basic phonetic organization, speech can be seen as a hierarchical organization of elementary units of increasing time-scale (Fig. 1). At the lowest level of hierarchy, there are *phones*, which are considered as physical realizations of more abstract *phonemes*. Sequences of phones are organized into *syllables*, and syllables make up *words* (where each word consists of one or more syllables). One or more words then make up utterances. Phones are sometimes referred to as *segmental units*, and speech phenomena, such as intonation, taking place at time-scales larger than individual phones are called as *suprasegmental phenomena*. In addition, speech is sometimes said to have so-called *double articulation* (aka. *duality of patterning*). This refers to the fact that meaningful units of speech (words, utterances) consist of non-meaningful units (phones/phonemes) that still signify distinctions in meaning. At all levels, units and their relative organization are language-dependent, such as which phones, syllables, and words are employed and how they are allowed follow each other. However, there are also certain common tendencies (aka. *linguistic universals*) that result from restrictions in the speech production and perception apparati or due to other shared characteristics of natural languages.

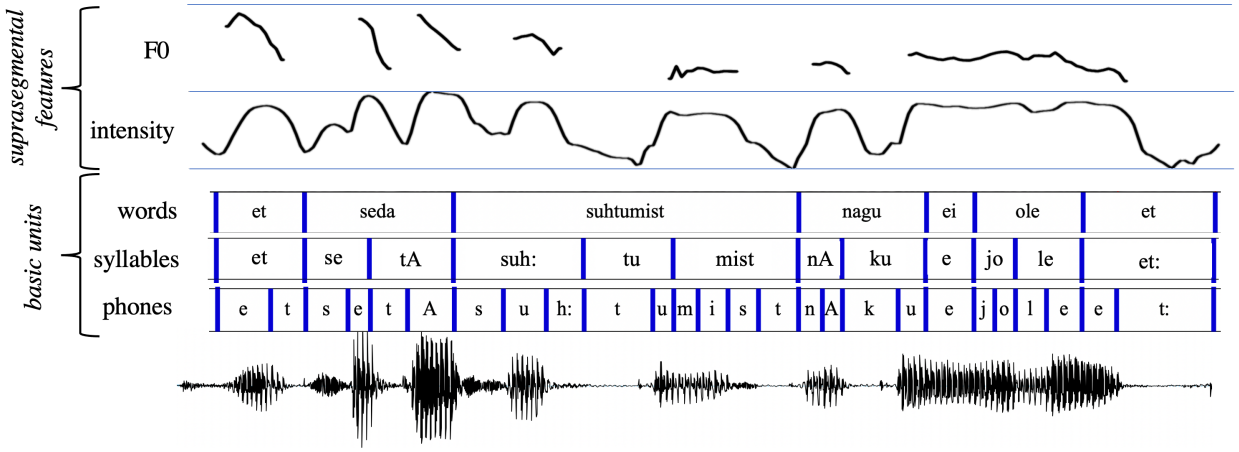



**Fig. 1: An example of the hierarchical organization of speech in terms of phones, syllables, and words for an Estonian speech sample. Two suprasegmental features, namely F0 and intensity, are also shown on top. In this example, syllables are denoted in terms of their phonetic constituents while words are represented orthographically. Example annotations taken from Phonetic Corpus of Estonian Spontaneous Speech (reproduced with permission) and represented in graphical form using Praat.**

When discussing units of speech, it is often useful to distinguish *unit types* from *unit tokens.* Type refers to a unique unit category, such all [r] sounds belong to the same phone type. Token refers to an individual realization of a type. For instance, word "roar" has two [r] tokens in it.

#### Phones

Phones are the elementary units of speech, associated with *articulatory gestures* responsible for producing them and with *acoustic cues* that make them distinct from other phones. Phonetic *transcription* is the process of marking down phones of speech with symbols (denoted with brackets [ ]). Phonetic transcription often makes use of the International Phonetic Alphabet (IPA). On a high level, phones can be divided into *vowels* (e.g., [a], [i], and [u]) and *consonants* (e.g., [p], [b], [s]). While all vowels are voiced sounds (see *speech production*), consonants can be voiced or unvoiced.

The primary determiner of vowel identity is position of the narrowest gap in the vocal tract (Fig. 2), as primarily controlled by tongue position in the mouth. Vowels can be categorized in terms of their *openness/closeness* (how wide is the narrowest part of the vocal tract between tongue and top of the mouth), and *frontness/backness* (how front/back is the tongue in the mouth). In additon, vowels with the same place of articulation can change depending on *lip rounding* (e.g., [o] versus [u]). Protrusion of lips during the rounding increases the effective length of the vocal tract, hence altering the resonance frequencies of the tract. A special vowel called "schwa" ([ə]) corresponds to an articulatory configuration, where jaw, lips, and tongue are completely relaxed, hence corresponding a central mid vowel.

Fig. 2: IPA chart for vowels (reproduced by CC BY-SA 3.0)

Consonants can be categorized in terms of their *manner and place of articulation.* Figure 3 shows IPA chart for consonants organized in terms of place of articulation (columns)and manner of articulation (rows).

Place of articulation refers to the point of tighest constriction in the vocal tract, as in vowels. However, in consonants, the gap at the place of constriction is smaller or the airflow in the tract is completely blocked for a period of time. In addition, the constriction can be created at different positions of the vocal tract using different articulators beyond the tongue, such as using lips, glottis, or uvula.

Manner of articulation refers to the manner that the constriction is created, including its temporal characteristics. For instance, *plosives* (e.g., unvoiced [k], [p] or voiced [g], [b]) consist of a complete blockage, aka. *closure,* of the vocal tract. This causes accumulation of air pressure in the tract before the blockage, which then results in as a noisy burst of airflow when the closure is released. *Trill consonants*, such as [r], consist of several rapid and subsequent closures of the tract caused by tongue vibrating against the oral cavity structures at the given place of articulation (e.g., tongue tip vibrating against alveolar ridge in [r]). *Taps and flaps* are similar to trills, but only consist of one quick closure and its release. *Fricatives* are turbulent sounds where the constriction is so narrow that the air flowing though it becomes turbulent. This turbulence causes the "hissing" sound characteristic of fricatives, such as in [s], [f], and [h] (e.g., as in "*she*" or "*foam*"). *Nasals *are sounds where the oral cavity has a complete closure, but the air can enter the nasal cavity through the velar port and exit through nostrils, adding another branch to the vocal tract with its own resonances and antiresonances (see also *speech production*). *Approximants* are sounds where articulators approach each other without becoming narrow enough to introduce turbulence to the sound. Among the approximants, *g*lides are sounds similar to vowels, but act as syllable boundaries instead of syllabic nuclei (see below) typically with articulatory movement taking place throughout the sound (e.g., [y] as in "yes" or [w] in "water"). Lateral approximants consist of sounds where some part of the tongue touches the roof of the mouth but the air can flow freely from both sides of the tongue (e.g., [l] as in "love"). There are also other categories of consonants and ways to categorize them, and an interested reader is suggested to consult some phonetics textbook for more information.

CONSONANTS (PULMONIC) — ©①◎ 2020 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 | | | |

Symbols to the right in a cell are voiced, to the left are voiceless. Shaded areas denote articulations judged impossible.

**Fig.3: IPA chart for pulmonic consonants (reproduced by CC BY-SA 3.0).**

The majority of consonant sounds, including those reviewed above and listed in Fig. 3, are called *pulmonic consonants.* This means that the air flow (energy) for sound production originates from lung pressure, whether voiced or not. Besides the pulmonic sounds, some languages make use of *non-pulmonic consonants*. These can include *implosives* (plosives that are formed during inhalation of air), *clicks* (produced by subpressurized pockets of air between two concurrent closures that are then suddenly released), and *ejectives* (where the excitation energy for the sound is formed at the glottis by lowering the phrarynx, closing the glottis, and then raising the pharynx, resulting in increased air pressure in the tract while an obstruction is maintained at the place of articulation).

*Coarticulation* is an important speech phenomenon where realization of speech sounds is affected by the neighboring speech sounds. This is since articulatory gestures for speech production are almost always anticipating the production of the next sound (aka. *anticipatory coarticulation)* or still recovering their positions from the previous sound (aka. *per-*

*severative* coarticulation). Since all articulators can only move with a finite speed from one articulatory configuration to another, coarticulation is present in virtually all observable speech beyond isolated vowels. Given that phones are the smallest sound units of a language, coarticulation can have a large effect on the acoustic form that phones take in actual continuous speech, introducing additional non-phonemic variability (see below) in the acoustical form of phones.

### Phones vs. phonemes

As mentioned above, phones are sounds of a language that have an articulatory, and thereby also acoustic, basis. Another commonly encountered basic unit of speech is a *phoneme* (denoted with slashes / /). Phonemes are defined in terms of their meaning contrasting function: two different phones of a language are also different phonemes, if they can change the meaning of a word. For example, consider words "*cat*" ([k] [ae] [t]) and "*bat*" ([b] [ae] [t]). In case of English, the initial [k] and [b] phones are also distinct phonemes /k/ and /b/, as they change the meaning of the otherwise identical word. Also note that [k] [ae] [t] and [b] [ae] [t] are so-called *minimal pairs*, as they only differ by one phoneme. A good rule of thumb is that *phones are defined in terms of their articulatory or acoustic properties,* whereas *each* *phonemic category consists of all possible sounds that can be substituted for each other without affecting the meaning of any word in the given language. *

*Allophones* are the alternative phones that all stand for the same phoneme in the given language. For instance, phones [r] and [l] can be considered as allophones of the same Japanese phoneme, as they can be used interchangeable in Japanese without affecting the communicated meanings. In English, [r] and [l] are not allophones, as they distinguish meanings (e.g., as in "*lock*" and "*rock*"). In fact, listeners tend to become worse in discriminating those non-native phonemic contrasts from each other that do not mark a phonemic contrast in their native language. This change in discrimination comes with language experience, and young infants start with the capability to discriminate both native and non-native contrasts (e.g., Werker & Tees, 1984).

### Syllables

The second unit in the "size hierarchy" of spoken language is a syllable. Syllables are sequences of sounds (one or more), consisting of a syllable *nucleus* (typically a vowel) and optional initial and final sound sequences, also known as *onset* and *coda*, respectively. Onsets and coda tend to have consonant sounds. Sometimes onset is separated from the *rime*, where the rime then consists of the nucleus and coda. Syllables can be considered as rhythmic units, as the alternation between consonants and vocalic syllabic nuclei give rise to the typical rhythmic patterns of different languages, as vowels generally have higher energy (are louder) than consonants. Concept of *sonority sequencing principle* (SSP) refers to this sequential alternation between less loud consonants and louder syllable nuclei. More specifically, in SSP, the first phone of syllable onset is supposed to be the least sonorous of the sounds preceding the syllable nucleus. Then the sonority increases towards the nucleus if more than one consonant exists in the onset. In the same manner, sonority of the consonants in the syllable coda (offset) decreases towards the end of the syllable. *Sonority hierarchy* determines the relative sonority ("loudness") of different phones. As wikipedia states, "*typically they *[relative sonorities] *are vowel > glide > liquid > nasal > obstruent (or > fricative > plosive > click)*".

Due to coarticulation that can have large impact on individual phone segments and due to rhythmic transparency of syllables, some authors consider syllables as more robust perceptual units of language than individual phones (e.g., Nusbaum & DeGroot, 1991), including also young children (see Hallé & Cristia, 2012, for an overview). In addition, children and illiterate listeners have better introspective access to syllabic structure of speech in contrast to underlying phonetic or phonemic constituents (Liberman et al., 1974; Morais et al., 1989). However, coarticulation has also effects across subsequent syllables, and human speech perception also makes use of cues beyond the time-scale of individual syllables. However, syllabic organization of speech is still central to understanding how speech is organized. Studies on conversational speech show that onset and nucleus of a syllable are much more central to succesful comprehension of speech, and hence also produced more accurately. In contrast, coda often undergo various types of *syllabic reduction* where one or more phones of the coda are not pronounced at all (e.g., Greenberg et al., 2003).

Different languages employ different syllabic structures. Syllables (of a language) are sometimes denoted in terms of their constituent vowels (V) and consonants (C), where CV-syllables (one consonant onset + a vowel nucleus) are universally the most common. In addition, each language has its own inventory of syllables that can be of form CVC, CVV, CVCC,

CVVCC etc.. For instance, English has on average quite long syllables, resulting on many *monosyllabic words (bat: [b]* [ae] [t], *food:* [f] [u:] [d], *laugh: *[l] [ae] [f]). In Finnish, CV syllables are frequent and hence many frequently spoken words emerge from combinations of several syllables (e.g., *kissa :* [k i s] . [s a] for *cat, *or *nauraa:* [n a u].[r a:] for *to laugh*; note that . marks for syllable boundary and : for a long vowel/consonant quantity).

### Words

Words are the minimum meaning-bearing units of spoken (and written) language, i.e., a word in isolation has some meaning attached to it. In contrast, isolated phones and syllables do not carry a meaning (unless the word is a monosyllabic one). Every word has at least one syllable, and, in principle, syllables do not cross word bondaries but align with them. In several languages, spoken words align relatively well with written words. However, words in speech are not separated by clear articulatory or acoustic markers, such as pauses, whereas written text transparently delimits individual words by intervening whitespaces.

### Utterances

Utterance is the smallest unpaused act of speech produced by one speaker, as delineated by clear pauses (or changes of speaker). In contrast to written language, where *a sentence* is one grammatical expression with a communicated meaning, utterances in spoken language can vary from individual words to much longer streams of words and grammatical constructs. In other words, speech does not consist of clearly delinated sentences (or clauses), but of speaking acts of varying durations. Speakers may use *fillers* (aka. *hesitations*, *filled pauses*) such as "*uhm*" or "*ah*" or prolonged vowels to signal that they aim to continue their utterance, but need some to reorganize their thinking in order to mentally construct the subsequent speech.

### Morphological units

Besides the above-listed units of speech, there are a number other units and structural concepts of language that linguistic theories make use of. They are perhaps less frequently utilized in speech processing, but become relevant when speech is being mapped to written language or vice versa, or when speech processing tools are used for linguistic research. One such an area of linguistics is *morphology*, which studies word forms, formation, and relationships between words in the same language. In morphology, *morpheme* is the smallest grammatical unit of speech, which may, but does not have to be, a word. Morphemes can be divided into *free morphemes*, which can act as words in isolation (e.g., "cat"), and *bound morphemes*, which occur as prefixes or suffixes of words (e.g., -s in "cats"). In addition, bound morphemes can be classified into *inflectional morphemes* and *derivational morphemes.* Inflectional morphemes change the grammatical meaning of the sentence, but do not alter the basic meaning of the word that is being inflected. Derivational morphemes alter the original word by creating a new word with a separate meaning. *Allomorphs* are different pronunciation forms of the same underlying morpheme.

Besides morphemes, morphology deals with *lexemes.* Lexeme is a unit of meaning from which different inflections (*word-forms*) can be derived. Hence, all words belong to some lexeme, but one lexeme can have many word-forms (e.g., *"do", "did", "does"*). Set of lexemes in a language is called *a lexicon*.

### 2.5.3 Prosody, aka. suprasegmental properties of speech

Prosody aka. suprasegmental phenomena in speech refers to those patterns in speech that take place at time-scales larger than individual phones (segments). Many of the suprasegmental phenomena, such as intonation, stress, and rhythm, play a linguistic function, hence providing an additional means to alter the meaning and implications of spoken message without changing the lexical and grammatical structure of the sentence. Others are related to other information encoded in speech, such as cues for speaker's emotional state, attitude, or social background. Here we only focus on those aspects of suprasegmentals that play a linguistic role.

#### Intonation

*Intonation* corresponds to variations in fundamental frequency (F0) of speech as a function of time in speech, as perceived in terms of pitch. By altering the relative pitch as a function of underlying linguistic constituents or as a function of relative position in the utterance, the speaker can signal information such as *emphasis*, *surprisal*, *question* (vs. statement), *or irony*. For instance, English speakers often use falling intonation contour (across the utterance) for statements and rising intonation pattern for questions. Marking of focus and stress can be done with sudden increase in pitch during the marked syllable or word. In addition, *tone languages* (e.g., Mandarin Chinese) use pitch to phonemically to distinguish meanings of phonetically otherwise equivalent words. In general, the presence and function of intonational patterns depends on the language in question. Intonation can also contain other structural cues to speech, such as *boundary tones* to signify end of an sentence or utterance (Pierrehumbert, 1980), thereby also signifying end of an intonational phrase.

#### Stress

*Stress* (aka. *accent*) corresponds to relative emphasis given to one syllable or word over the others in the given phrasal context. Stress can be realized by many means, including alternations in energy (loudness), pitch, and segment (typically vowel) duration with respect to the surrounding speech. Besides signifying emphasis, some languages also employ fixed or semi-regular stress patterns on words.

*Word stress* refers to the emphasis of a syllable or particular set of syllables within a word, and where multiple stressed syllables can be divided into those with primary and secondary stress. For instance, Finnish as nearly always primary stress on the first syllable of the word, and secondary stress falls on the following odd-numbered syllables. In English, words tend to have stress on on the initial syllables, but there are multiple exceptions to this (e.g., word "*guitar*", where the stress is on "-tar"). Some languages are sometimes considered to be completely void of stress.

*Sentence stress* (aka. *prosodic stress*) refers to stress on certain words or parts of words within an utterance, either signifying emphasis or contrast (e.g., "No, **I** went home" vs. "No, I went **home**"; stressed word highlighted).

#### Rhythm

Speech *rhythm* is a result of complex interplay of several factors in speech production, where the flow of chosen words is also affected by stress patterns, segmental and pause durations, and general syllable structure of the given language. At a general level, rhythm refers to some kind of sense of recurrence in the speech, such as alternation between stressed and unstressed syllables.

A more narrow definition of rhythm relates to the idea of *isochrony*, according to which languages can be categorized into three rhythmic categories in terms of what is the determining recurrent structure in the signal (Pike, 1945; see also, e.g., Nespor et al., 2011). The first category consists of syllable-timed languages, where duration of each syllable is equal. The second corresponds to mora-timed languages, where duration of each mora is equal (see mora on Wikipedia). The last category consists of stress-timed languages, where the interval between stressed syllablees is equal. In practice, evidence for such a precise rhythmic regularities in actual speech data is limited, although many would likely agree that some languages tend to share some rhythmic characteristics that make them distinct from the rhythm of others.

### 2.5.4 Phonetic transcription and speech annotation

*Phonetic transcription* is the process of marking down the phonetic structure of speech, typically aligned with the timeline of the speech waveform. Transcription is usually conducted according to IPA standards. The process can be carried out using either *narrow transcription* or *broad transcription*. In broad transcription, the marking typically consists of the most distinct phonetic elements in the speech data, such as individual phones and their basic allophonic variations. In *phonemic transcription*, which is the broadest level possible, only the phonemes corresponding to the speech are transcribed. In contrast, narrow transcription contains more phonetic detail regarding realization of the speech sounds. These details can include information on stress and intonation, and the transcription may also consists of diacritics, which provide more details on the articulatory/acoustic realization of the phones compared to their standard definitions in the IPA system (see the full IPA chart).

*Annotation of speech* is a more general term than phonetic transcription. Annotations may consist of different layers of information in addition to (or instead of) the phonetic information. For instance, speech annotations may contain syllable and word boundaries and identities of the corresponding units. These identities can be marked down phonetically or ortographically, and are recorded either as actually pronunced or as *canonical* (i.e., as they would be listed in a dictionary). When contents of speech are marked as regular text, it is called *orthographic transcription*. Other commonly utilized annotation layers include utterance boundaries, speaker identities/turns, grammatical information such as parts of speech, morphological information, or potential presence of special non-speech events such as different categories of background noise. Since production of high-quality annotations usually requires manual human work and is very slow, different speech datasets tend to contain annotations that were of primary research interest to the person or team collecting and preparing the dataset.

Automatic speech processing tools can be of use in speeding up the annotation by, e.g., providing an initial version of annotations for humans to verify and correct. A particularly useful instance of automatic tools is the so-called *forced-alignment* with an automatic speech recognizer, which is suitable for cases where the acoustic speech signal and the corresponding spoken text are known. Assuming that the text is faithful to the underlying speech contents, a good recognizer can produce much more accurate phonemic transcription and/or timestamps for word boundaries than what would be achievable by regular automatic speech recognition without the reference text.

### 2.5.5 References


- Greenberg, S., Carvey, H., Hitchcock, L., and Chang, S. (2003). Temporal properties of spontaneous speech – a syllable centric perspective. *Speech Communication*, *31*, 465–485, https://doi.org/10.1016/j.wocn.2003.09.005
- Hallé, P., & Christia, A. (2012). Global and detailed speech representations in early language acquisition. In S. Fuchs, M. Weirich, D. Pape, & P. Perrier (Eds.). *Speech planning and dynamics*. Frankfurt am Main: Peter Lang.
- Liberman, I. Y., Shankweiler, D., Fischer, W. F., & Carter, B. (1974). Explicit syllable and phoneme segmentation in the young child. Journal of Experimental Child Psychology, 18, 201–212.
- Morais, J., Content, A., Cary, L., Mehler, J., & Segui, J. (1989). Syllabic segmentation and literacy. Language and Cognitive Processes, 4(1), 56–67.
- Nespor, M., Shukla, M., & Mehler, J. (2011). Stress-timed vs. syllable-timed languages. In van Oostendorp et al. (Eds.), The Blackwell Companion to Phonology (pp. 1147-1159). Malden, MA: Blackwell.
- Nusbaum, H. C., & DeGroot, J. (1991). The role of syllables in speech perception. In M. S. Ziolkowski, M. Noske, & K. Deaton (Eds.). *Papers from the parasession on the syllable in phonetics and phonology*. Chicago: Chicago Linguistic Society.
- Pierrehumbert, Janet B. (1980) "The Phonology and Phonetics of English Intonation" Ph.D. Thesis, Massachusetts Institute of Technology.
- Pike, K. (1945). The intonation of American English, vol 1. Ann Arbor: University of Michigan Press. pp. 34–35.

- Werker, J. F., & Tees, R. C. (1984). Cross-language speech perception: Evidence for perceptual reorganization during the first year of life. Infant Behavior & Development, 7(1), 49–63. https://doi.org/10.1016/S0163-6383(84)80022-3


Also, substantial reuse of materials from related Wikipedia articles.

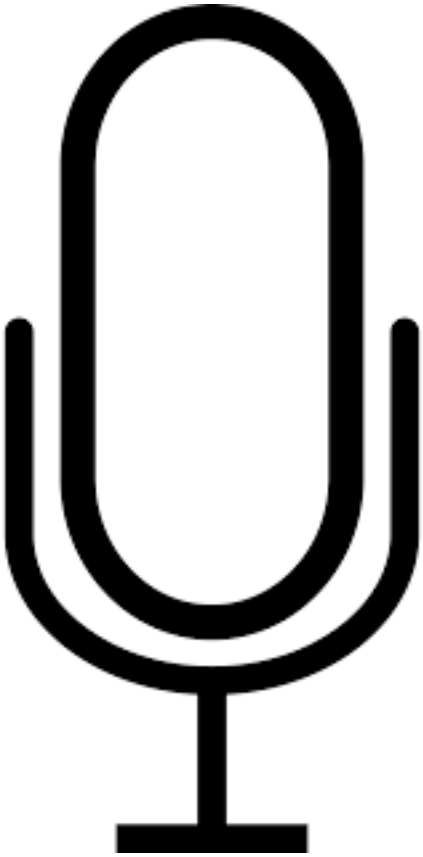
## 2.6 Applications and systems structures

### 2.6.1 Applications

Speech processing is used in, for example;

- Telecommunication
  - Phones and mobile phones
  - Teleconferencing systems
  - Voice-over-IP, like Skype, Google Hangouts, Facetime, Zoom
  - Podcasts, digital radio and TV
  - Virtual reality and gaming applications
- Speech operated virtual assistants, like Siri, Alexa, Google Assistant, Mycroft, Cortana etc.
  - Also speech interfaces of robots
- Navigators with speech feedback, like TomTom and Garmin
- Automated telephone services (like the helpdesk of an airline)
- Automated transcription
  - Youtube subtitles
  - Recorded notes from doctors
- Microphones and microphone systems
  - Headsets, with or without noise attenuation, with or without active noise cancelling
  - Stage microphones (related to audio processing)
- Studio recording systems (related to audio processing)
  - Autotune
  - Noise attenuation/reduction
- Dubbing movies e.g. for translation of audio
  - Also on-line translation
- Speech synthesis services
  - Automated e-book readers
  - User-interfaces for the blind and the handicapped
- Less common applications
  - Access control and fraud detection with speaker identification and verification
  - Anonymization and obfuscation (e.g. witness protection) of speech signals

- Speech synthesizers for disabled people (e.g. Stephen Hawking)
- Medical analysis of speech signals (e.g. Alzheimer detection)



Such applications can be categorized according to functionality, roughly as:

- Transmission and storage of speech
  - Enable communication with a far-away person
- Speech operated user-interfaces
  - Enable spoken interaction with a machine
- Information extraction from a speech signal
  - Like automated transcription to generate subtitles for movies
- Speech synthesis, i.e. automated generation of speech
- "Improvement" of a speech signal, such as
  - Such as noise reduction, translation

Note that these categories are in many senses overlapping; for example, noise reduction can be a part of any speech processing system and information extraction is practically a mandatory part of speech operated user-interfaces.

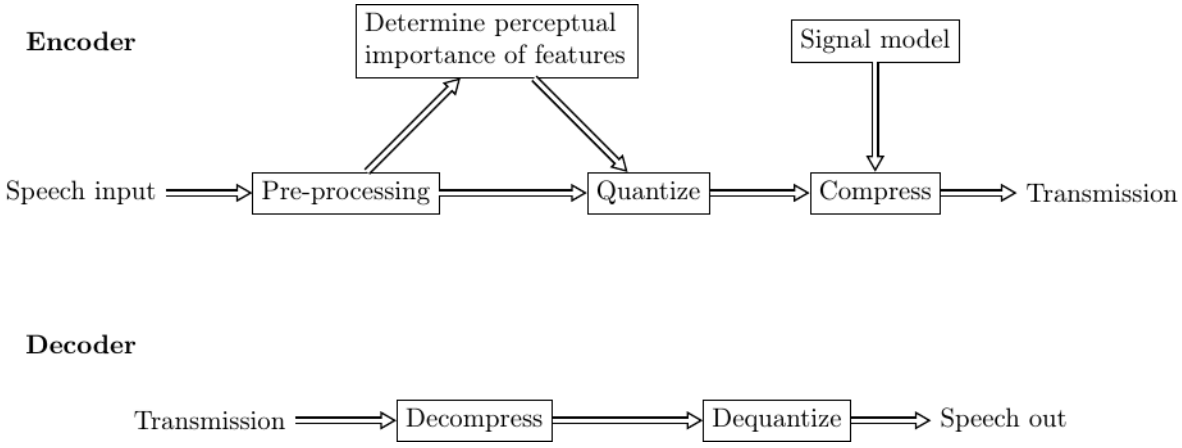
### 2.6.2 Systems structures

#### Transmission and storage

The objective of speech transmission systems is to compress the signal to as few bits as possible, while keeping the sound quality at the output as good as possible. This requires that the degradations that we introduce are chosen such that their perceptual influence is as small as possible. In other words, we would not like the listener to notice (or to notice as little as possible) that the signal has been degraded. In the illustration on the right, at the encoder on the sender side, we therefore have a model of perceptual importance, which determines how the signal is quantized. The quantized signal is then compressed to as few bits as possible. For such compression, we use statistical information about speech signals.

The decoder at the receiving side, reverses the steps by decompression and dequantization.

Pre-processing operations would typically include noise attenuation and voice activity detection (see below).

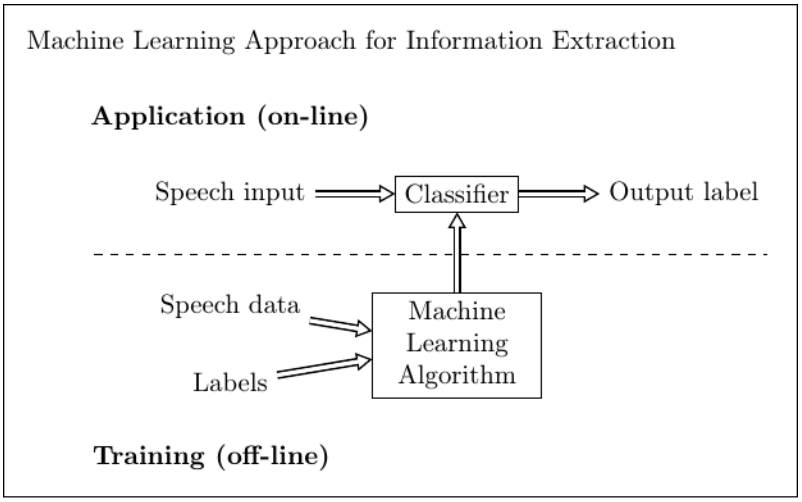


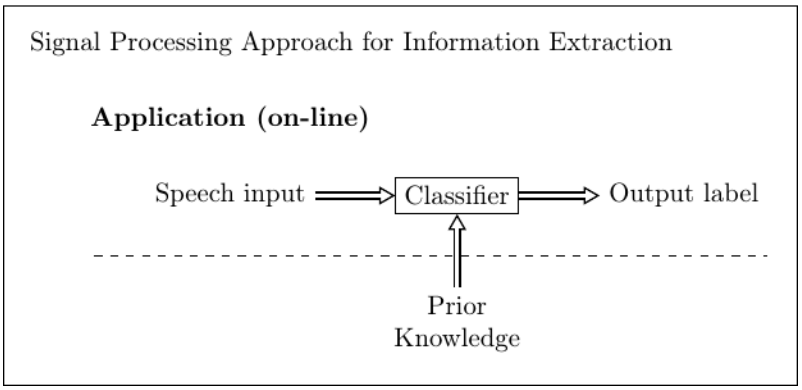

#### Information extraction

We can extract many types of information from a speech signal, like text content and speaker identity. Many such forms of information can be categorized by *labels*, that is, we give a label to a particular speech signal. That label can be for example to word which was pronounced or the speaker identity. Alternatively, such extracted information can be continuous-valued, such as the age of the speaker or mood (how glad/angry are you?), but we can treat both types of information as labels.

Such information extraction methods are today predominantly machine learning methods. A typical configuration is illustrated on the right, where the systems is trained off-line with a database of speech and corresponding labels. Once the system has been trained, it "knows" how to derive labels from speech input, such that in the actual use (application) of the model, it can classify input speech to give an estimate of the label.

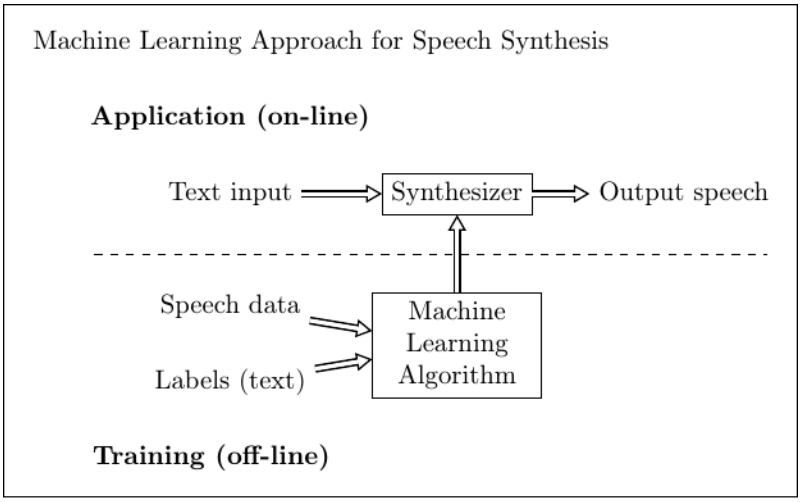
In many cases, information extraction can also be implemented as a signal processing task, where we use prior knowledge of the signal to device our algorithm. For example, for estimating the fundamental frequency (pitch) of a speech signal, we can readily use our knowledge to device efficient algorithms. Such algorithms are usually an order of magnitude simpler than machine learning methods, but if the task is complicated, then the accuracy the output is reduced correspondingly.

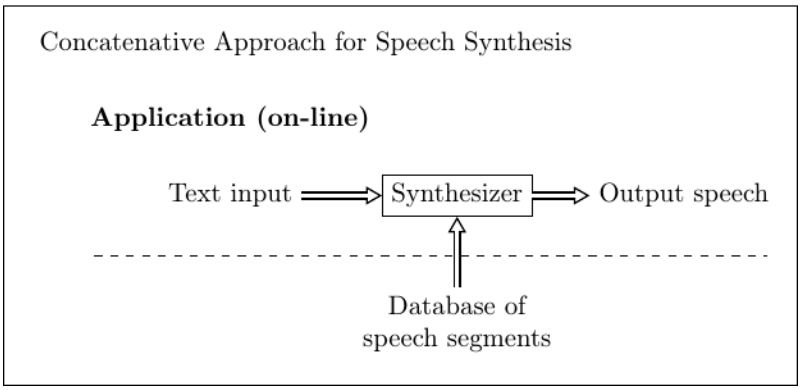



### 2.6.3 Speech synthesis

When we want to make a computer speak, we need a speech synthesiser, which takes text as input and outputs speech. It is thus the reverse of the information extraction -task, in that the roles of speech and labels (text) have been switched (see figure on the right). As in information extraction, also here we can also use simpler methods when applicable. The classical method is concatenative synthesis, where segments of speech, from a database, are fused together to form continuous sentences. Such methods are common for example in public announcement systems (e.g. train stations), where the range of possible announcements is known in advance.



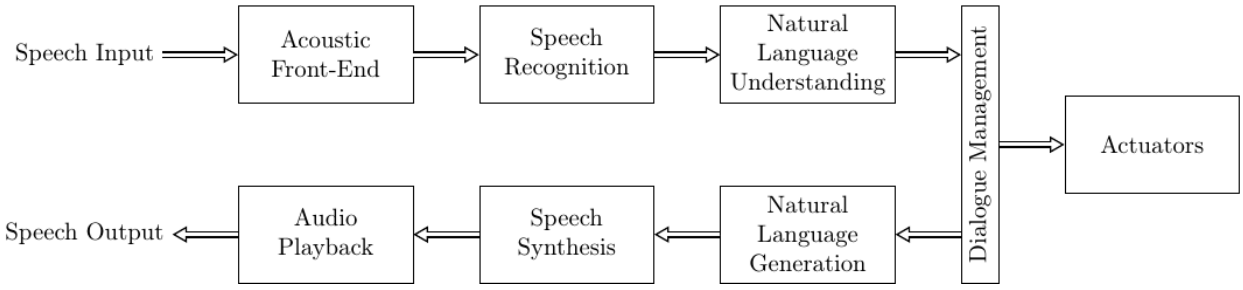

### 2.6.4 User-interfaces

User interfaces can employ speech to accept speech commands and/or respond with speech. For example,

- A car navigator can give instructions with speech, while accepting commands only through the touch (tactile) interface.
- An automatic door can accept speech commands ("Door, open") but give no audible feedback.
- Fully speech operated interfaces, like smart speakers, both accept speech commands and give spoken feedback.

The unidirectional systems with only speech recognition or only speech generation are thus subsets of "full" speech operated systems. The stack of modules of such a complete speech operated system is illustrated on the right. Here the acoustic front-end (can) contain such pre-processing methods described in the following section. Natural language understanding assigns meaning to a sequence of words, dialogue management maps that to a specific action, implemented by the actuator(s), and natural language generation refers to the generation of an answer, in text from.

Information flow in a Voice User-Interface




### 2.6.5 Processing and preprocessing

Irrespective of application, most systems which operate with speech signals suffer from similar type so of problems;

- The main application is expensive to run, so it would be useful to have a pre-processing unit which detects when it makes sense to trigger the main application. For example, when nobody is speaking, it does not make sense to run a speech recognizer.
- Devices are used in real-world environments, where background noises and room echo are common. It would therefore be useful to clean up the signal prior to feeding to the main application.

Such functions typically constitute the acoustic front-end.

#### Voice activity detection

When there is no speech, most speech processing operations are meaningless. Voice activity detection refers to the classification of signal segments according to whether they contain speech or not.

#### Keyword spotting or Wake-word detection

In most practical situations where voice operated devices are present, we want to be able to talk with people and not only the device. In other words, we need to know when the user is speaking to the device and when not. The device then doesn't have to try make sense of speech which is directed to someone else. Wake-word detection (or spotting) refers to a process which is just waiting for a specific word, which triggers the main application. For example, smart speakers of the Amazon Alexa brand wait for the user to say the wake-word "Alexa", and only after identifying that word, it starts the main process.

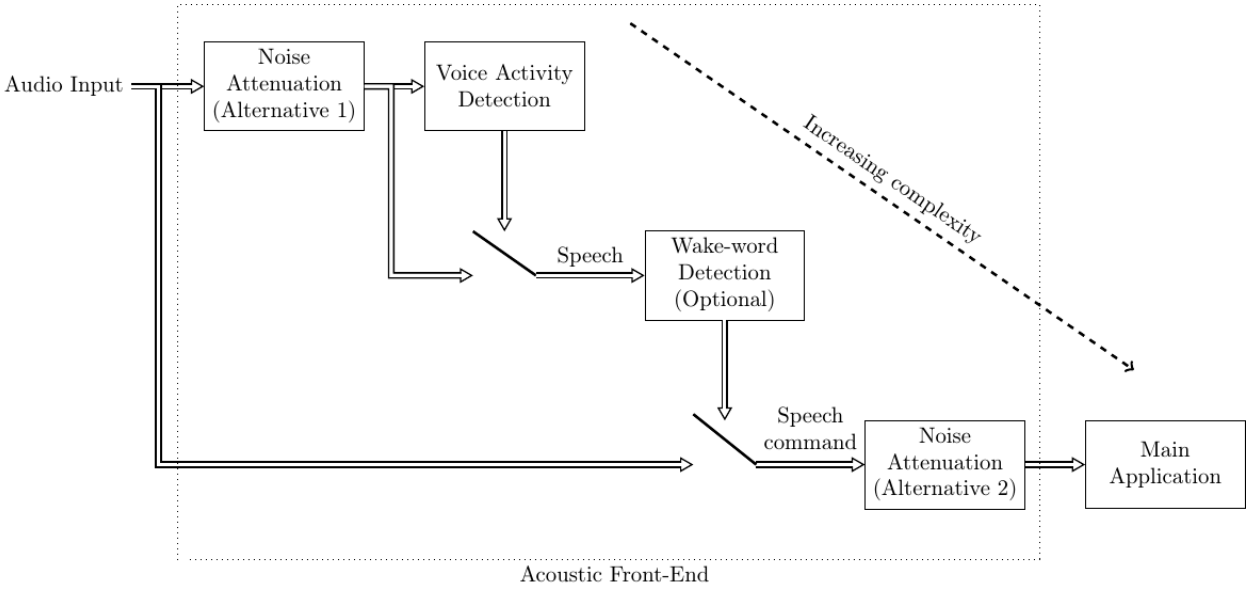
#### Speech enhancement

We can try to remove the detrimental effect of background noises and room echoes with speech enhancement methods. Imagine for example how difficult it is to understand someone on the phone, when the remote speaker is standing on a busy street loud cars driving by. With noise attenuation we can reduce such noises to make speech more pleasant to listen at the receiving end. Also any other processing becomes simpler if the speech signal is more clean. Thus a voice user-interface can feature noise attenuation as pre-processing. Note however that also other processes such as voice activity detection and wake-word detection become easier on the clean signal (see alternative 1 in the figure on the right). However, since noise attenuation can be a computationally expensive process, it might be better to apply when we already know that the signal is a speech command (see alternative 2 in the figure on the right), though this will reduce the accuracy of voice activity and wake-word detection.

### Other signal processing

Speech signals can be processed further by an array of different algorithms as desired. For example:

- Signals can be modified for artistic purposes with tools such as auto-tune, where the pitch of a singing voice is modified to match a desired pitch.
- Speech signals can be translated to some other language.
- The identity of a speaker can be hidden (or spoofed) for security purposes like witness-protection, or for illegal activities like fraud.



CHAPTER

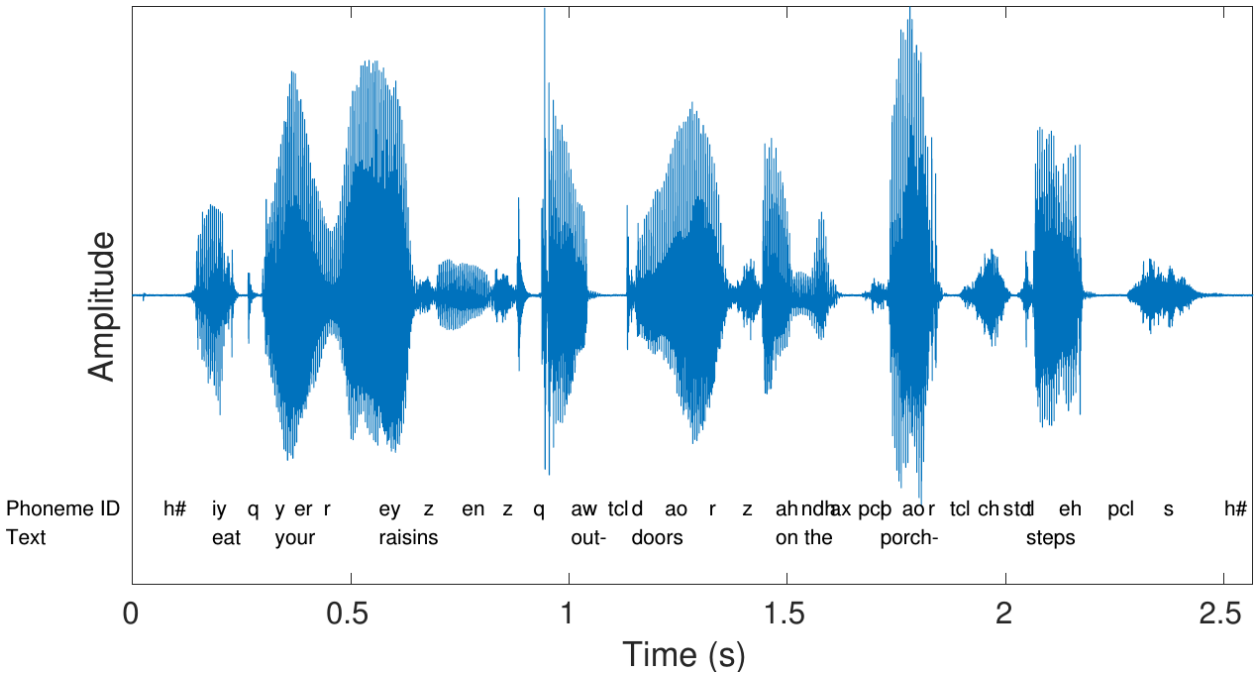
# THREE

# BASIC REPRESENTATIONS

1. *Waveform*
2. *Short-time analysis*
3. *Short-time processing*
4. *Windowing*
5. *Signal energy, loudness and decibel*
6. *Spectrogram and the STFT*
7. *Autocorrelation and autocovariance*
8. *Cepstrum and MFCC*
9. *Linear prediction*
10. *Fundamental frequency (F0)*
11. *Zero-crossing rate*
12. *Deltas and Delta-deltas*
13. *PSOLA*
14. *Jitter and shimmer*
15. Crest factor (Wikipedia)

## 3.1 Short-time analysis of speech and audio signals

### 3.1.1 The speech signal



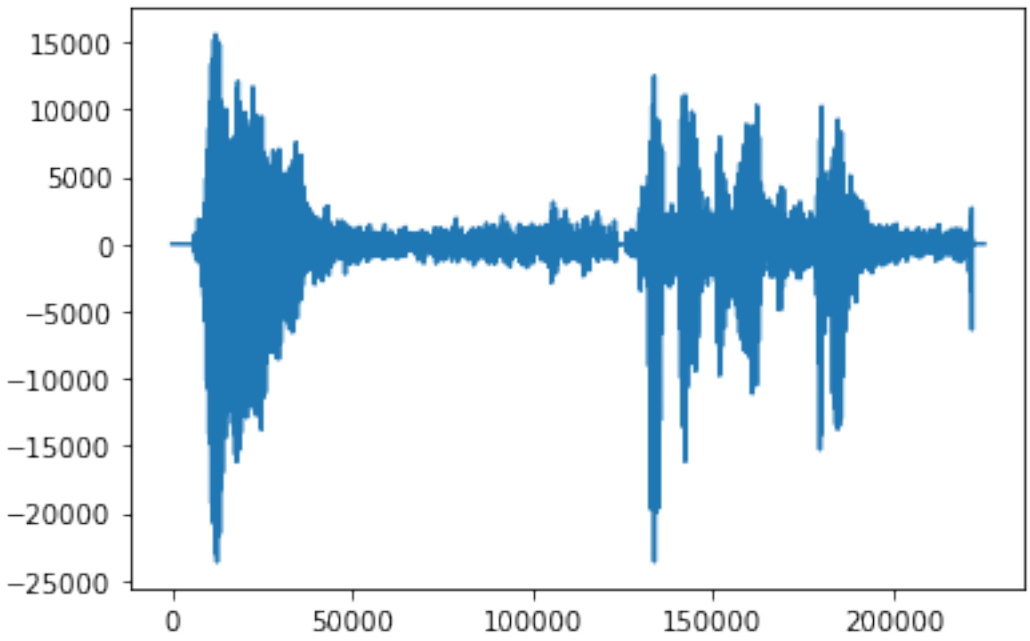

- Speech sounds are acoustic signals
  - Emitted from the mouth
  - Pressure variations travelling through the air
  - Received by the ears
- The figure on the right has an example of a speech sample, together with a time-aligned phonetic and text transcription.
- The acoustic perssure waveform is captured by a microphone
  - The microphone converts it to an analog electric signal
  - Further converted to a digital signal with an AD converter
  - Usually a zero-mean signal (zero is average/ambient pressure level)
- We will here always assume that we have access to a digital representation of the speech signal
- It is a highly time-variant signal – it is more often changing then stationary
- Analysis tools assume that the signal is stationary
  - -> Fundamental problem!
  - For example, if we analyze the (Fourier) spectrum of a speech signal, we would see the spectrum of a mixture of all the phonemes.
  - -> Not useful!
- In *short-time analysis* of speech, the signal is split into small segments (windows).
  - If windows are short enough, signal is stationary

- Can use standard DSP tools within windows

### 3.1.2 Sound example

Let's take a demonstration sound.

```
<IPython.lib.display.Audio object>
```

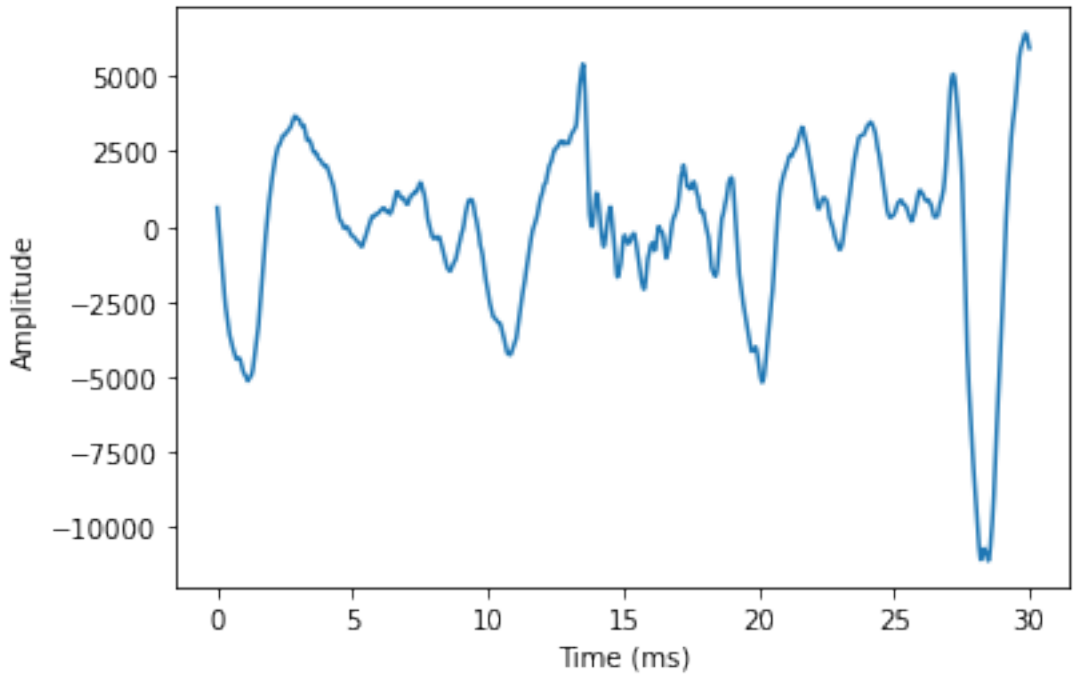



### 3.1.3 What would be a suitable window size?

- Choose a window length in milliseconds.
- Try to find the longest window where segments still look stationary (waveform character does not change much during segment).

```
window_length_ms = 30

window_length = int(window_length_ms*fs/1000)
print('Window length in samples ' + str(window_length))

data_length = data.shape[0]

# choose segment from random position in sample
starting_position = np.random.randint(data_length - window_length)

time_vector = np.linspace(0,window_length_ms,window_length)

plt.plot(time_vector,data[starting_position:(starting_position+window_length)])
plt.xlabel('Time (ms)')
plt.ylabel('Amplitude')
plt.show()
```

```
Window length in samples 1440
```

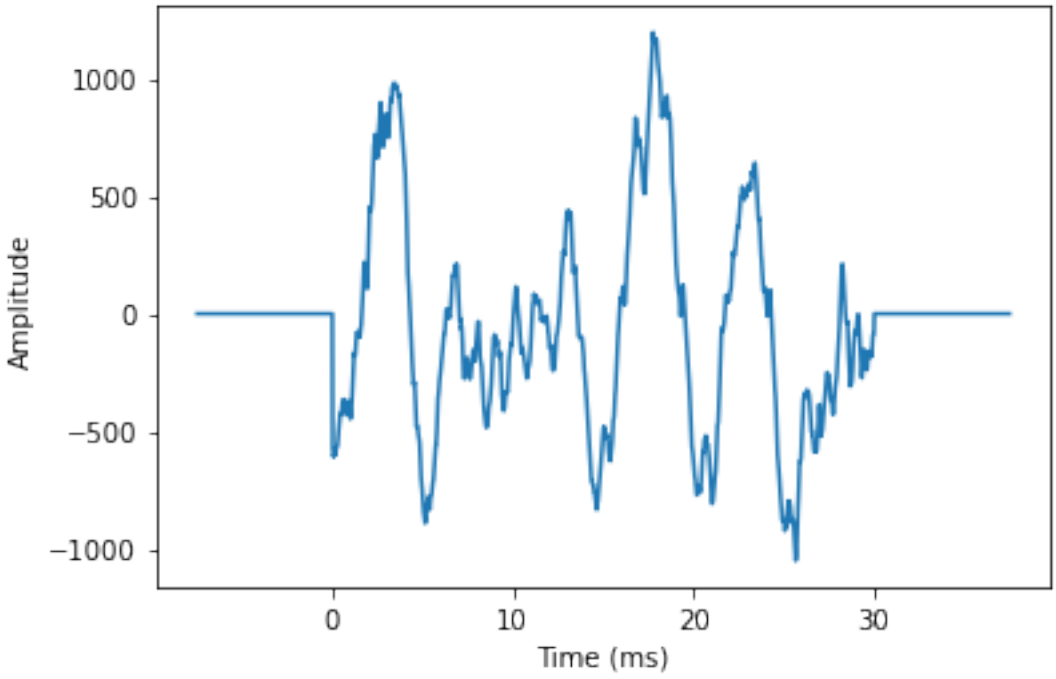



- A window length in the range 20 to 30 ms typically gives fairly stationary windows.
  - An exception is plosives, like /t/ or /p/, where speech suddenly starts (i.e. onset). There we will never get a stationary waveform, but then the single event characterizes the window, so it is then a single-event window and thus "stationary" in a wide sense.
- Longer windows would give more data to analyze = more efficient/accurate
- However, analysis methods such as Fourier analysis, typically assume that a window is stationary.
  - Breaking the assumptions reduce efficiency/accuracy

### 3.1.4 Windowing functions

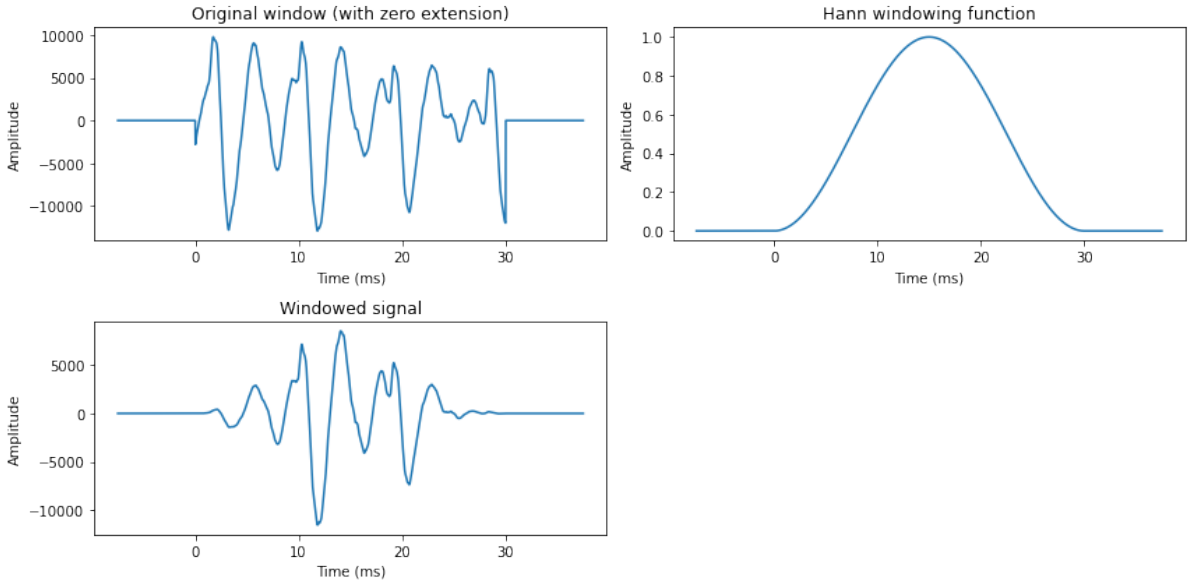
#### Problem

- A signal $\{x_1, \ldots x_n\}$ of a signal is equivalent with a zero-extended signal $\{0, \ldots 0,\ x_1 \ldots x_n,\ 0 \ldots 0\}$
- Heuristic interpretation: The window looks identical to one which has a discontinuity at the border. It does not look stationary = Inaccurate and inefficient.



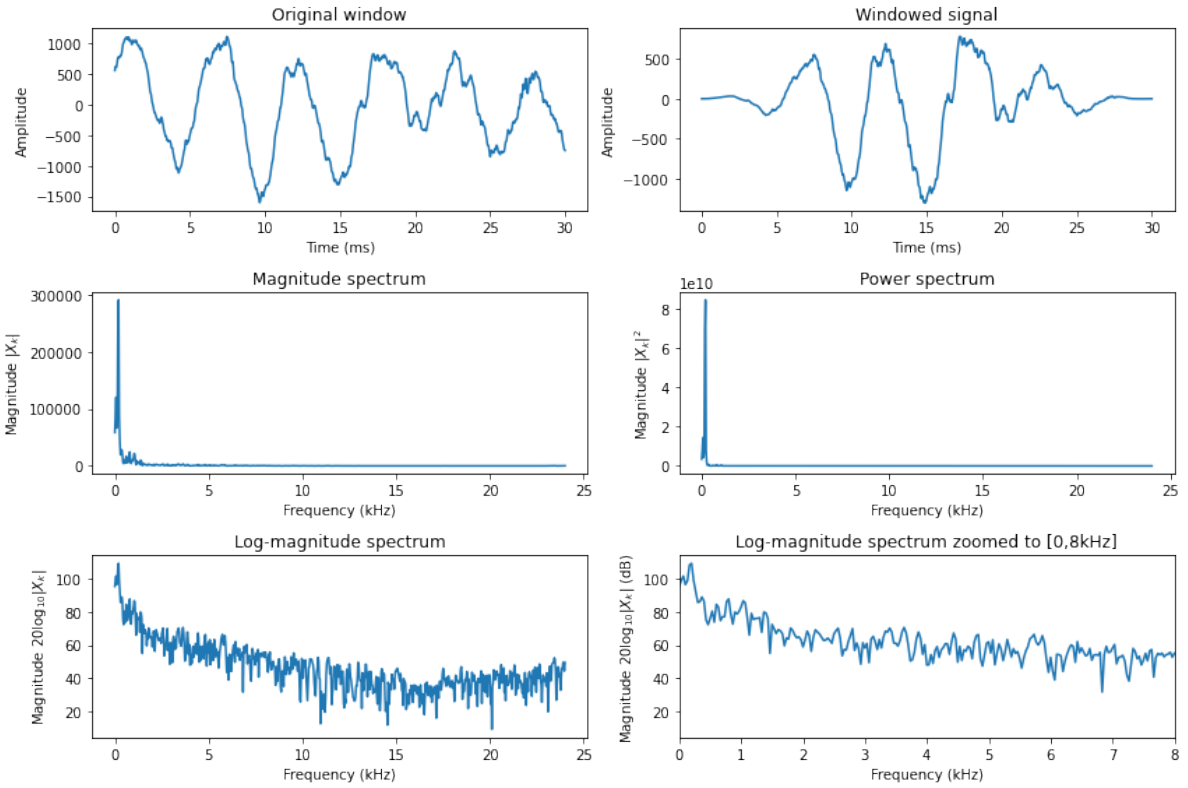

### Solution

- Multiply window with smooth function, the windowing function, which goes to zero at the borders.
- Local structure of the signal is not changed much = similar to original signal.
- Goes smoothly to zero at borders = no discontinuity = no problem.
- Typically, we multiply the signal $x_n$ with a windowing function $w_n$ $\tilde{x}_n := w_n x_n where \begin{cases} w_n > 0, & N \in [0, N-1], \\ w_n = 0, & \text{otherwise.} \end{cases} and w_n goes smoothly to zero at the segment borders \begin{cases} \lim_{n \to 0} w_n & = 0, \\ \lim_{n \to N-1} w_n & = 0. \end{cases}$
- Typical windowing functions include the Hann $w_n = \frac{1}{2}\left[1 - \cos\left(\frac{\pi(n+\frac{1}{2})}{N}\right)\right] and the Hamming windows w_n = 0.54 - 0.46\cos\left(\frac{\pi(n+\frac{1}{2})}{N}\right).$
- The choice of windowing function is usually based on preference with respect to resolution in the spectrum.
- More details at https://en.wikipedia.org/wiki/Window_function




### 3.1.5 Spectral analysis

- The discrete Fourier transform (DFT) can be used to compute the frequency spectrum $\tilde{X}_k = \sum_{n=0}^{N-1} \tilde{x}_n e^{-i2\pi kn/N}, where the windowed signal is \tilde x_n = w_n x_n$.
- Consequences
  - The Fourier transform is $\tilde{X}_k = W_k * X_k$ where '$*$' is the convolution.
  - The true spectrum $X_k$ is smoothed with windowing function
  - Observation is biased (but best we can do for a non-stationary signals)

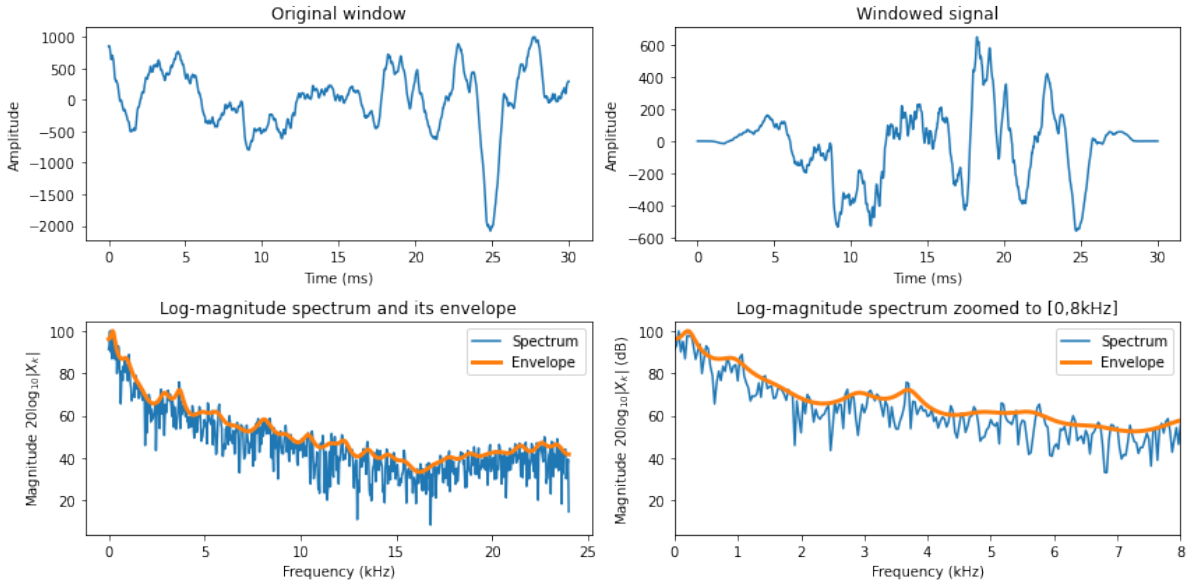



- Observations
  - The DFT is complex-valued and cannot be easily visualized as such.
  - The absolute $|X_k|$ or absolute squared $|X_k|^2$ value of the spectrum can be plotted, but different frequencies have very different ranges so it is hard to see anything useful.
  - In the log-magnitude spectrum $\log|X_k|$ spectral features are much better visualized.
  - The usual unit is decibel, $20\log_{10}|X_k|$ (dB).
- In fact, the log-intensity scale corresponds roughly to a perceptual scale, that is, it corresponds more or less to how humans perceive loudness for individual frequencies.

### 3.1.6 Speech features in the spectrum

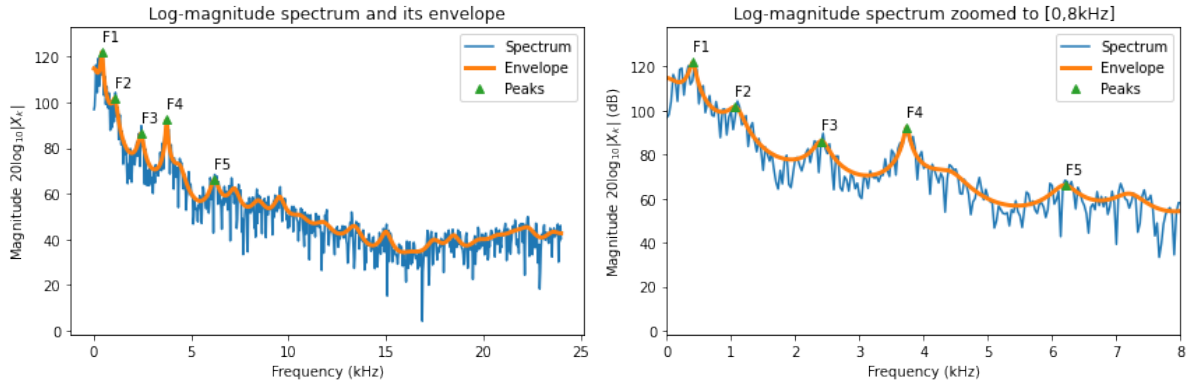
#### Envelope

The macro-shape of the spectrum is known as the envelope. It can be also be defined as the smooth shape which connects spectral peaks.




## Formants

- Peaks of the envelope are known as *formants*. They are thus high-energy areas of the spectrum. The formants are numbered from left to right, F1, F2, F3 etc.
- The location of formants uniquely identify vowels. In other words, each vowel is defined by a unique combination of the two first formants.

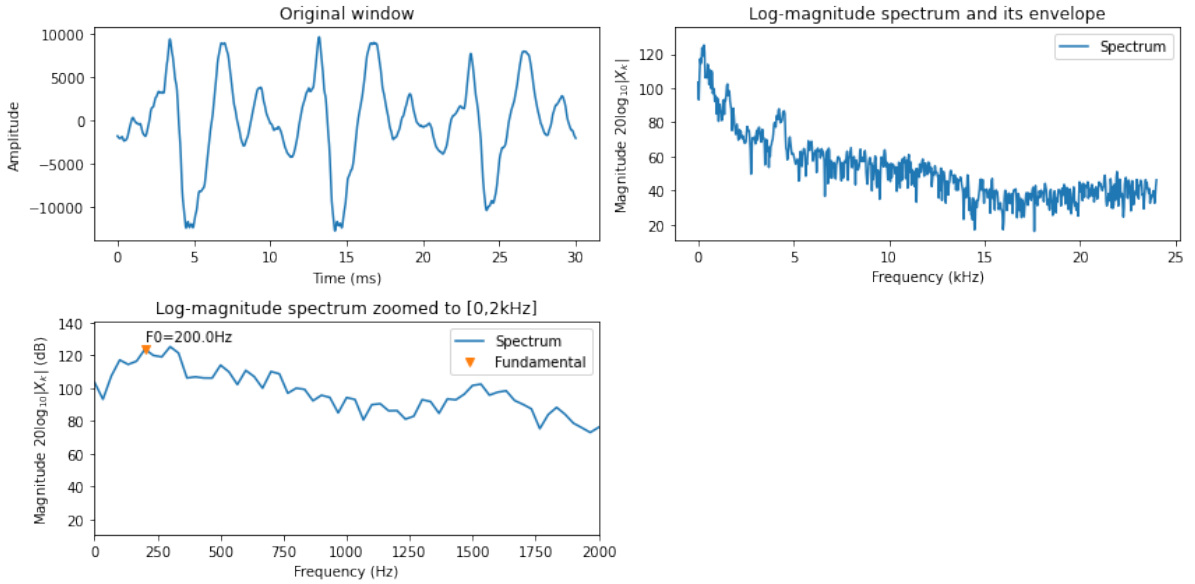



## Fundamental frequency

Note: A more thorough description of the fundamental frequency is found at Fundamental frequency.

Voiced phonations are “more or less” periodic or equivalently, we call them quasi-periodic. The periodicity is generated by the vocal folds which oscillate in the airflow, periodically closing the passing airflow to open again, with a frequency approximately in the range 80 to 400 Hz. It is visible in the waveform as a repeated waveform, in the spectrum as a comb-structure, that is, relatively densly spaced peaks in the spectrum.

The fundamental frequency is often called the F0. It is a bit of an unfortunate choice, because it can be confused to the formants, which are F1, F2, F3, although there is no connection between formants and the fundamental frequency.




## Harmonics of the fundamental

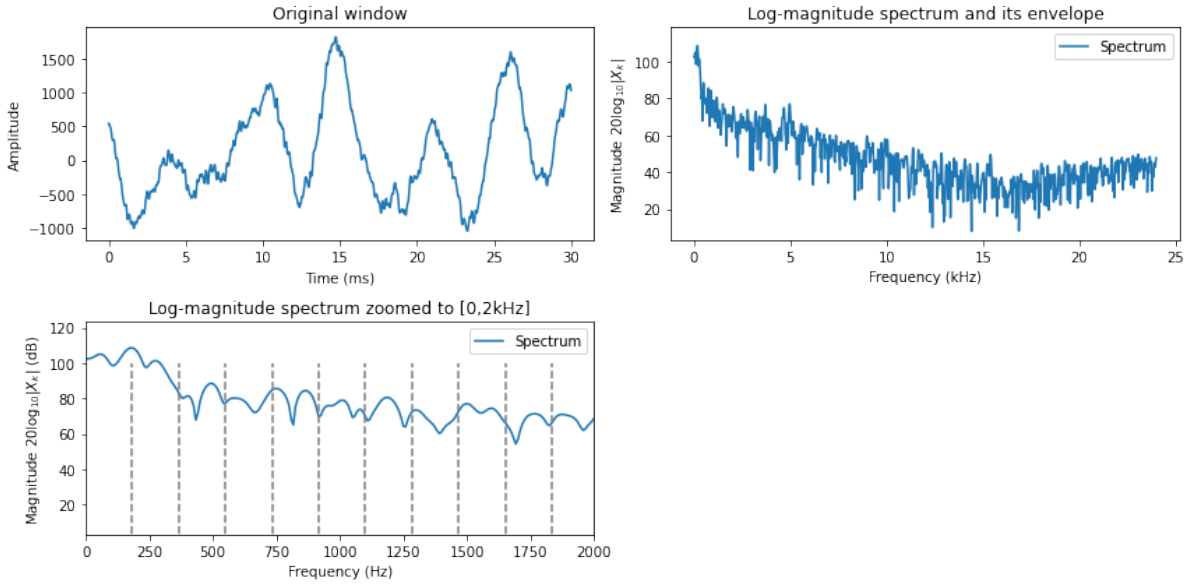
Observe that voiced signals are harmonic. That is, when the glottal folds oscillate, the main event in each oscillation is when the glottal folds smash into each other. This event forms a node in the waveform, which means that all sub-components of the waveform must align at this point. If the length of the glottal period is $T$, then any waveform which has the length $T/k$, where $k$ is a positive integer would then be periodic with length $T$. The frequencies of the harmonics are correspondingly $kF_0$, for a fundamental $F_0$.

In other words, in the spectrum, we will see a sequence of peaks at frequencies $kF_0$. Such a structure is known as a comb-structure.

Observe that the demonstration code will always return *some* fundamental frequency, even if the signal does not really have any periodicity. Similarly, it will always show a harmonic structure even if there is none. This is though a typical feature of speech analysis; measurements are noisy at best, and sometimes the desired effect is not even present at all. The software however must be somehow able to cope with that.

Typical errors in the observation are

- Sometimes there is no fundamental frequency, but a harmonic model will still be plotted
- Sometimes the fundamental frequency estimate is inaccurate, such that the higher harmonics are very much inaccurate (why?)
- Sometimes a multiple of the fundamental is incorrectly identified as the fundamental. Then the identified harmonics only cover some of the true harmonics.




### 3.1.7 Spectrogram

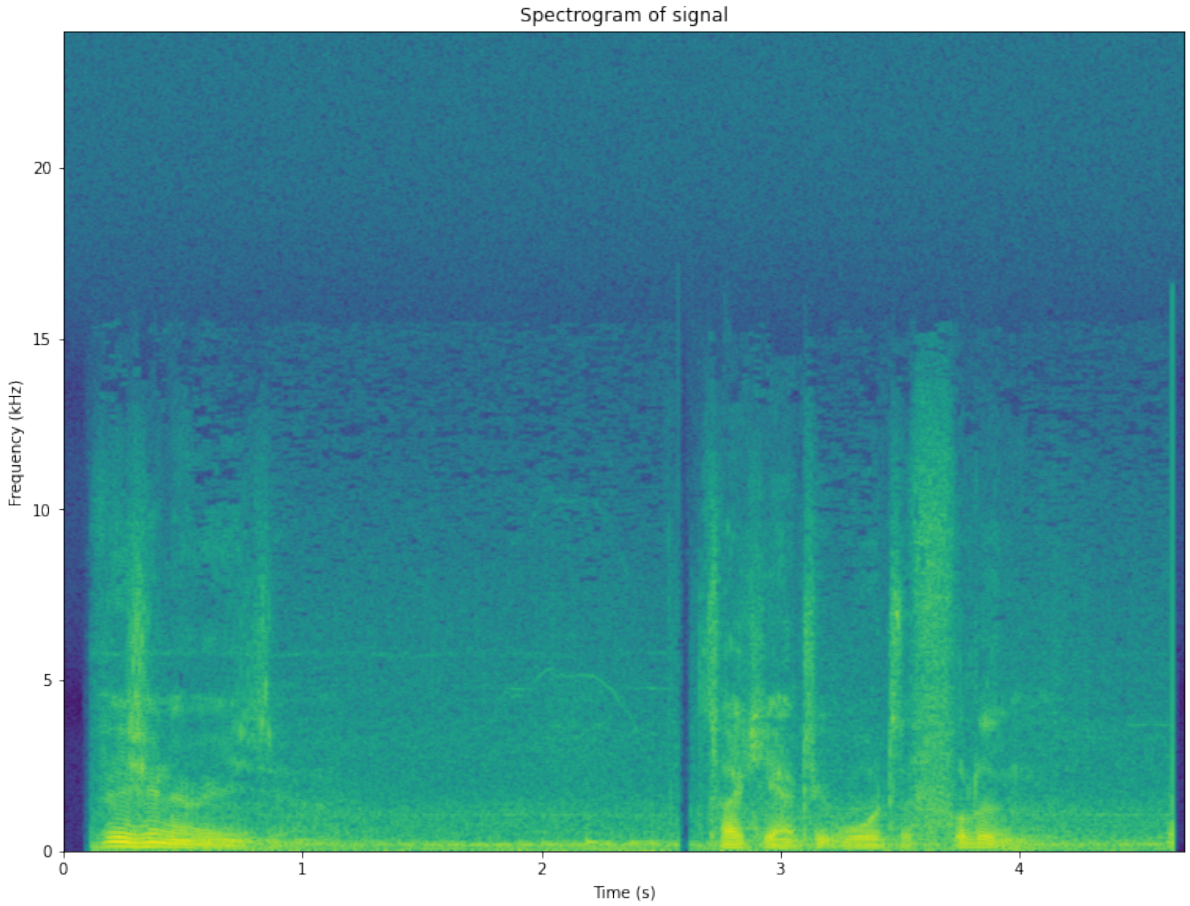
One window gives a "snapshot" image of the signal. Analysis of multiple, consecutive windows give a "movie". We can therefore analyze how the signal is changing over time.

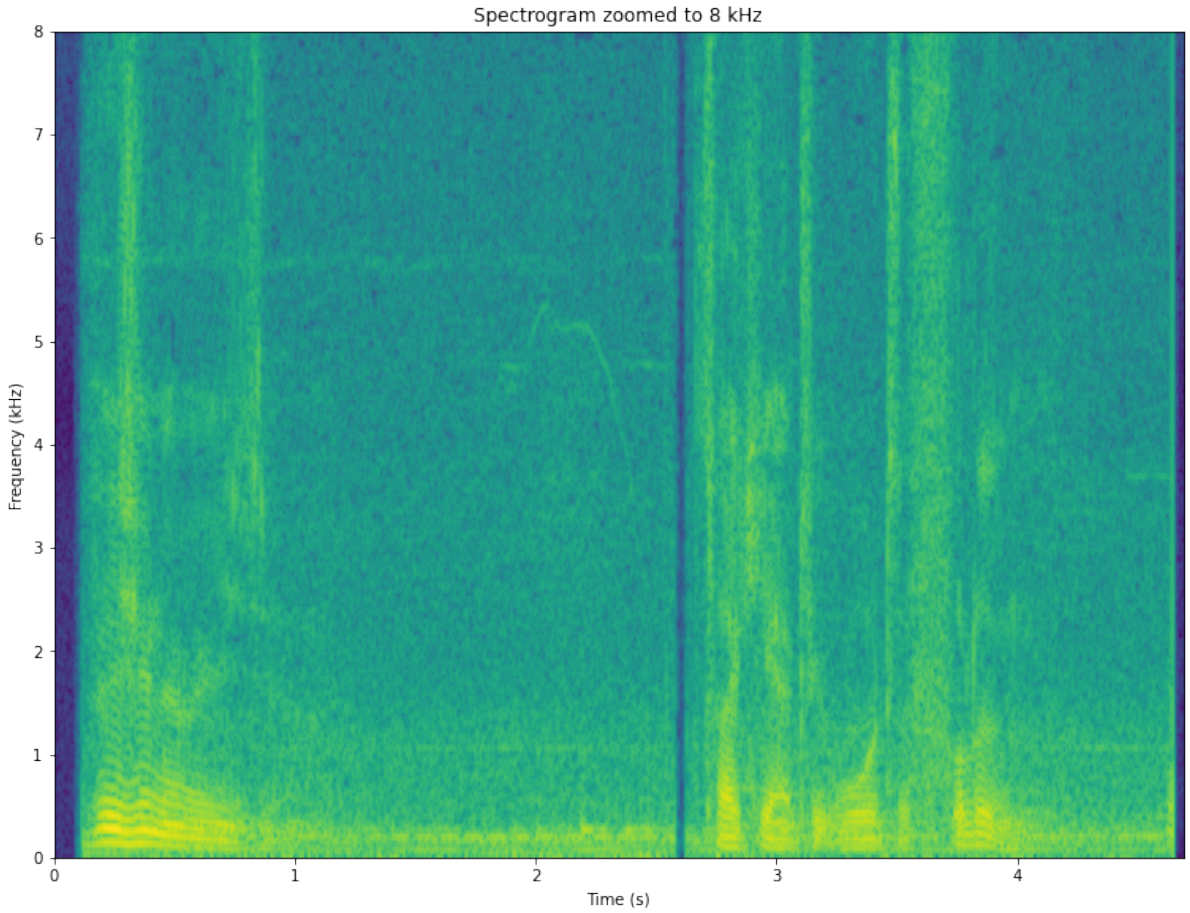
We use a sliding window $\tilde{x}_{n,k} = w_{n-k}x_n whereeachvalueof$k$ gives a different snapshot of the signal.

By taking the DFT of each window, we obtain the *short-time Fourier transform (STFT)*, which can be plotted as a *spectrogram*-representation of the signal. Since many of the fundamental features of speech signals are visible in the spectrum, we can then expect to see the same features also in the spectrogram. Moreover, we will be able to see how these features change over time.

Algorithm:

1. At position $k$, apply windowing (typically, Hamming windowing) to obtain segment of the signal of length $N$.
2. Apply the fast Fourier transform to obtain the spectrum $X_k(\omega)$.
3. Take the logarithm of the absolute value $20 \log_{10} |X_k(\omega)|$ to obtain the logarithmic spectrum.
4. Advance position by $K$, that is, $k := k + K$ and return to 1.


Spectrogram of signal
Frequency (kHz)
0
5
10
15
20
0
1
2
3
4
Time (s)




### Exercise

- What is the fundamental, its harmonics and where can you see them change?
- Where are the formants?
- Can you find vertical lines in the plot? Listen to the corresponding audio; can you find out what the vertical lines correspond to?

### Accuracy

The resolution or accuracy of the spectrogram depends on

- Step between windows
- Length of windows
- Amount of zero-extension of signal
- Shape of windowing function

We will discuss each (except the shape of the windowing function) below.

Analyzing windows is similar to sampling a signal. The more often you take windows (or samples), the higher the resolution over time will be. The price is a larger amount of data. Moreover, at some point, there really is not that much to be

gained by a denser sampling. Similarly, we can reduce the density to lower computationaly cost, but at some point you start loosing essential information.

The number of points in the spectrum corresponds to the length of the vector given to the DFT. That is, a longer window will give more spectral components and thus a higher spectral resolution. More accurate is good in the sense that, for example, from a accurate spectrum we can estimate the fundamental frequency more accurately. However, the compromise is due to the inherent changes in speech signals. Because speech signals are continuosly changing, when we take longer windows, we will more frequently have instances where a single window contains two or more distinct sounds, such that the spectrum is a combination of these sounds. The spectrum is then smeared and less accurate, even if we wanted to get a more accurate spectrum. This is a compromise where we have to always seek a balance depending on the objectives in each particular application.

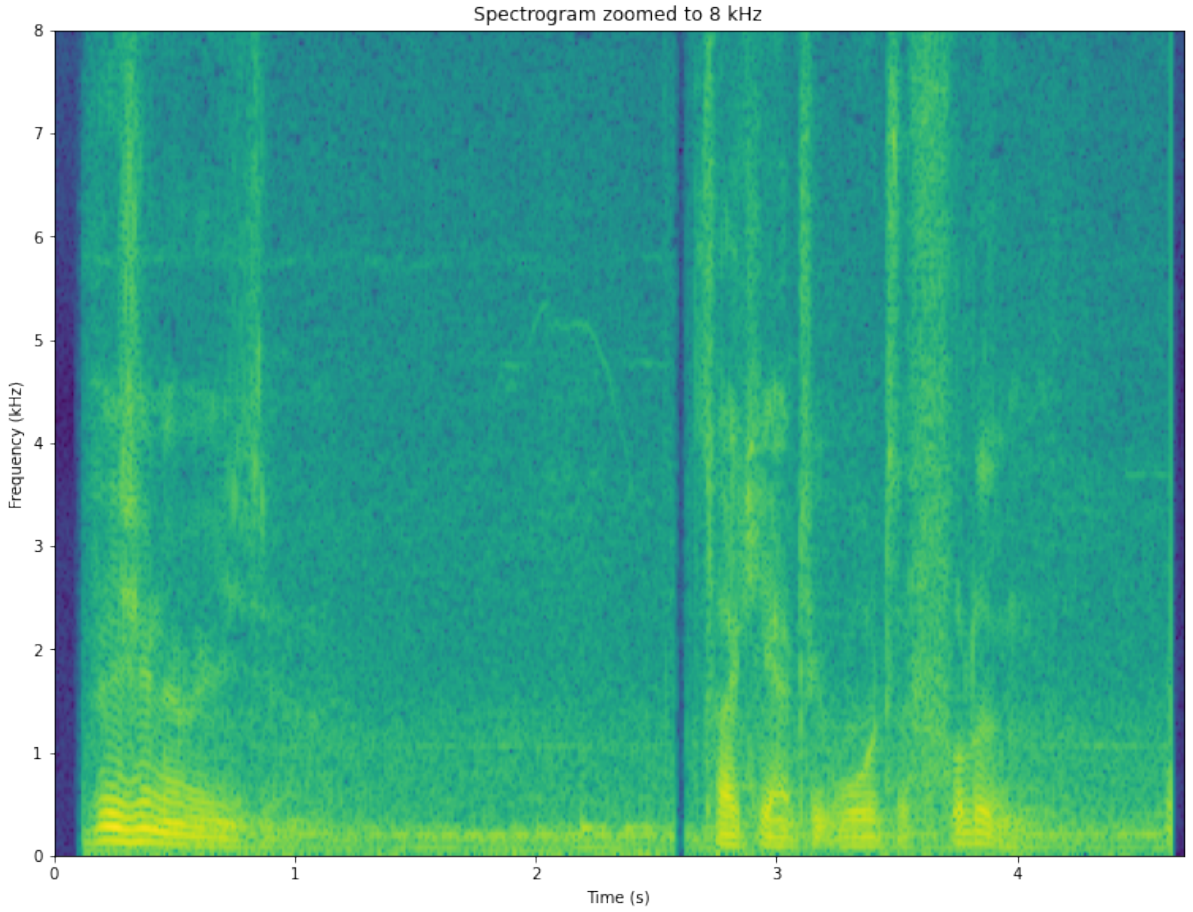
A second method for increasing the accuracy of the spectrum is known as zero-extension. It is somewhat artificial in the sense that it does not actually bring any new information to the analysis. It is achieved by simply extending the windowed signal by zeros. Since it thus has more samples, the spectrum will have more resolution. We can then, for example, find a more accurate estimate of the fundamental frequency. However, since zero-extension does not introduce any new information into the analysis, it should be treated as a interpolation method and it thus gives an interpolated spectrum. In analysis applications that is often a desirable feature.

```
# try different length, step and zero-extension factors
window_length_ms = 30
window_step_ms = 10
zero_extension_factor = 1 # multiplicator by which we extend the length of the␣
 ↪spectrum

# derived parameters
window_length = int(window_length_ms*fs/1000)
window_step = int(window_step_ms*fs/1000)
total_length = int(window_length*zero_extension_factor)


window_count = int(np.floor((len(data)-window_length)/window_step)+1)
spectrum_length = int((total_length+1)/2)+1
windowing_function = np.sin(np.pi*np.arange(0.5,window_length,1)/window_length)**2

spectrogram = np.zeros((window_count,spectrum_length))
time_vector = np.linspace(0,window_length_ms,window_length)
frequency_vector = np.linspace(0,fs/2000,spectrum_length)


for k in range(window_count):
    starting_position = k*window_step

    data_vector = data[starting_position:(starting_position+window_length),]
    window_spectrum = np.abs(scipy.fft.rfft(data_vector*windowing_function,n=total_
 ↪length))

    spectrogram[k,:] = window_spectrum


black_eps = 1e-1 # minimum value for the log-value in the spectrogram - helps making␣
 ↪the background really black

import matplotlib as mpl
default_figsize = mpl.rcParamsDefault['figure.figsize']
mpl.rcParams['figure.figsize'] = [val*2 for val in default_figsize]
```

(continues on next page)

(continued from previous page)

```
mpl.rcParams['figure.figsize'] = [val*2 for val in default_figsize]
plt.imshow(20*np.log10(np.abs(np.transpose(spectrogram))+black_eps),aspect='auto',
 ↪origin='lower',extent=[0, len(data)/fs,0, fs/2000])
plt.xlabel('Time (s)')
plt.ylabel('Frequency (kHz)')
plt.axis([0, len(data)/fs, 0, 8])
plt.title('Spectrogram zoomed to 8 kHz')
plt.show()
```




### Speech features visible in the spectrogram

- Comb-structure of the fundamental frequency F0 is visible as horizontal lines. Changes in F0 over time can be easily followed.
- Time-structure of speech is clearly visible. We can spot breaks in speech as well as changes in phonemes.
- Unvoiced phonemes (speech sounds without a fundamental frequency) are also visible. Typically sustained consonants (fricatives, such as /h/ or /s/) have noise at the high frequencies. Stop consonants (/p/, /k/, /t/) are short wide-band bursts.

### 3.1.8 Short-time spectral analysis summary

We have presented a spectral *analysis* methods for speech.

- Spectral analysis can nicely visualise many of the most important features of speech signals.
- The decibel scale corresponds roughly to perceptual sensitivity.
- Short-time Fourier transform is the most common speech analysis method.
- Spectrogram is visualizes spectral behaviour over time.

## 3.2 Short-time processing of speech signals

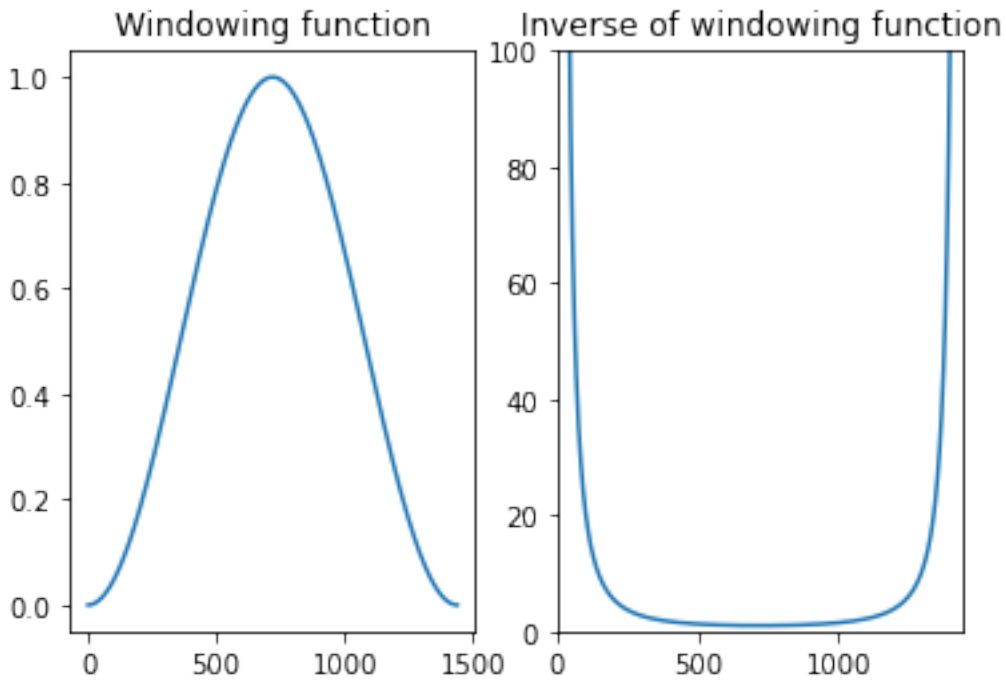
In the short-time *analysis* section we already discussed speech signals and analysis methods, and found that we need to split the signal into shorter segments and apply windowing functions. Here we discuss what extra steps we need to consider when we want to *process* signals, that is, when we want to modify the signal.

To process a windowed signal, we thus need the steps:

1. windowing
2. time-frequency transform such as the DFT (*optional*)
3. apply the desired processing
4. inverse time-frequency transform (*when DFT was applied*)
5. reverse of windowing (?).

Here the analysis steps 1-2 were already discussed and the desired processing is whatever modification to the signal that you might have. The question is thus about the inverse transforms in steps 4 and 5. Time-frequency transforms such as the discrete Fourier transform (DFT) and the discrete cosine transform (DCT) are orthonormal and have well-known fast algorithms for their inverses. The main challenge is thus the *"reverse of windowing"*, whatever that might be.

The direct approach of just multiplying with the inverse of the windowing function has a problem.



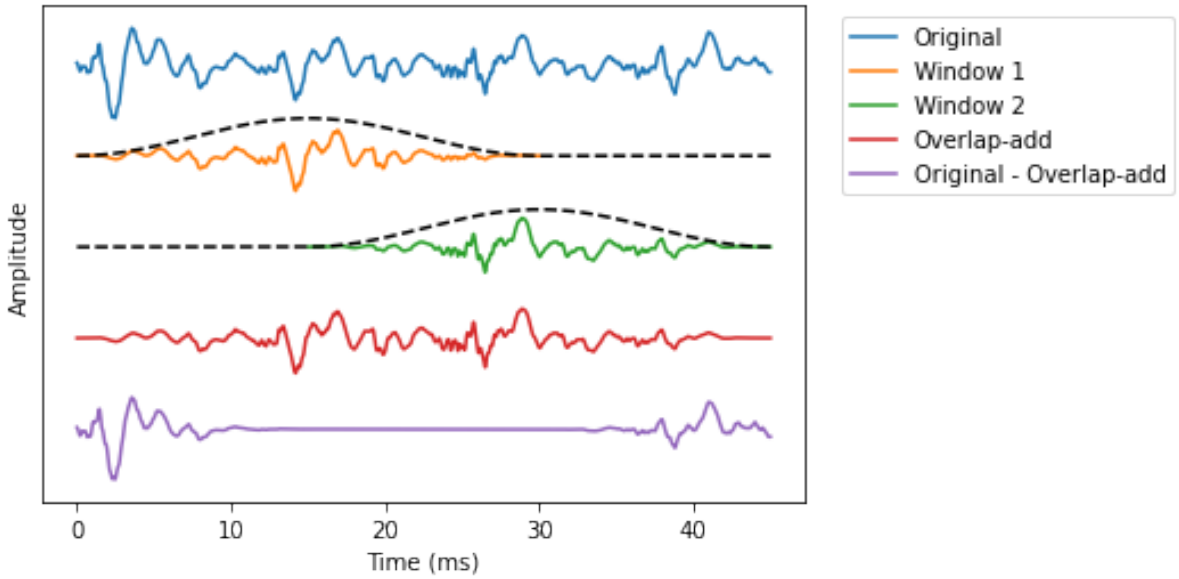

Namely, near the ends of the window, the windowing function goes smoothly to zero and thus the inverse of the windowing function approaches infinity. That clearly leads to numerical issues; very small changes in the windowed signal can lead to arbitrarily large samples after inverse windowing.

We thus need a method which does not rely on inverting the windowing function.

### 3.2.1 Overlap-add

A majority of modern speech processing is based on the overlap-add method, where the input signal is windowed in overlapping segments, such that when the overlapping parts are added together, we can perfectly reconstruct the original signal. It is like a cross-fade between subsequent windows.

The most common scenario where overlap-add cannot be used are applications which require a very low algorithmic delay. For example, stage microphones at concerts and theaters require a low-delay, such that interaction between people on the stage is not disrupted. Almost always otherwise, the best option is overlap-add.



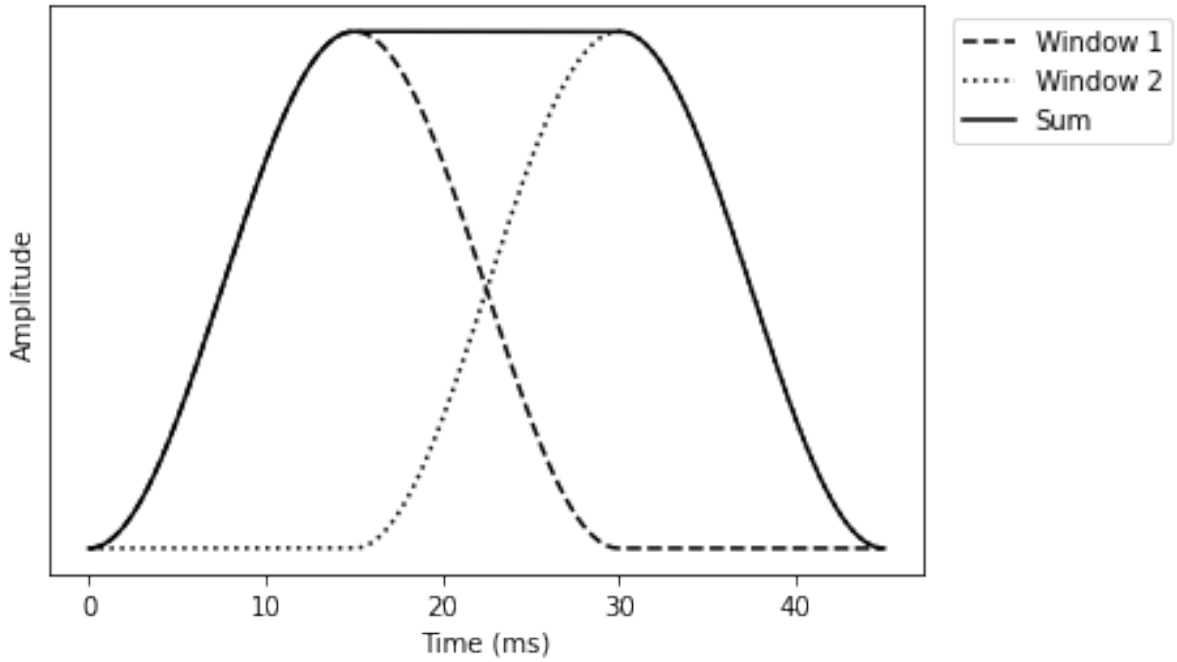

In the above example, we see two overlapping windows extracted from the original signal. When we add the two together we obtain the "Overlap-add" signal which, we subtracted from the original, is zero in the whole region of the overlap. At the left and right ends, where we do not have an overlapping window, we do not get perfect reconstruction.

### Windowing and reconstruction

Overlapping segments are added together (fade-in/fade-out). Windowing should be chosen such that reconstruction is exactly equal to the original.

- Let original signal be $x_n$.
- The left and right parts of overlap windows are then $w_{L,n}$ and $w_{R,n}$ and the windowed signals are $w_{L,n}x_n$ and $w_{R,n}x_n$.
- The reconstruction is $\hat{x}_n = w_{L,n}x_n + w_{R,n}x_n = (w_{L,n} + w_{R,n})x_n$.
- With $w_{L,n} + w_{R,n} = 1$ we obtain *perfect reconstruction*, $x_n = \hat{x}_n$.

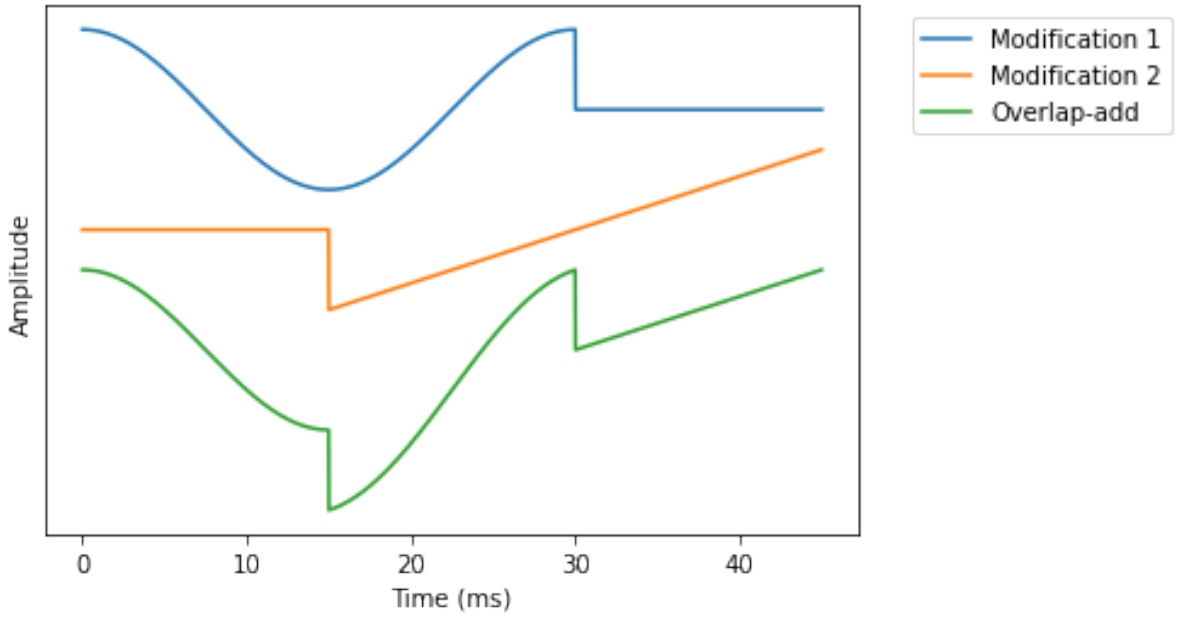
The window in the above example already adheres to this requirement.



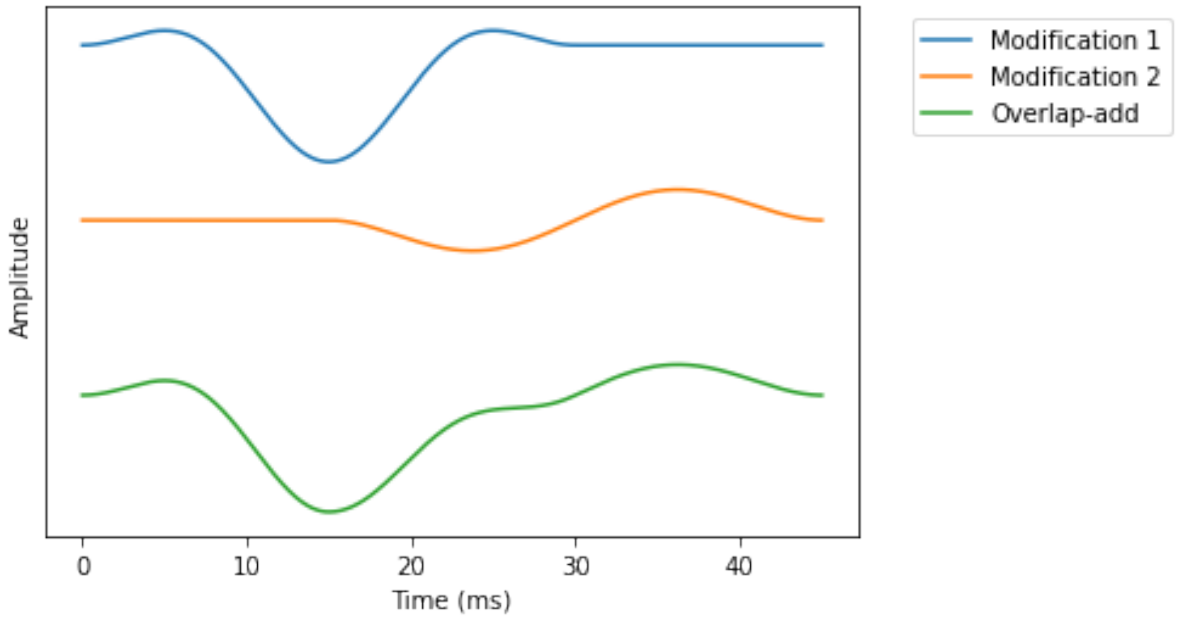

### Windowing and processing

The whole point of windowing for processing was that we could modify the windowed signal and that we can then synthesise the modified signal. Suppose $x' := wx$ is the windowed signal and the modified signal is $\hat{x}'$, such that the modification part is $e = x' - \hat{x}$. In overlap-add, we would then take multiple modified windows $\hat{x}'_k$ and add them together. We have already seen that the part which corresponds to the original signal $x'$ will perfectly reconstruct to the original signal. The question is however what happens to the modification part?

With a direct implementation, we would just add the modifications together. There is then no guarantee that consecutive modifications play nicely together and we can, for example, have discontinuities between windows.



We therefore need to multiply also the modified windows with a windowing function.




### Algorithm ``Overlap-add''

Let $w_{in,n}$ and $w_{out,n}$ be the input and output windowing functions.

1. The input signal is $x_n$ is windowed at the input to obtain $w_{in,n}x_n$.
2. The windowed signal is modified with $e_n$ to obtain $w_{in,n}x_n + e_n$.
3. We apply an output window $w_{out,n}$ to the modified signal to obtain $w_{out,n}(w_{in,n}x_n + e_n)$.
4. Add subsequent, overlapping windows together to obtain the output signal.

Then if the output windows go to zero at the border, then the output signal will be continuous. With no-modification $e_n = 0$, the output is $w_{out,n}w_{in,n}x_n$. In other words, if the left and right parts add up $$w_{L,out,n}w_{L,in,n} + w_{R,out,n}w_{R,in,n} = 1$$ then we have perfect reconstruction.

If we the modification is uniform white noise, then the modification part overlap is $w_{L,out,n}e_{L,n} + w_{R,out,n}e_{R,n}$. The energy expectation of the modification is then

$$E\left[\left(w_{L,out,n}e_{L,n} + w_{R,out,n}e_{R,n}\right)^2\right] = \left(w_{L,out,n}^2 + w_{R,out,n}^2\right) E[e_n^2].$$

If $w_{L,out,n}^2 + w_{R,out,n}^2 = 1$ then output energy is uniform. To fulfil the criteria, we can set the input and output windows to be the same $w_{in,n} = w_{out,n}$.

We can then require that (*Princen-Bradley condition*)

$$\boxed{w_{L,n}^2 + w_{R,n}^2 = 1}.$$

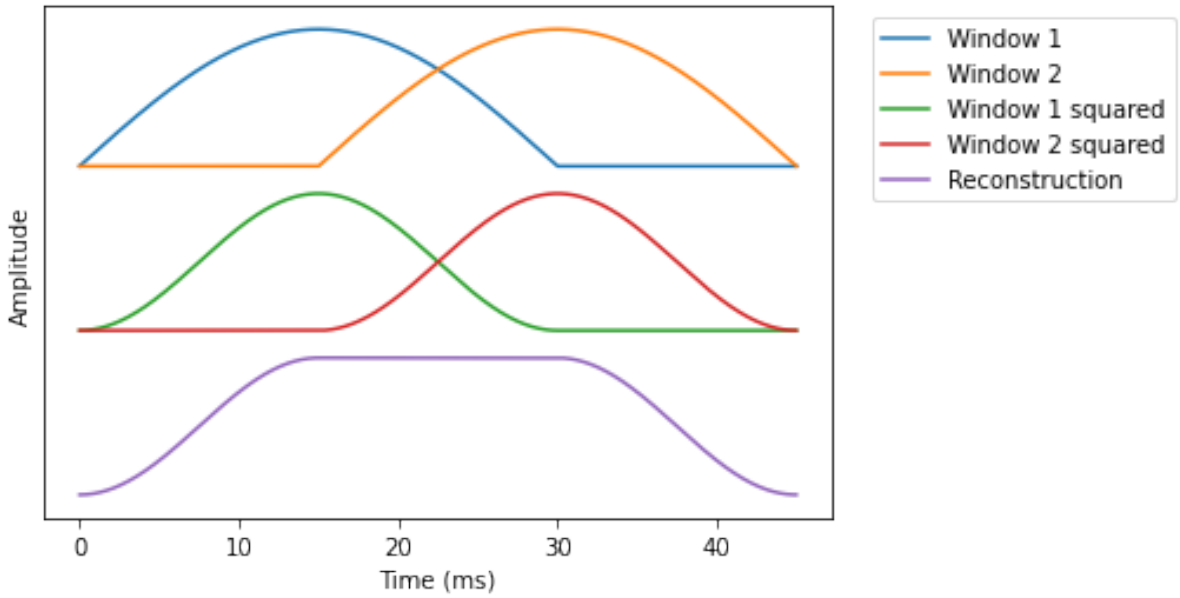
and that $w_n$ goes to zero at the borders. Overlap-add obtains the following properties:

- **Perfect reconstruction** – If there is no modification, we can reconstruct the original signal.
- **Continuous output** – There are no discontinuities.
- **Uniform noise energy** – Output noise does not have temporal structure (noise has a smooth energy envelope).

One such windowing function is the half-sine

$$w_n = \sin\left(\frac{\pi n}{N}\right). \qquad \text{(It is the square root of a Hann-window.)}$$

We can readily show that it fulfils the Prince-Bradly condition.




### Overlap-add summary

Overlap-add is a method for windowing a signal such that we can modify the segments *and* reconstruct the modified signal.

*Algorithm*

1. Applying windowing function $w_n$.
2. Modify/process window with your-algorithm-of-choice.
3. Applying windowing function $w_n$ again.
4. Add overlapping segments together to obtain output signal.

Usually we would perform a time-frequency transform on the windowed signal $w_n x_n$ and perform modifications in the frequency-domain. (Almost) all frequency-domain processing algorithms are based on overlap-add.

## 3.2.2 The short-time Fourier transform (STFT)

Overlap-add is typically combined with taking discrete Fourier transforms of the windowed signal, as well as an inverse transforms after processing. This algorithm is known as the *short-time Fourier transform (STFT)* and it is the most commonly used domain for speech and audio processing. It is so common that often when we talk about a time-frequency transform in conjunction with processing algorithms, we implicitly mean the STFT.

*Algorithm*

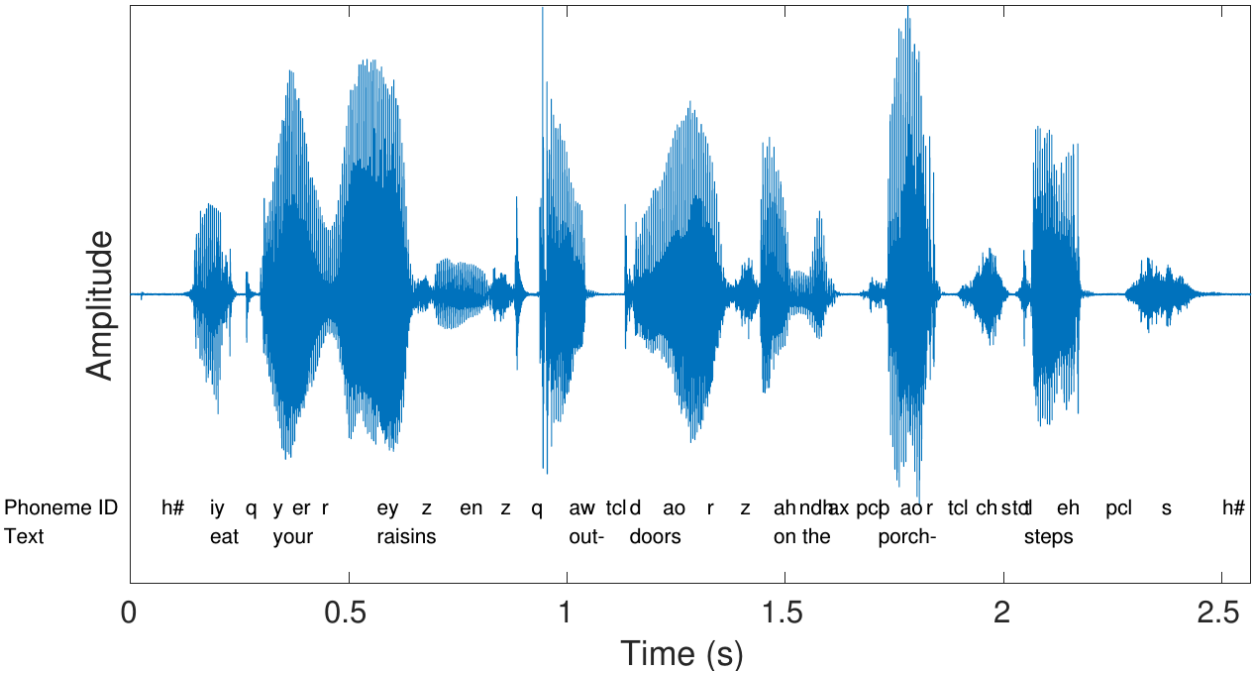
1. Applying windowing function $w_n$.
2. Apply the discrete Fourier transform (DFT) on the windowed signal.

3. Modify/process window with your-algorithm-of-choice.
4. Apply the inverse DFT on the modified windows.
5. Applying windowing function $w_n$ again.
6. Add overlapping segments together to obtain output signal.

## 3.3 Waveform

Speech signals are sound signals, defined as pressure variations travelling through the air. These variations in pressure can be described as waves and correspondingly they are often called sound waves. In the current context, we are primarily interested in analysis and processing of such waveforms in digital systems. We will therefore always assume that the acoustic speech signals have been captured by a microphone and converted to a digital form.

A speech signal is then represented by a sequence of numbers $x_n$ , which represent the relative air pressure at time-instant $n \in \mathbb{N}$ . This representation is known as pulse code modulation often abbreviated as *PCM*. The accuracy of this representation is then specified by two factors; 1) the sampling frequency (the step in time between $n$ and $n+1$) and 2) the accuracy and distribution of amplitudes of $x_n$.



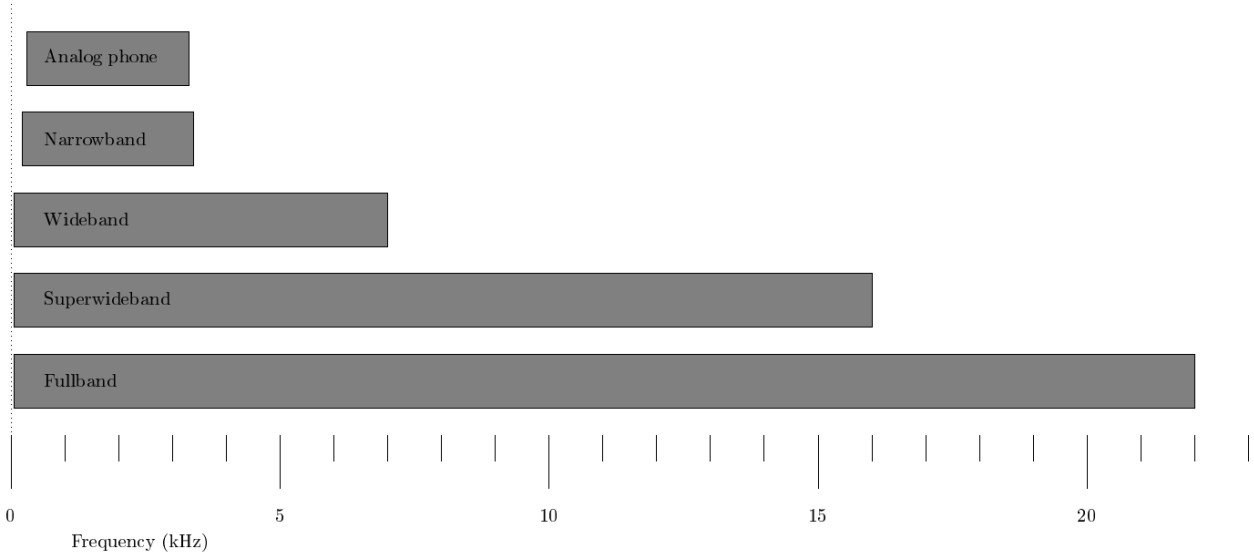

### 3.3.1 Sampling rate

Sampling is a classic topic of signal processing. Here the most important aspect is the Nyquist frequency, which is half the sampling rate $F_s$ and defines the upper end of the largest bandwidth $\left[0, \frac{F_s}{2}\right]$ which can be uniquely represented. In other words, if the sampling frequency would be 8000 Hz, then signals in the frequency range 0 to 4000 Hz can be uniquely described with this sampling frequency. The AD-converter would then have to contain a low-pass filter which removes any content above the Nyquist frequency.

The most important information in speech signals are the formants, which reside in the range 300 Hz to 3500 Hz, such that a lower limit for the sampling rate is around 7 or 8kHz. In fact, first digital speech codecs like the AMR-NB use a sampling rate of 8 kHz known as narrow-band. Some consonants, especially fricatives like /s/, however contain substantial energy above 4kHz, whereby narrow-band is not sufficient for high quality speech. Most energy however remains below 8kHz

such that wide-band, that is, a sampling rate of 16 kHz is sufficient for most purposes. Super-wide band and full band further correspond, respectively, to sampling rates of 32 kHz and 44.1 kHz (or 48kHz). The latter is also the sampling rate used in compact discs (CDs). Such higher rates are useful when considering also non-speech signals like music and generic audio.

Frequency-range of different bandwidth-definitions




### 3.3.2 Static demo

Sound samples at different bandwidths

```
Original (0 to 22050 Hz)
```

```
<IPython.lib.display.Audio object>
```

```
Narrowband (300 Hz to 3.3 kHz)
```

```
<IPython.lib.display.Audio object>
```

```
Wideband (50 Hz to 7 kHz)
```

```
<IPython.lib.display.Audio object>
```

```
Superwideband (50 Hz to 16 kHz)
```

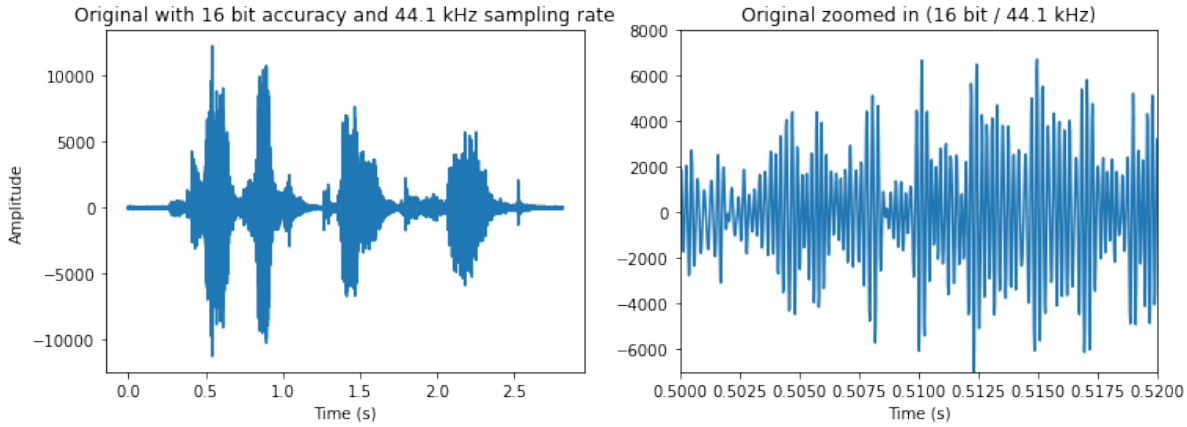
```
<IPython.lib.display.Audio object>
```

```
Fullband (50 Hz to 22 kHz)
```

```
<IPython.lib.display.Audio object>
```

### 3.3.3 On-line demo

#### Original sound sample




```
<IPython.lib.display.Audio object>
```

#### Resampling

```
interactive(children=(IntSlider(value=16000, description='sampling_rate',␣
↪max=44100, min=2000, step=500), Outp…
```

### 3.3.4 Accuracy and distribution of steps on the amplitude axis

In digital representations of a signal you are forced to use a finite number of steps to describe the amplitude. In practice, we must quantize the signal to some discrete levels.

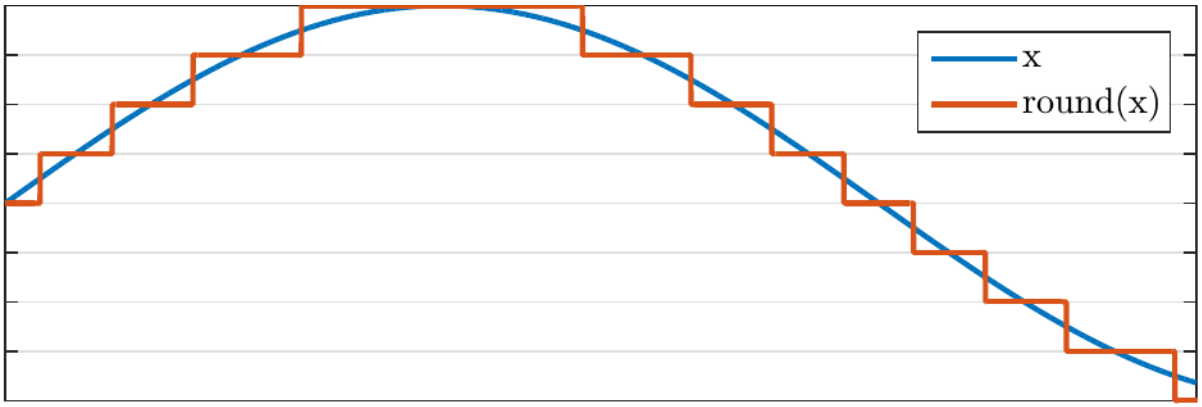
#### Linear quantization

Linear quantization with a step size $\Delta q$ would correspond to defining the quantized signal as

$$\hat{x} = \Delta q \cdot \text{round}(x/\Delta q).$$

The intermediate representation, $y = \text{round}(x/\Delta q)$, can then be taken to represent, for example, signed 16-bit integers. Consequently, the quantization step size $\Delta q$ has to be then chosen such that $y$ remains in the range $y \in (-2^{15}, 2^{15}]$ to avoid numerical overflow.

The beauty of this approach is that it is very simple to implement. The drawback is that this approach is sensitive to the choice of the quantization step size. To make use of the whole range and thus get best accuracy for $x$, we should choose the smallest $\Delta q$ where we still remain within the bounds of integers. This is difficult because the amplitudes of speech signals vary on a large range.




```
interactive(children=(IntSlider(value=16000, description='sampling_rate',␣
↪max=44100, min=2000, step=500), IntS…
```

### Logarithmic quantization and mu-law

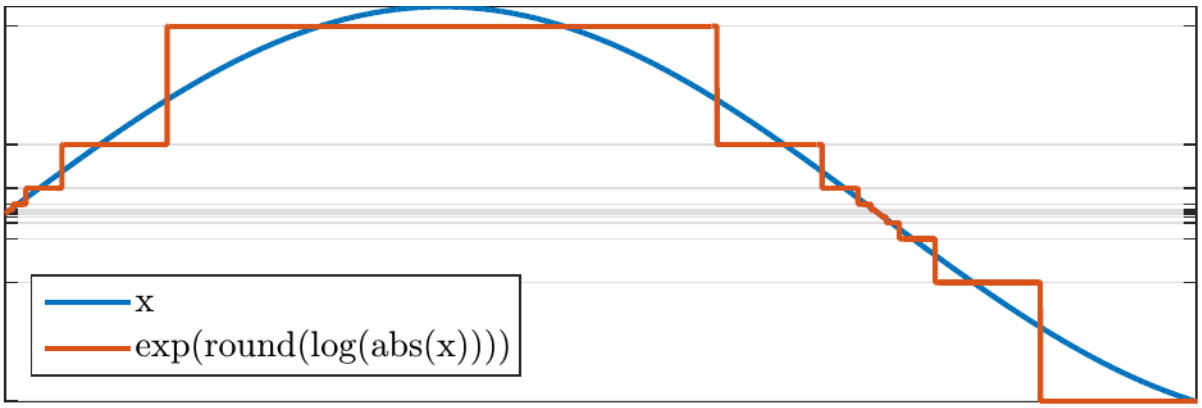
To retain equal accuracy for loud and weak signals, we *could* quantize on an logarithmic scale as $\hat{x} = \text{sign}(x) \cdot \exp\left[\Delta q \cdot \text{round}\left(\log\left(\|x\|\right)/\Delta q\right)\right]$ Such operations which limit the detrimental effects of limited range are known as *companding* algorithms.

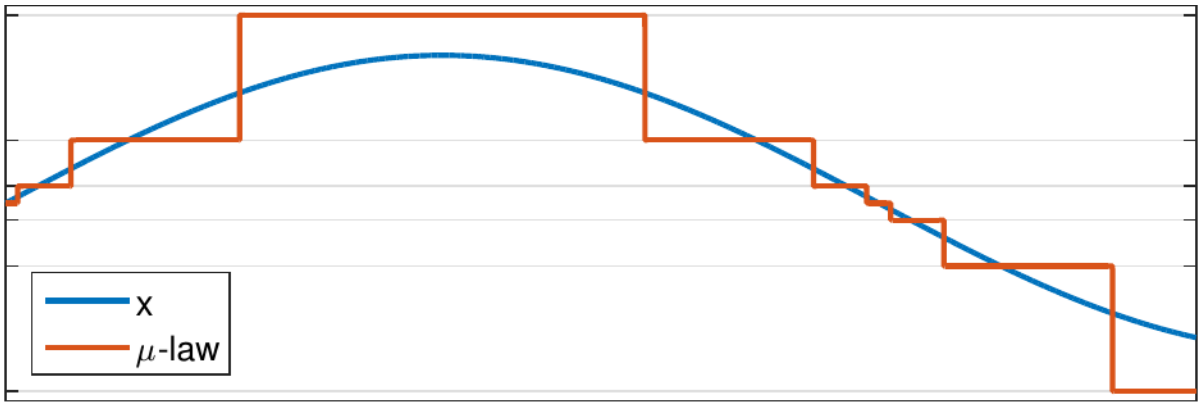
Here the intermediate representation is $y = \text{round}\left(\log\left(\|x\|\right)/\Delta q\right)$ which can be reconstructed by $\hat{x} = \text{sign}(x) \cdot \exp\left[\Delta q \cdot \|y\|\right]$. A benefit of this approach would be that we can encode signals on a much larger range and the quantization accuracy is relative to the signal magnitude. Unfortunately, very small values cause catastrophic problems. In particular, for $x = 0$, the intermediate value goes to negative infinity $y = -\infty$, which is not realizable in finite digital systems.

A practical solution to this problem is quantization with the mu-law algorithm, which defines a modified logarithm as

$$F(x) := \text{sign}(x) \cdot \frac{\log\left(1 + \mu\|x\|\right)}{(1+\mu)}.$$

By replacing the logarithm with $F(x)$, we retain the properties of the logarithm for large $x$, but avoid the problems when $x$ is small.





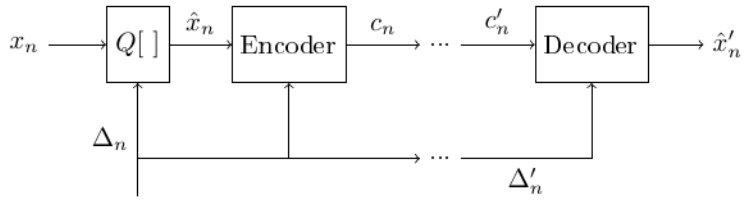

### 3.3.5 Wav-files

The most typical format for storing sound signals is the wav-file format. It is basically merely a way to store a time sequence, with typically either 16 or 32 bit accuracy, as integer, mu-law or float. Sampling rates can vary in a large range between 8 and 384 kHz. The files typically have no compression (no lossless nor lossy coding), such that recording hours of sound can require a lot of disk space. For example, an hour of mono (single channel) sound with a sampling rate of 44.1kHz requires 160 MB of disk space.

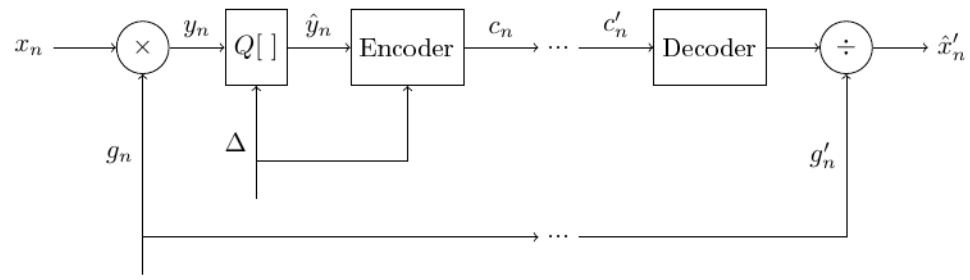
### 3.3.6 Adaptive quantization, APCM

- To obtain a uniform quantization error during single phones or sentences, the quantization error has to change slowly over time.
- In *adaptive quantization* (adaptive PCM or APCM) the quantization step size is adapted slowly such that
  - the available quantization levels cover a sufficient range such that numerical overflow can be avoided,
  - the quantization error is stable over time and
  - as long as the above constraints are fulfilled, quantization error is minimized.
- An alternative, equivalent implementation to the change in quantization step size is to apply an adaptive gain to the input signal before quantization.

APCM with adaptive quantization step size



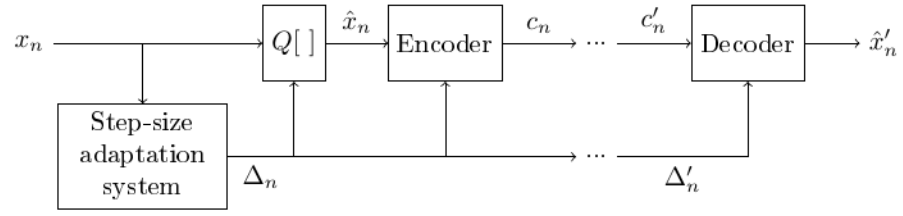
APCM with adaptive gain




### Adaptive quantization with the feed-forward algorithm using an adaptive quantization step

- The feed-forward algorithm requires that in addition to the quantized signal, also the gain-coefficients or the quantization step is transmitted to the recipient.
  - Transmitting such extra information increases the bit-rate, whereby the feed-forward algorithm is not optimal for applications which try to minimize transmission rate.

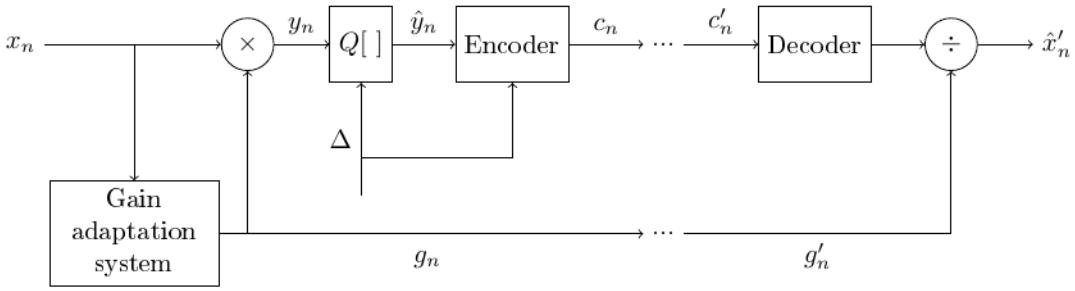
APCM with feed-forward algorithm for adaptive quantization step size




### Adaptive quantization with the feed-forward algorithm using an adaptive gain (compressor)

- In *feed-backward* algorithms the quantization step or gain-coefficient is determined from previous samples which are already quantized.
- Since the previous samples are available also at the decoder, the quantization step or gain-coefficient can be determined also at the decoder without extra transmitted information.
- If the signal grows very rapidly, this approach can however not guarantee that there are no numerical overflows, since adaptation is performed only after quantization.

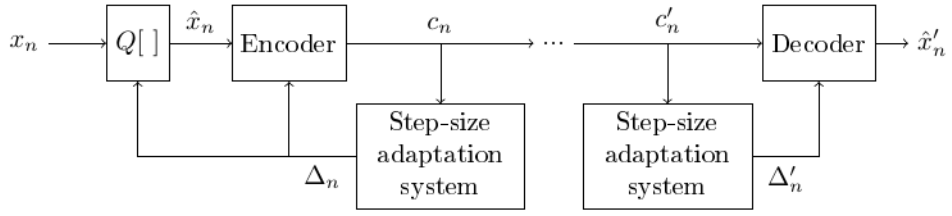
APCM with feed-forward algorithm for adaptive gain



### Adaptive quantization with the feed-backward algorithm using an adaptive quantization step

- Note that the feed-forward algorithms all require transmission of the scaling or gain coefficient, which can increase demand on bandwidth and adds to the complexity of the system.
- The parallel transmission line can be avoided by predicting those coefficients from previously transmitted elements, with a feed-backward algorithm.

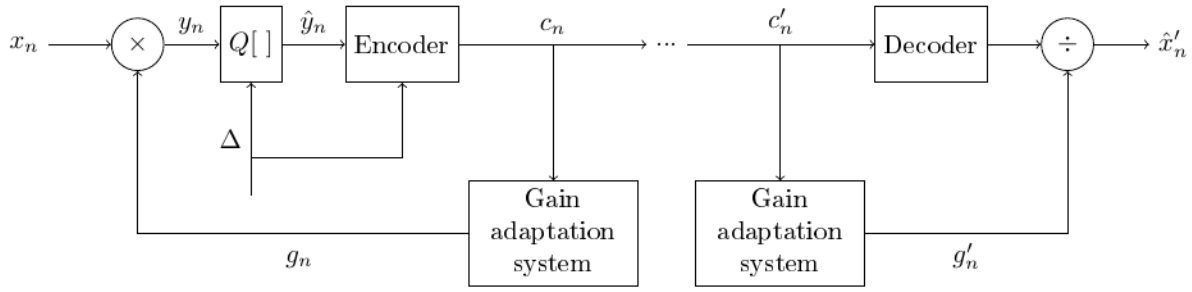
APCM with feed-backward algorithm for adaptive quantization step size




### Adaptive quantization with the feed-backward algorithm using an adaptive gain coefficient

- The feed-backward algorithm can naturally be applied on gain adaptation as well.

APCM with feed-backward algorithm for adaptive gain



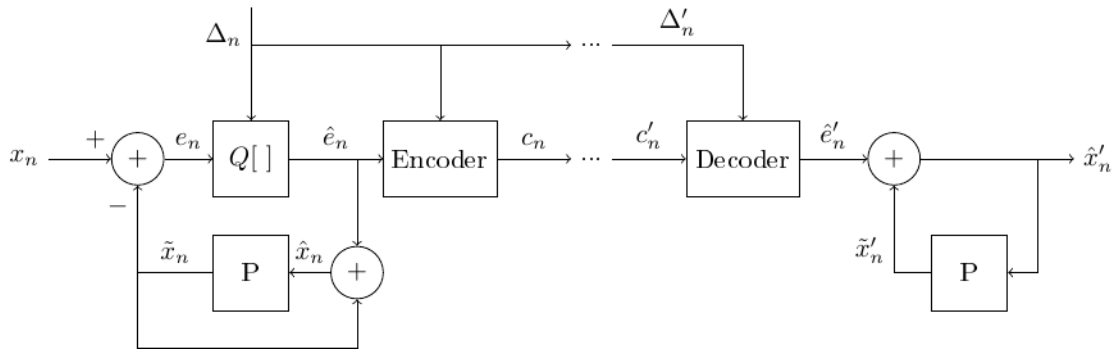

### Differential quantization DPCM

- In *differential quantization* we predict the subsequent sample, whereby we can quantize only the difference between the prediction and the actual sample.
- If the predictor is simply $\tilde{x}_k := x_{k-1}$, then the error is $e_k = x_k - \tilde{x}_k = x_k - x_{k-1}$.
- The first difference (delta modulation) is the simplest predictor, which uses the assumption that subsequent samples are highly correlated.
- The reconstruction is obtained by reorganization of terms as $x_k = e_k + x_{k-1}$.
- Observe that the reconstruction is needed at both the encoder and decoder, to feed the predictor.
- NB: At this point the flow-graphs start to get a bit complicated as there are several feedback loops.
- More generally, we can use a predictor $P$, which predicts a sample based on a weighted sum of previous samples

$$\tilde{x}_k = -\sum_{h=1}^{M} a_h x_{k-h},$$

  where the scalars $a_h$ are the predictor parameters.
- A feed-backward would here use the past quantized samples $\hat{x}_k$.

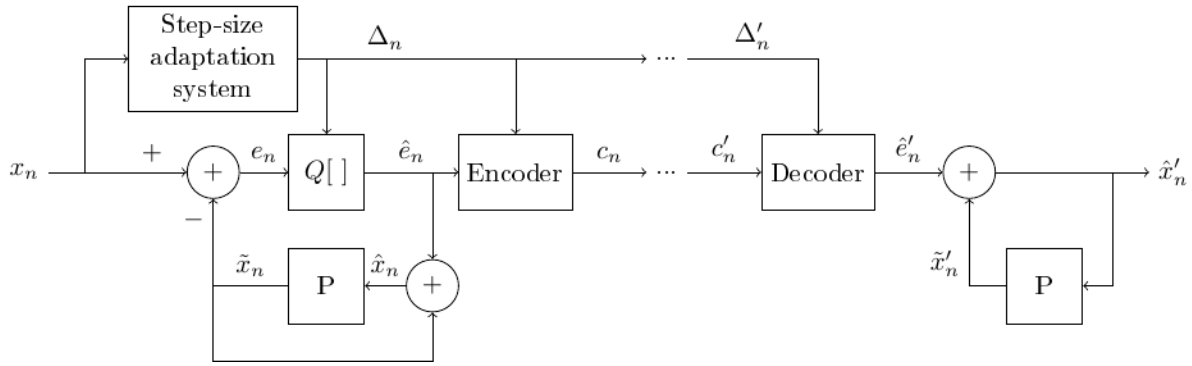
Differential PCM (DPCM)




### Adaptive and differential quantization with feed-forward

- The differential, source-model based quantization can naturally be combined with adaptive, perception-based quantization.
- The adaptive differential PCM (ADPCM) adaptively predicts the signal and adaptively choosing the quantization step.

Adaptive differential PCM (ADPCM)

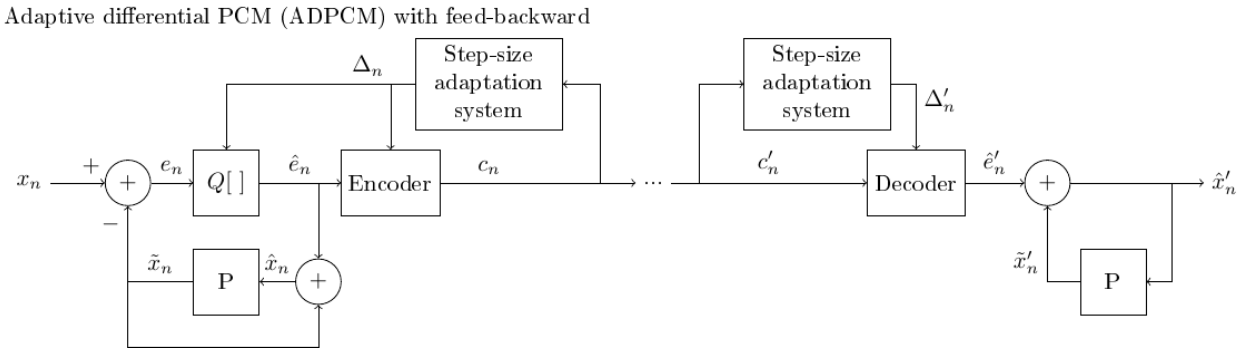



### Adaptive differential quantization with feed-backward

- The ADPCM can again, naturally, be implemented as a feed-backward algorithm as well.

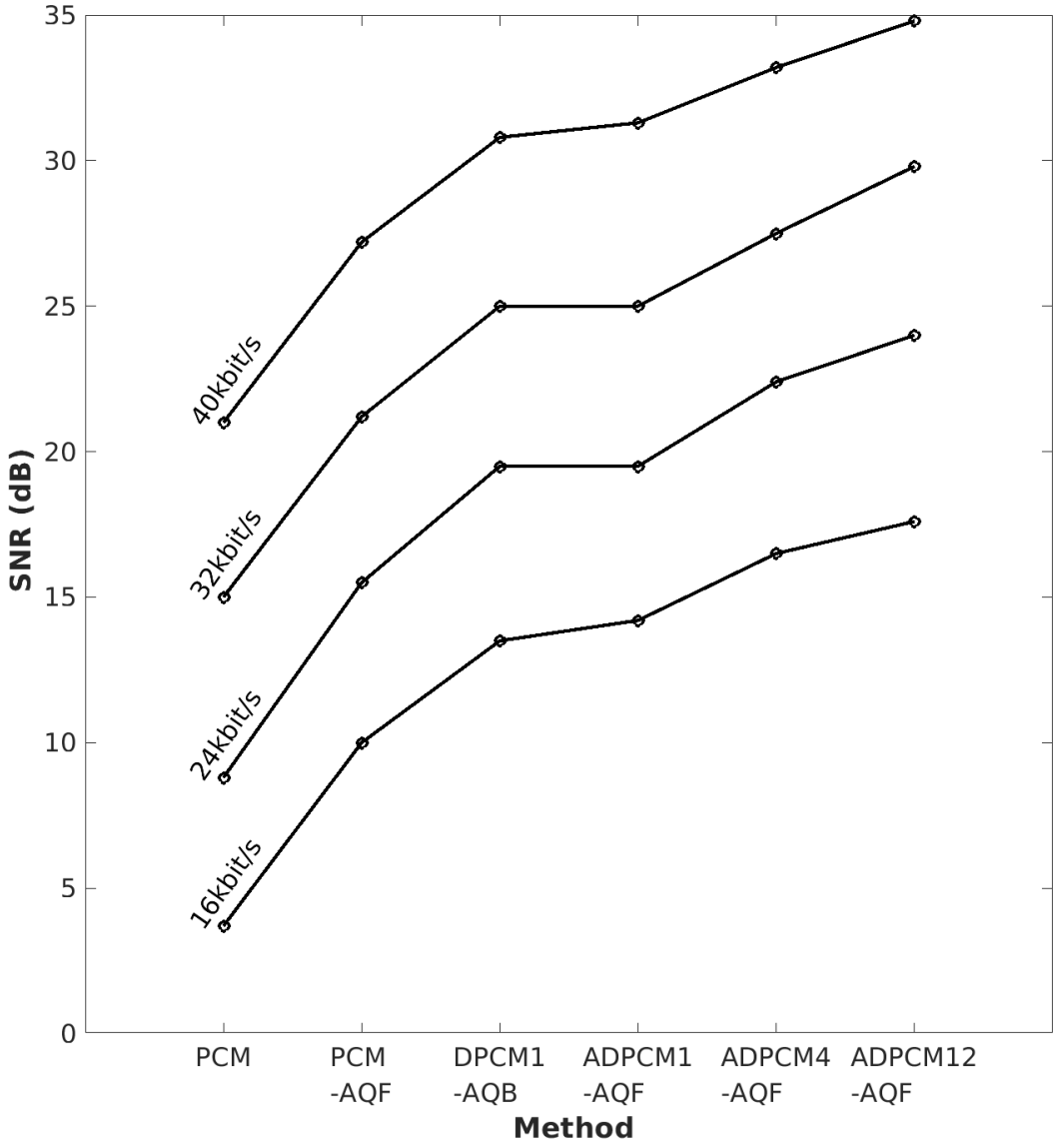
### Adaptive differential quantization w/ adaptive predictor

- A differential quantizer can be further improved by letting also the predictor be adaptive.
- The predictor learns adaptively properties of the signal.
- The flow-graph becomes complicated and is omitted here.




### Comparison of the SNR of different quantizers (not perceptual)

- The more bits we can use the better the quality (Duh!).
- The more prior information we can use about the signal the better the efficiency (SNR/bits).
  - More advanced models (can) improve quality.
  - More parameters (can) improve quality.
  - It would naturally also be entirely possible to create complicated models which do not improve quality, but simple models can go only so far.
- The linear prediction -approach can be extended into a full-blown speech production model (see separate chapter).
- Note that quality as measured by SNR does not necessarily reflect perceptual quality.

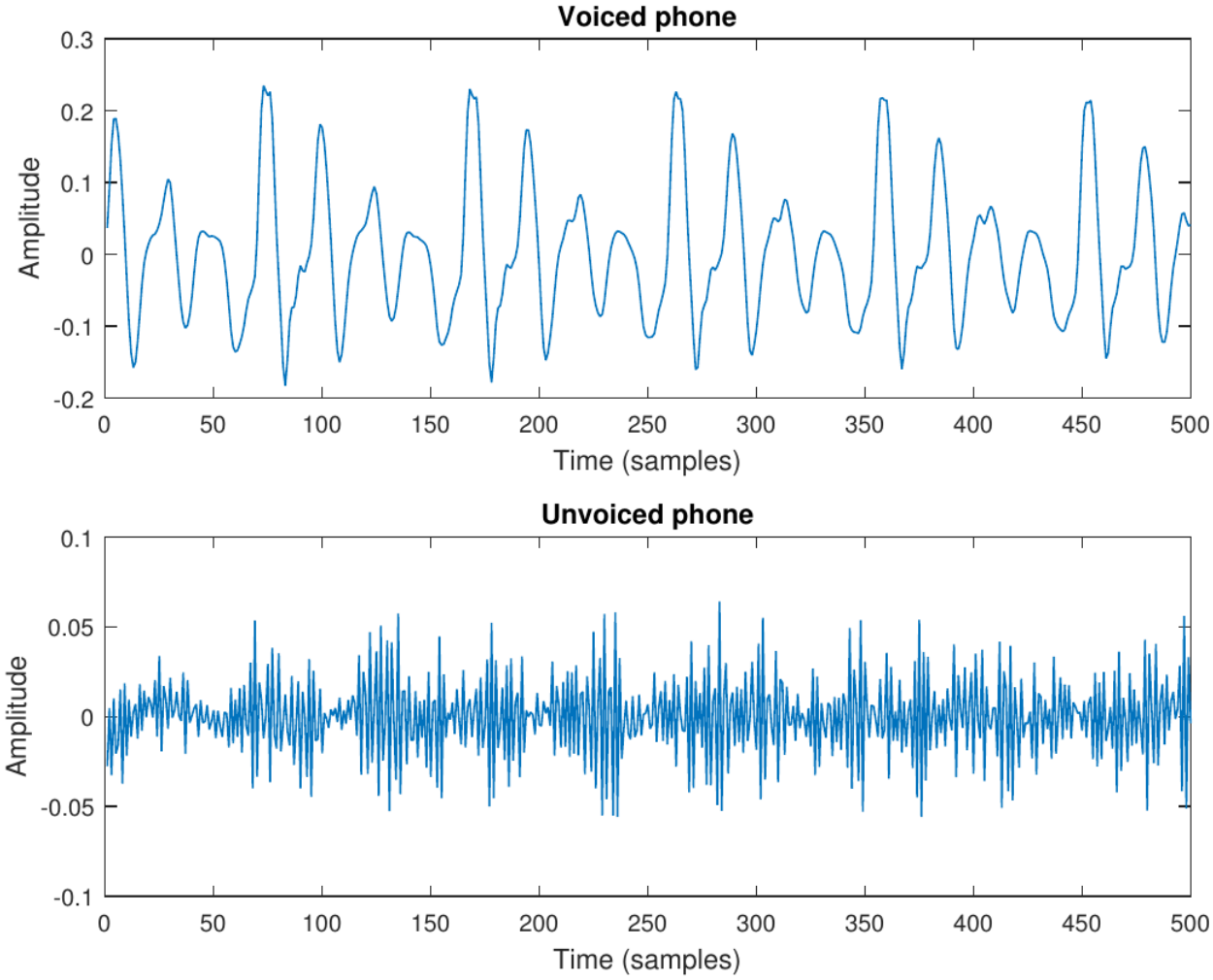



Adapted from [Noll, 1975]

### 3.3.7 Source modelling in quantization

- Adaptive quantization is based on our understanding of perception; we use the knowledge that we prefer slowly changing quantization errors.
  - This is a simple *perceptual model*.
  - Perceptual models are *quality evaluation* models.
- When we know that the signal is *speech*, we can use that to further improve quantization.
  - Models of speech signals are known as *source* models.

- At its simplest form, we can use the fact that voiced phones are fairly continuous signals = they have low-pass character = are dominated by low-frequency components.
  - Samples have a high correlation.
  - The difference between subsequent samples is much smaller than the magnitude of samples!
- The amplitude of the first difference is 41% of the original.
- Uniform quantization of the first difference thus gives a 59% reduction in the range which is approx 1 bitsample. At 44kHz that would be 44kbit/s improvement in bitrate, which is definitely noticeable.



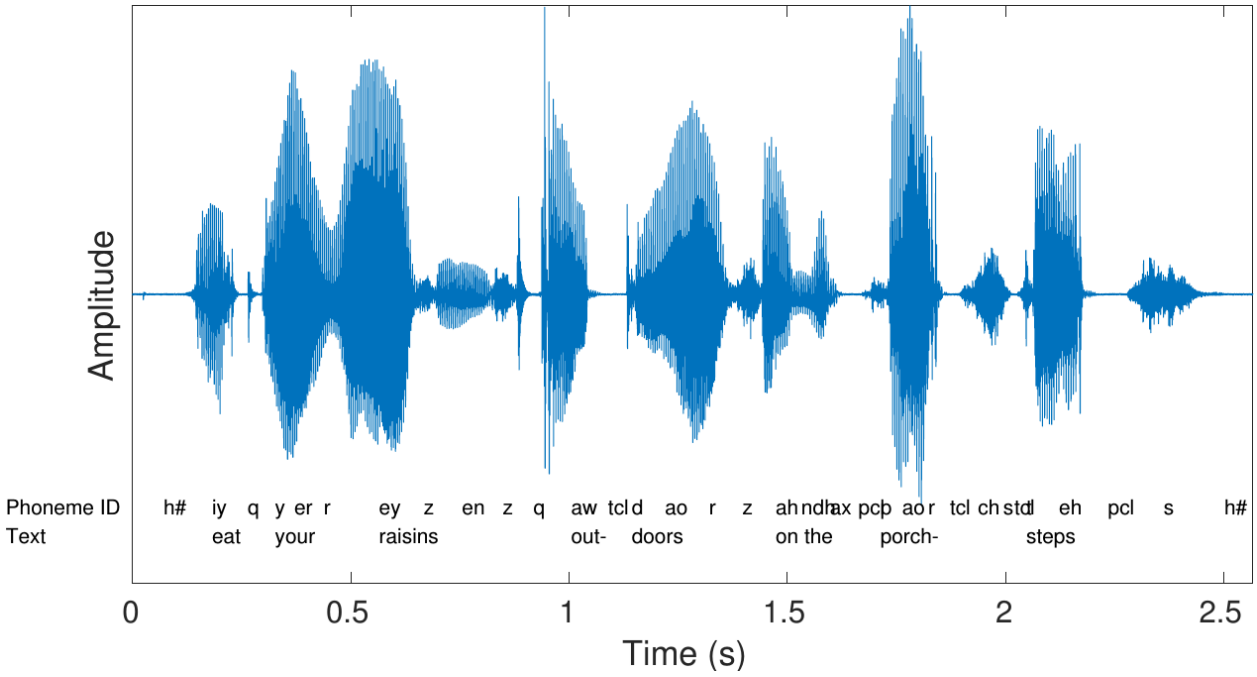

### 3.3.8 Conclusion

- Time-domain representation of speech signals is simple in floating-point processors.
  - We only need to choose a sampling rate (typically in the range 8 to 48 kHz).
- For fixed-point and lower-bitrate representations, we have several considerations and options;
  - We would like the quantization noise to be relative to the signal magnitude but stable over time for best perceptual quality.
  - We can use information about signal properties to improve efficiency with a source model.

  - If we want to reduce bit-rate, then we must make sure that the required information to decode the signal is available also at the receiving end.
- Practical processing algorithms for speech operate on a digital representation of the acoustic signal.
  - Accuracy is determined by sampling rate and quantization.
- Most common (high-quality) storage format for digital speech and audio signals is PCM (such as WAV-files).
- Some very basic analysis tools for speech signals include the autocorrelation and the zero-crossing rate.
- Many classical DSP algorithms, in their flow-grap representation, are very much alike modern machine learning methods.

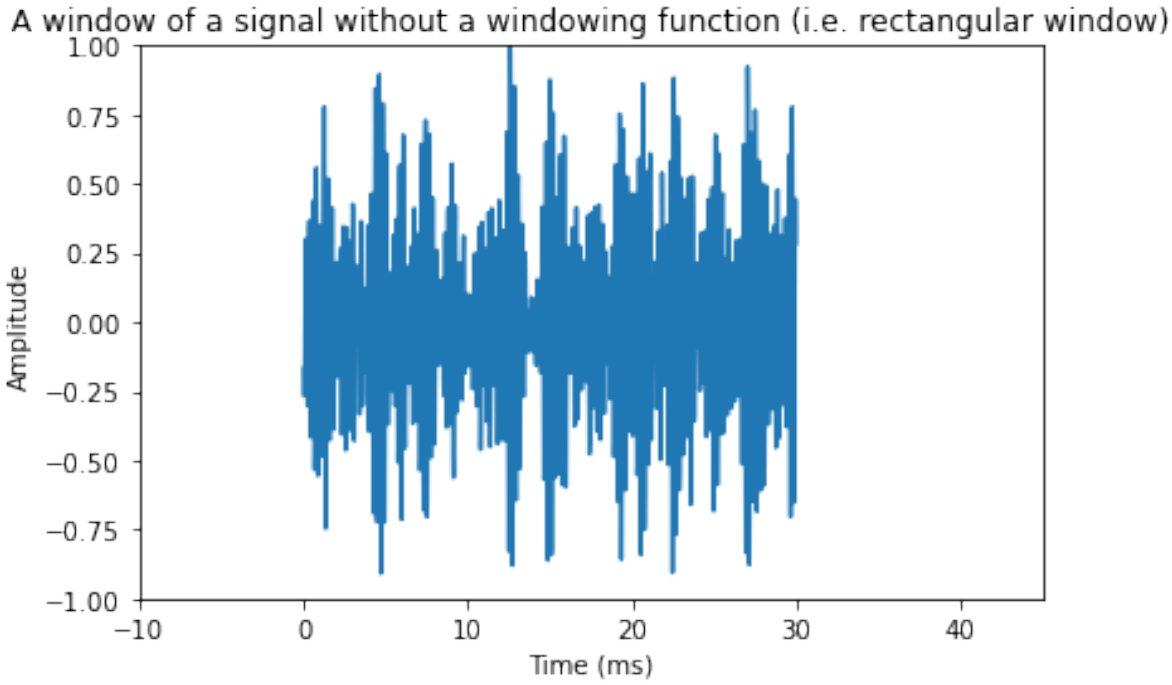
### 3.3.9 Refrences

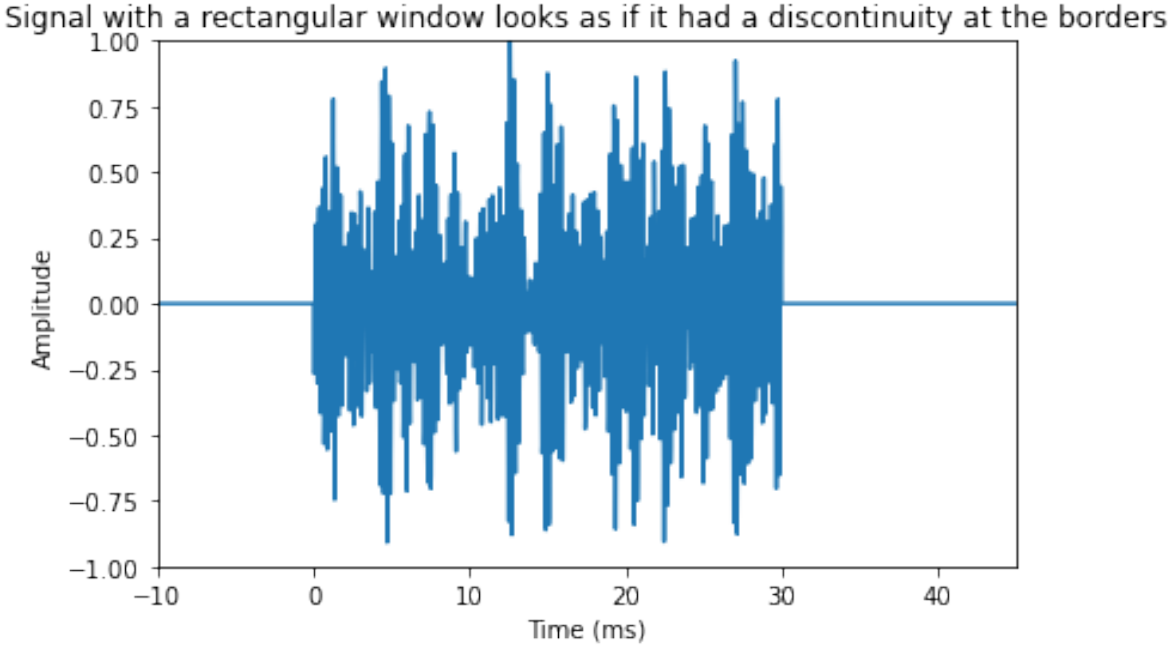
## 3.4 Windowing

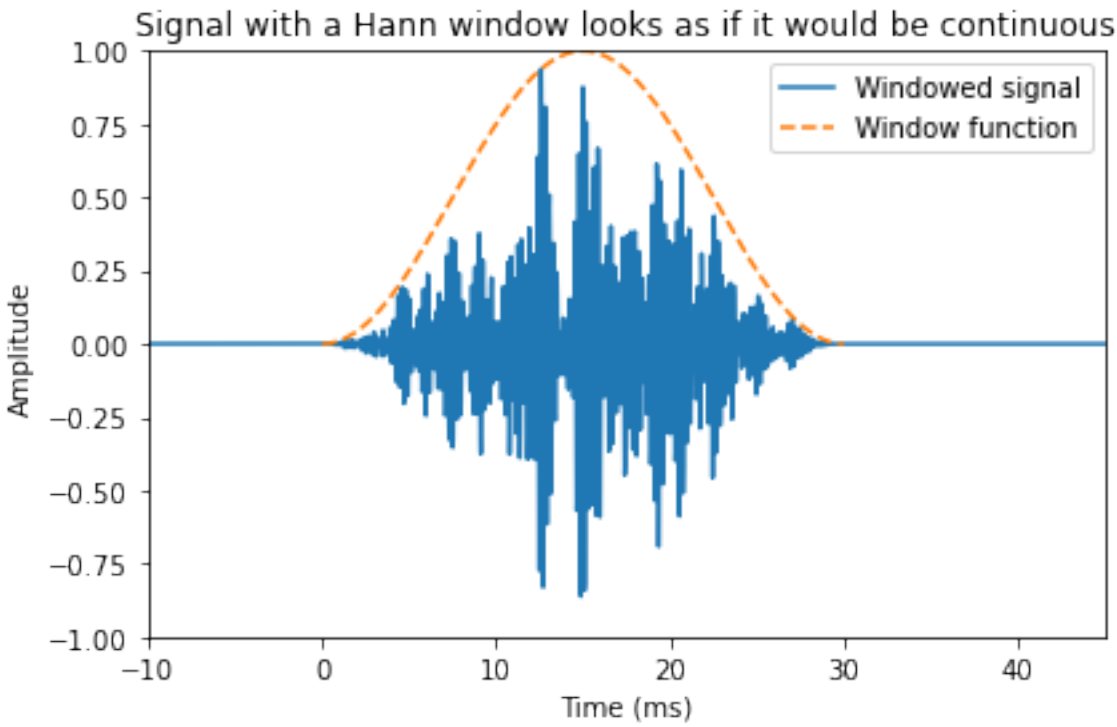



A spoken sentence is a sequence of phonemes. Speech signals are thus time-variant in character. To extract information from a signal, we must therefore split the signal into sufficiently short segments, such that, heuristically speaking, each segment contains only one phoneme. In other words, we want to extract segments which are short enough that the properties of the speech signal does not have time change within that segment.

*Windowing* is a classical method in signal processing and it refers to splitting the input signal into temporal segments. The borders of segments are then visible as discontinuities, which are incongruent with the real-world signal. To reduce the impact of segmenting on the statistical properties of the signal, we apply windowing to the temporal segments. Windowing functions are smooth functions which go to zero at the borders. By multiplying the input signal with a window function, the windowing function also goes to zero at the border such that the discontinuity at the border becomes invisible. Windowing does thus change the signal, but the change is designed such that its effect on signal statistics is minimized.


A window of a signal without a windowing function (i.e. rectangular window)
Amplitude
Time (ms)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10
0
10
20
30
40


Signal with a rectangular window looks as if it had a discontinuity at the borders
Amplitude
Time (ms)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10
0
10
20
30
40




### 3.4.1 Quick reference

There are two distinct applications of windowing with different requirements; 1) analysis and 2) processing. In analysis, we only care about extracting information as accurately as possible given computational constraints, while in processing applications, we in addition need the ability to recreate the signal from a sequence of windows.

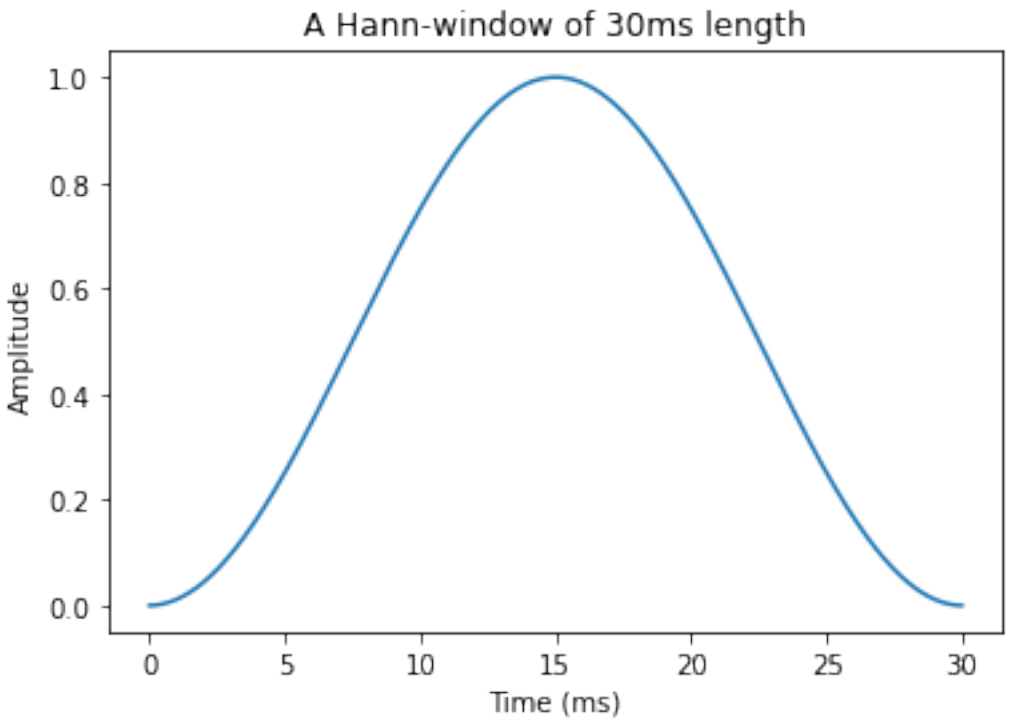
#### Windowing for *analysis* applications

This is a classical signal processing topic covered by any basic book on signal processing. Here we therefore present only the very basics. Given an input signal $x_k$, defined for all $k$, and a windowing function $w_k$, defined on a limited range $k \in [0, L)$ we can extract a window of the signal as

$$x_{k,n} = x_{n-k} w_n.$$

A classical windowing function, the Hann-window $w_n = [\sin(\pi n/L)]^2$ is shown below.

The main optimization criteria in choosing windowing functions is spectral distortion. Namely, we would like that the windowed signal resembles the original signal as much as possible. However, since it is only a short sample, it cannot be exact. As windowing is multiplication in the time-domain (see above equation), it corresponds to convolution in the frequency domain. By looking at the spectrum of the windowing function, we can therefore determine how much spreading of peaks in the frequency will occur when we apply the windowing function.




### Windowing for *processing* applications; Overlap-add

When we intend to modify the windowed signal with some processing, the most common approach is to use a technique known as overlap-add. As seen in the figure below, in overlap-add, we extract overlapping windows of the signal, apply some processing, and reconstruct by windowing a second time and then adding overlapping segments together.

An obvious requirement would then be that if the signal is not modified, that we could then reconstruct the original signal perfectly; known as the *perfect reconstruction* property. It is straightforward to demonstrate that perfect reconstruction is achieved if overlapping regions of the windowing function add up to unity. Note that here we need to take into account the windowing is applied twice. That is, we obtain perfect reconstruction if (Princen-Bradley criteria)

$$w_n^2 + w_{n+L/2}^2 = 1. \qquad \text{for} \qquad k \in [0, L/2).$$

Here the squares follow from the fact that windows are applied twice. Note that subsequent windows are then at a distance of half $L/2$ the length of the window.

A classical windowing function which follows the perfect reconstruction criteria is the half-sine window, which is actually the square root of the Hann-window. However, we have to here take special care that indices are defined correctly, such that the half-sine is defined as $w_n = \sin\left(\pi(n+0.5)/L\right)$. Observe that the difference to the Hann-window is thus the absence of a square. It the follows that, after squaring, overlapping parts add up to unity.

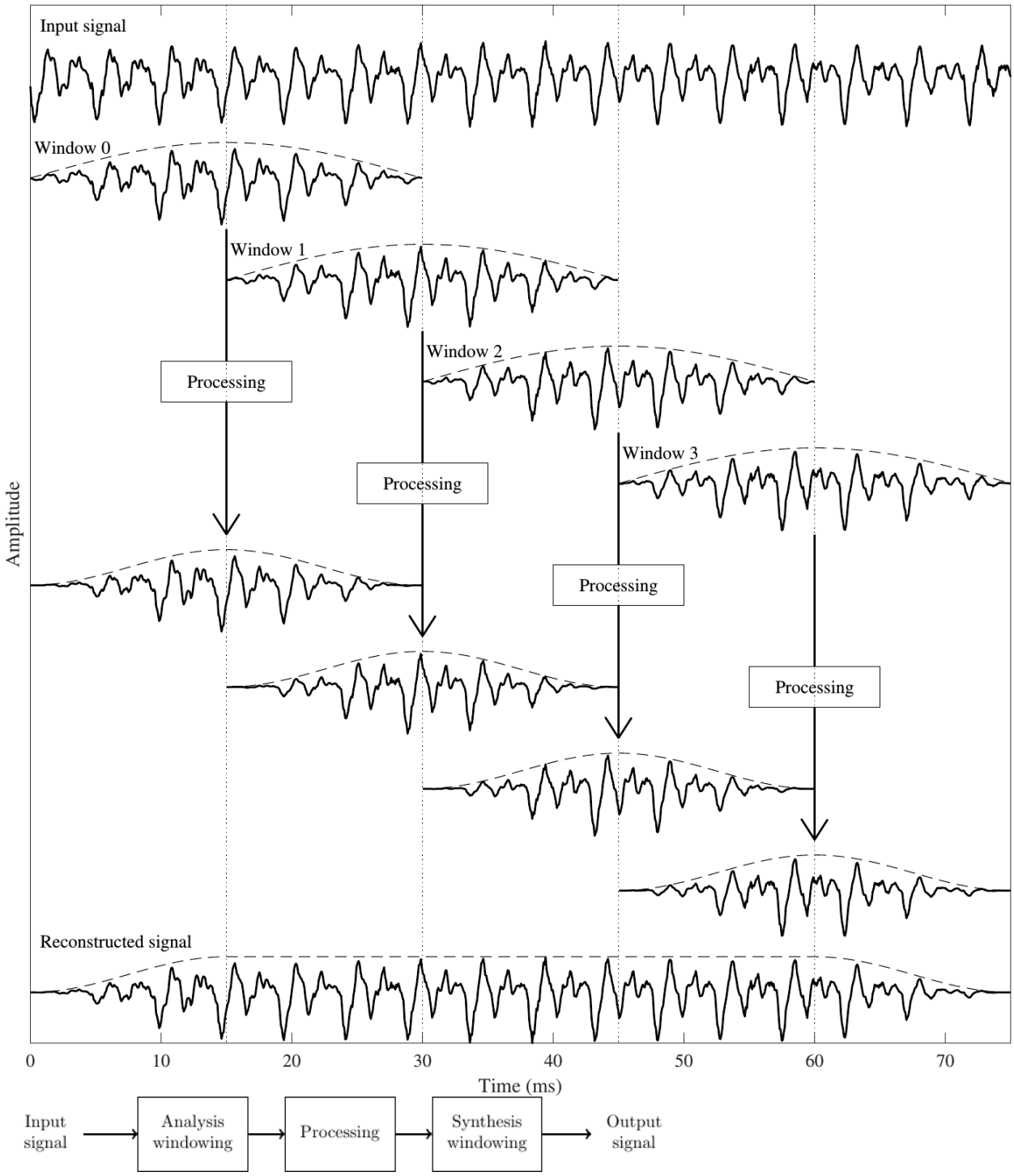
The length of windows in the figure below is 30 ms, while the shift between windows is 15 ms. This is known as 50% overlap and it is the most common approach, though it is possible to design low-overlap windows (useful in low-delay applications). We can then observe that analysis of the first window requires that we the signal is at least 30 ms long. *Analysis* of each additional window then requires 15 ms more signal. That is, for analysis we have

$$(Analysis\, signal\, length) = (windows - 1) \times (step) + (window\, length).$$

However, for reconstruction, we see that we have perfect reconstruction only in the segment between 15 ms and 60 ms. That is, only those overlap areas are perfectly reconstructed, where we have access to both the left and right windows.

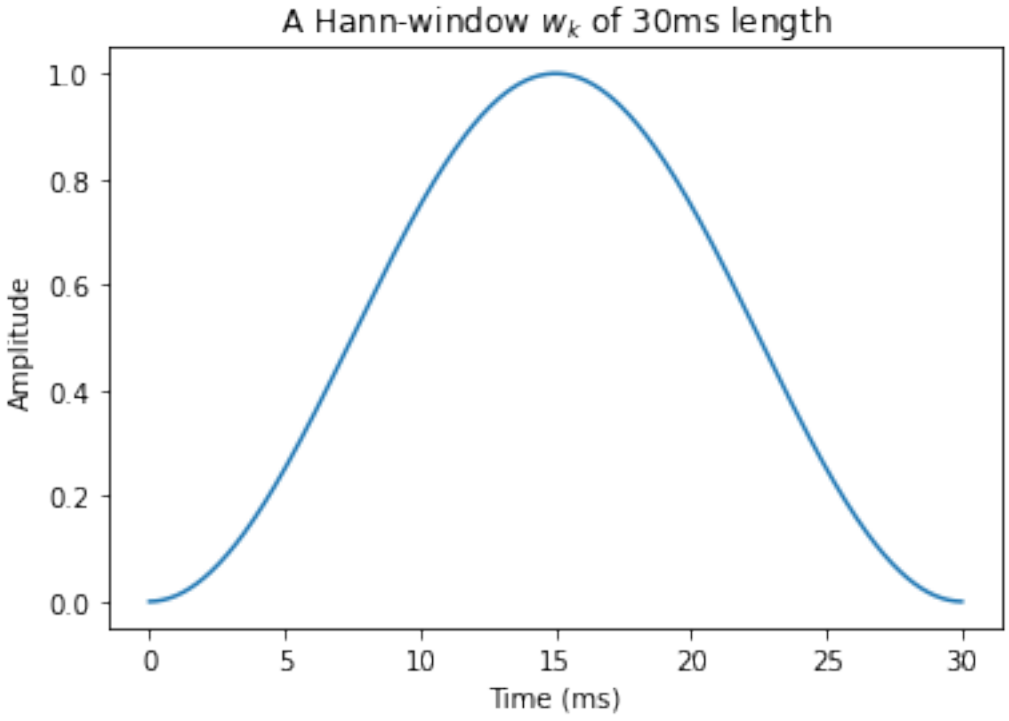
For reconstruction we then have

$$(Reconstruction\ signal\ length) = (windows - 1) \times (step).$$



### 3.4.2 Comprehensive description

Specifically, suppose $x_k$ is the $k$th sample of the input signal. Let $w_k$ be a windowing function (like the one in the figure below) such that $\$\begin{cases} w_k > 0 & k \in [0, L-1] \\ w_k = 0 & k < 0 \text{ and } k \geq L Thewindowedsignalof lengthListhenx'_k = w_k x_k.Inclassicalsignalprocessing, themaindesigncriteriaforchoosing\text{w\_k}\$$ are related to spectral resolution. Windowing causes undesirable spreading of frequency components into nearby frequencies and by choosing the windowing function, we can choose how much and how far such a components are spread.

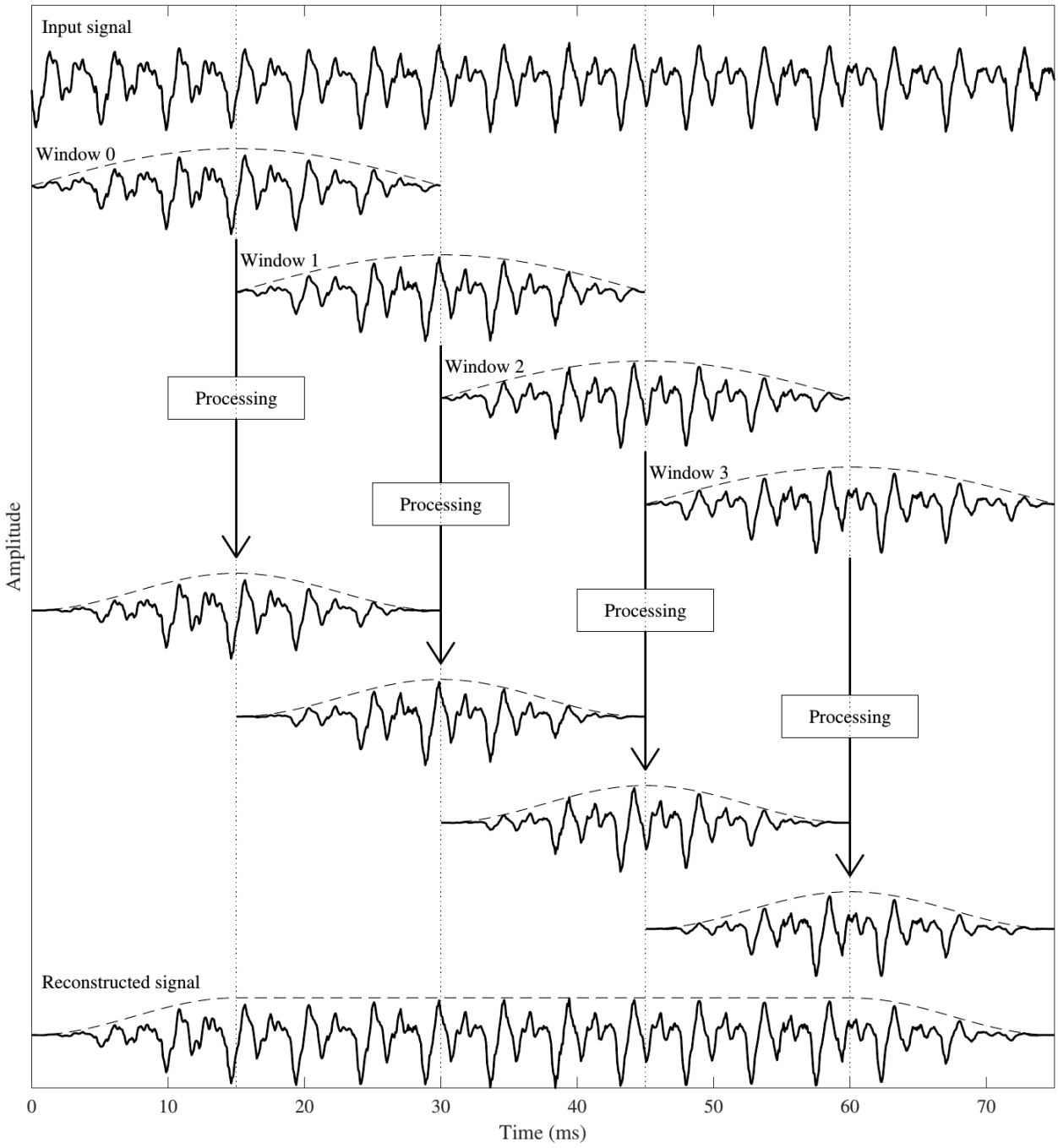



In difference to classical signal analysis, speech processing applications have a range of additional requirements. Most importantly, speech processing applications are not only analyzing the signals, but their purpose is to reconstruct the (modified) signal. The figure on the right illustrates the process. If the signal is not modified, commonly, our objective is that the signal can be perfectly reconstructed from the sequence of windows. This is known as the *perfect reconstruction* property.

In other words, a transform is said to have perfect reconstruction if the original signal can be recovered perfectly from the transformed representation.

In application using windowing, perfect reconstruction is achieved with a process known as *overlap-add* (sometimes abbreviated as OLA).

The basic principle of overlap and add is to apply windowing in overlapping segments, such that when the windows are later added together, the original signal is recovered (see Figure below).




As a first approach, let us define window $h$ as

$$x_{k,h} = w_{k-Lh/2} x_k.$$

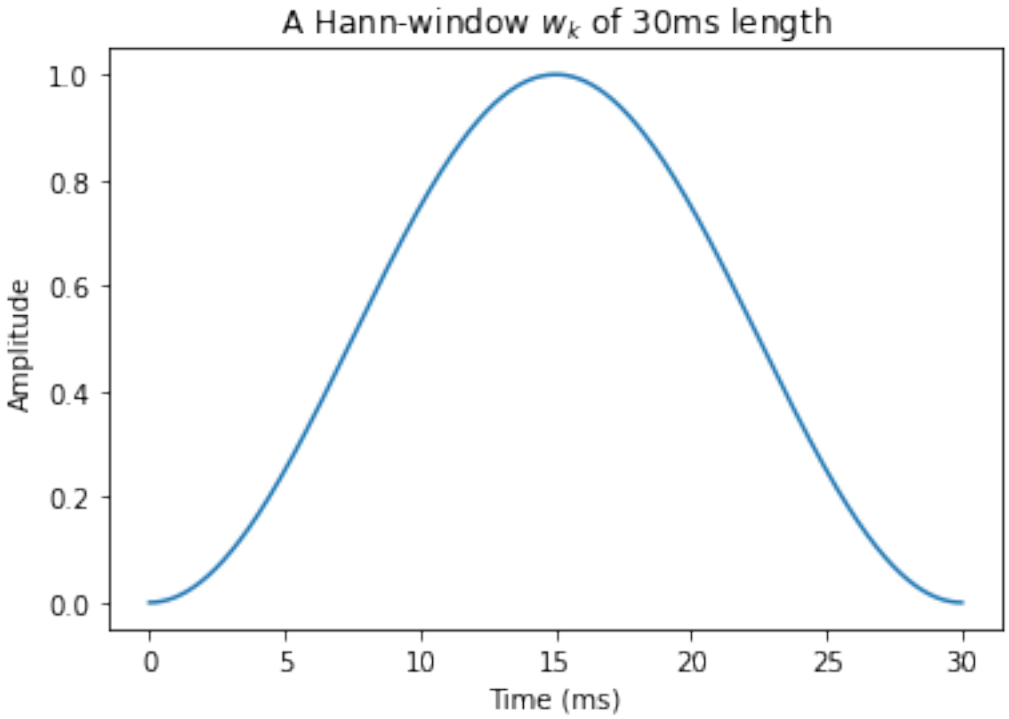
Subsequent windows $x_{k,h-1}$ and $x_{k,h}$, then have non-zero portions which are overlapping (see figure) in the region $k \in [Lh/2,\ L(h+1)/2)$ . When we add them together, we obtain

$$\begin{aligned} x_{k,h-1} + x_{k,h} &= w_{k-L(h-1)/2} x_k + w_{k-Lh/2} x_k \\ &= \left( w_{k-L(h-1)/2} + w_{k-Lh/2} \right) x_k. \end{aligned}$$

It follows that the reconstruction is exactly equal to the original $x_{k,h-1} + x_{k,h} = x_k$, iff $$w_{k+L/2} + w_k = 1, \qquad \text{for } k \in [0,\ L/2).$$ An example of a window which satisfies this requirement is the raised cosine (or Hann) window, illustrated

below and defined as

$$w_k = \frac{1}{2}\left[1 - \sin\left(\frac{2(k+0.5)\pi}{L}\right)\right] = \left[\sin\left(\frac{\pi(k+0.5)}{L}\right)\right]^2.$$




Unfortunately, when applying the above windowing in a processing application, there is a problem. Suppose the windowed signal $x_{k,h}$ is modified in some way, for example, the signal could be quantized and coded for transmission. The receiving device would then see a modified signal $\hat{x}_{k,h} = x_{k,h} + e_{k,h}$ , where $e_k, h$ is the modification applied to window $h$ and $e_{k,h}$ is non-zero only for $k \in [Lh/2, L(h+1)/2)$ . The reconstructed signal, for the windows $h$ and $h+1$, would then be (for $k \in [Lh/2,\ L(h+1)/2)$ )

$$\hat{x}_{k,h-1} + \hat{x}_{k,h} = x_k + e_{k,h-1} + e_{k,h}.$$

The reconstruction error is thus $e_{k,h-1} + e_{k,h}$. The problem here is that the modifications, $e_{k,h-1}$ and $e_{k,h}$, appear here without windowing. Consequently, if the modifications $e_{k,h}$ are non-zero near the window borders, the reconstruction will have discontinuities.

To avoid discontinuities for the modification parts $e_{k,h}$, we need to apply windowing also on the output signal. We therefore apply windowing at both the input and output:

- Input: $x_k$
- Analysis windowing: $x_{k,h} = w^{\text{in}}_{k-Lh/2} x_k$.
- Processing: $\hat{x}_{k,h} = x_{k,h} + e_{k,h}$.
- Synthesis windowing: $\hat{x}'_{k,h} = w^{\text{out}}_{k-Lh/2} \hat{x}_{k,h}$.
- Overlap-add for the region $k \in [Lh/2,\ L(h+1)/2) : \hat{x}'_k = \hat{x}'_{k,h-1} + \hat{x}'_{k,h}$.
- Output: $\hat{x}'_k$.

The input and output windows are further illustrated in the Figure on the right.

The output then has

$$\begin{aligned}
\hat{x}'_{k,h-1} + \hat{x}_{k,h} &= w^{\text{out}}_{k-L(h-1)/2}\hat{x}_{k,h-1} + w^{\text{out}}_{k-Lh/2}\hat{x}_{k,h} \\
&= w^{\text{out}}_{k-L(h-1)/2}(x_{k,h-1} + e_{k,h-1}) + w^{\text{out}}_{k-Lh/2}(x_{k,h} + e_{k,h}) \\
&= w^{\text{out}}_{k-L(h-1)/2}(w_{\text{in},k-L(h-1)/2}x_k + e_{k,h-1}) \\
&\quad + w^{\text{out}}_{k-Lh/2}(w_{\text{in},k-Lh/2}x_k + e_{k,h}) \\
&= \left(w^{\text{out}}_{k-L(h-1)/2}w^{\text{in}}_{k-L(h-1)/2} + w^{\text{out}}_{k-Lh/2}w^{\text{in}}_{k-Lh/2}\right)x_k \\
&\quad + w^{\text{out}}_{k-L(h-1)/2}e_{k,h-1} + w^{\text{out}}_{k-Lh/2}e_{k,h}.
\end{aligned}$$

We immediately observe that all output errors $e_{k,h}$ have been multiplied with windowing functions, whereby discontinuities are avoided. Moreover, perfect reconstruction is achieved iff

$$w^{\text{out}}_{k+L/2}w^{\text{in}}_{k+L/2} + w^{\text{out}}_{k}w^{\text{in}}_{k} = 1, \qquad \text{for } k \in [0,\, L/2).$$

This leaves us with the design task of two windowing functions, $w^{\text{in}}_k$ and $w^{\text{out}}_k$ .

To choose the output window, we can assume that the modifications to the signal $e_{k,h}$ are uncorrelated white noise of zero mean and variance $\sigma^2$ . The output error energy is then (for $k \in [0,\, L/2)$ )

$$\begin{aligned}
&E\left[\left(w^{\text{out}}_{k-L(h-1)/2}e_{k,h-1} + w^{\text{out}}_{k-Lh/2}e_{k,h}\right)^2\right] \\
&= E\left[\left(w^{\text{out}}_{k-L(h-1)/2}e_{k,h-1}\right)^2\right] + E\left[\left(w^{\text{out}}_{k-Lh/2}e_{k,h}\right)^2\right] \\
&= \left[\left(w^{\text{out}}_{k-L(h-1)/2}\right)^2 + \left(w^{\text{out}}_{k-Lh/2}\right)^2\right]\sigma^2.
\end{aligned}$$

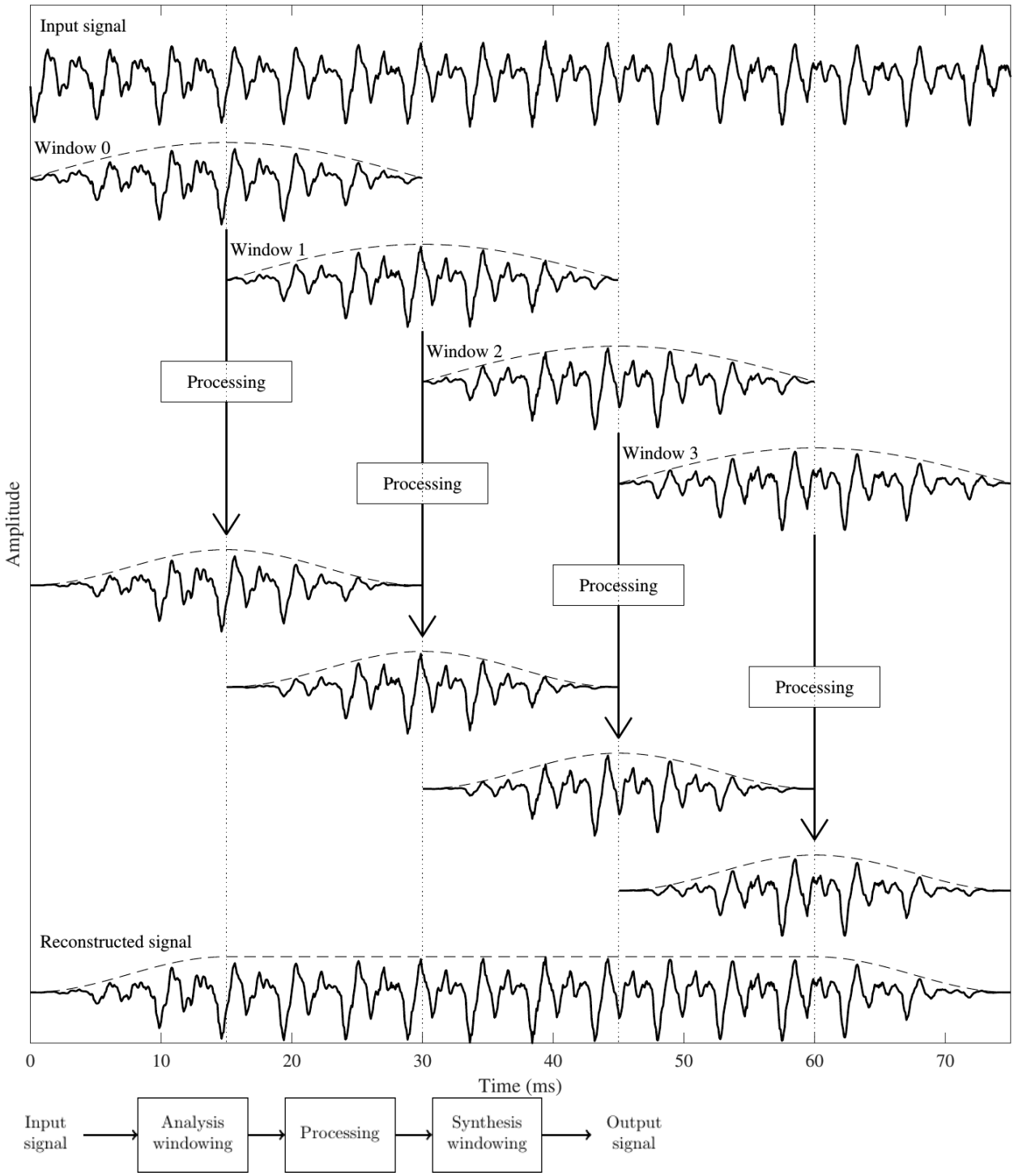
Modulations in signal energy are perceptually undesirable, whereby we can require that

$$\left(w^{\text{out}}_{k+L/2}\right)^2 + \left(w^{\text{out}}_{k}\right)^2 = 1, \qquad \text{for } k \in [0,\, L/2).$$

To simultaneously satisfy constraints on both input and output windows, we set $w_k = w^{\text{in}}_k = w^{\text{out}}_k$, such that our only criteria is

$$\boxed{w^2_{k+L/2} + w^2_k = 1, \qquad \text{for } k \in [0,\, L/2).}$$

This is known as the Princen-Bradley condition for overlapping windows.



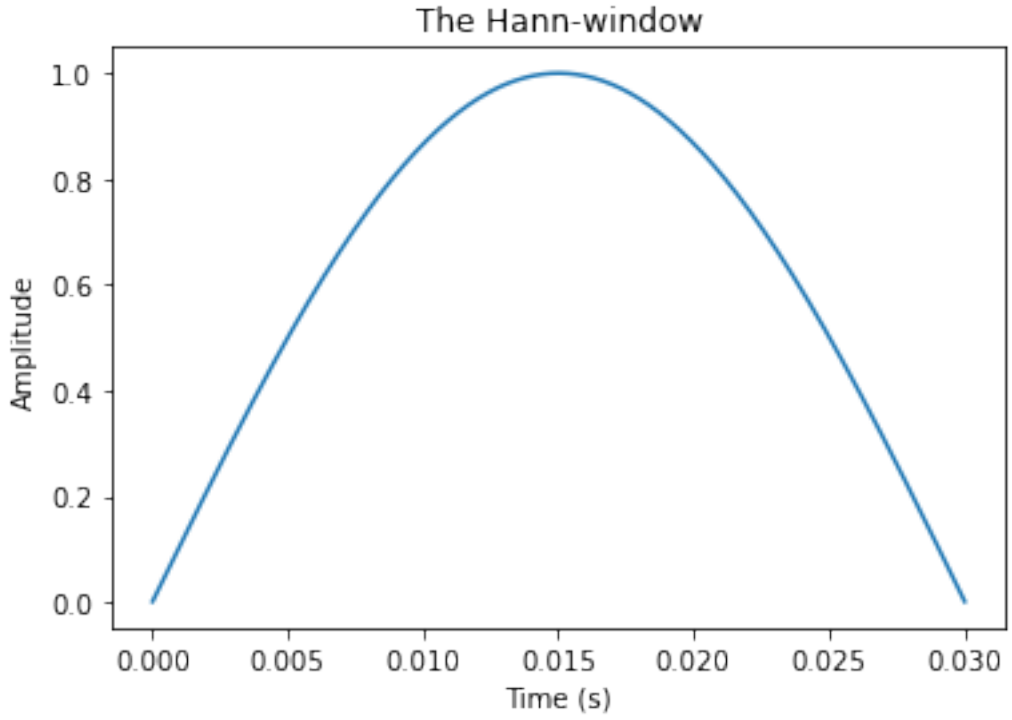

Several windowing functions which satisfy the above criteria are known. In fact, from any window which satisfies the reconstruction criteria, we can obtain a window which satisfies the Princen-Bradley condition by taking the square root. For example, we have the half-sine window

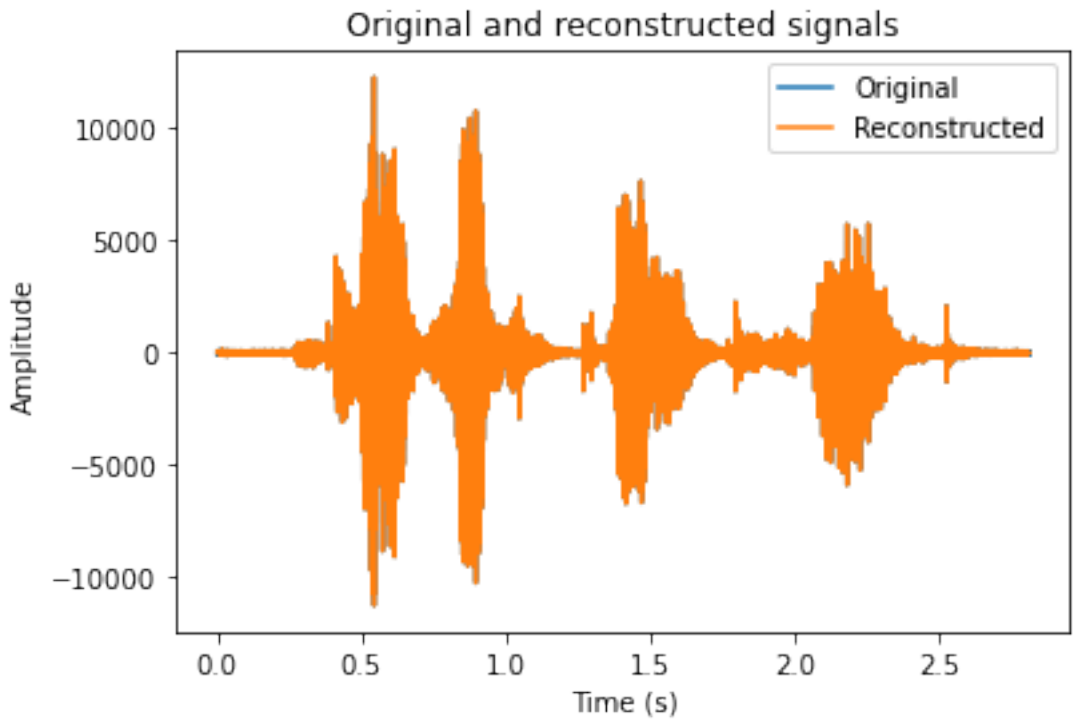
$$w_k = \begin{cases} \sin\left(\frac{(k+0.5)\pi}{L}\right), & \text{for } 0 \leq k < L \\ 0, & \text{otherwise} \end{cases}$$

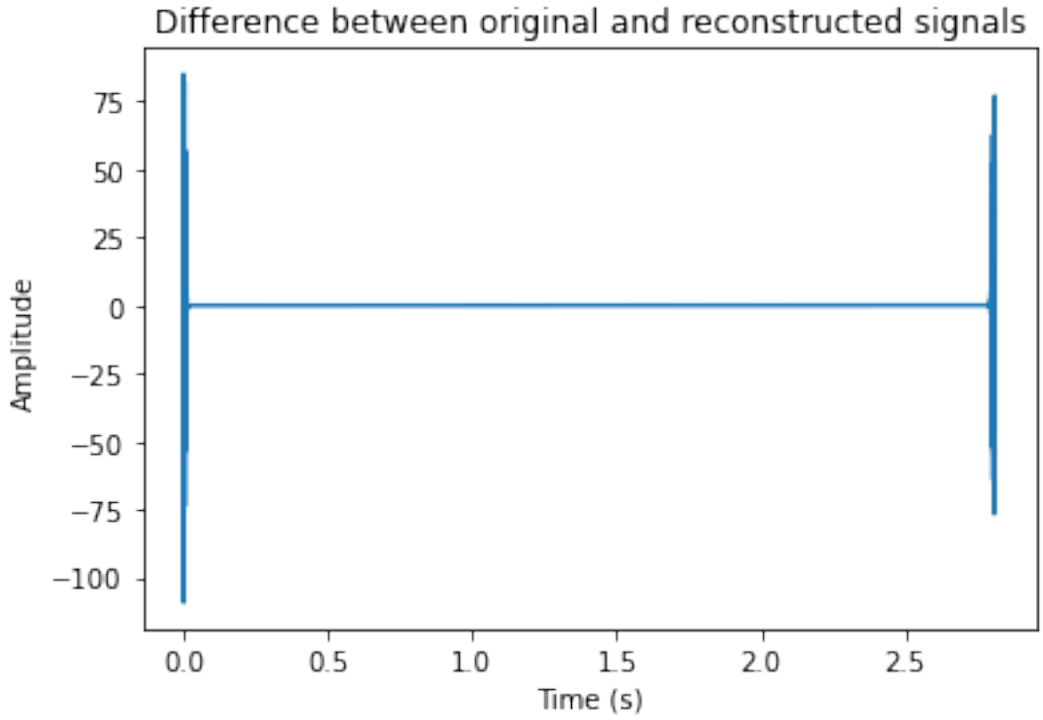
and the Kaiser-Bessel-derived (KBD) window

$$w_k = \begin{cases} \gamma \sqrt{\sum_{h=0}^{k} I_0\left(\pi\alpha\sqrt{1-\left(\frac{2h}{L-1}-1\right)^2}\right)}, & \text{for } 0 \le k < L \\ 0, & \text{otherwise,} \end{cases}$$

where $I_0()$ is the zeroth order modified Bessel function of the first kind and $\gamma$ is a scalar scaling coefficient chosen such that Princen-Bradley holds.






We see that the reconstruction is identical to the original, for the exception of the borders. In this example, we started windowing from the borders, such that the first window has only the right-hand side window in the overlap-region. Since the left window is missing, reconstruction does not give perfect output. The same happens at the end of the signal. In practice this is actually desirable, because now the reconstruction does not have discontinuities at its borders. However, if this reconstruction would be combined with other signals, then we would have to stitch their borders together.

#### CELP windowing

Another type of windowing which supports perfect reconstruction is applied in speech codecs using the code-excited linear prediction (CELP) paradigm. Here, temporal statistics of the signal are modeled with a predictive (IIR) filter and the filter residual is windowed with square windows. In practice, this approach works only with a computationally complex analysis-by-synthesis methodology, and it has not received much attention outside the speech coding community.

#### Conclusion

Windowing with overlap-add is a basic and commonly used tool in speech processing. It allows algorithms to modify sections of the signal such that the modifications do not cause discontinuities to the signal. A properly designed windowing for overlap-add does not in itself cause distortions and the original signal can be perfectly reconstructed from the windows. The only notable disadvantage of overlapping windowing is that overlaps cause redundancy, since information which appears in an overlap region between windows $k$ and $k+1$, will then always be included in computations in both window $k$ and $k+1$. Overlap-add processing can be modified to remove the redundancy, by projecting the overlap area into two orthogonal subspaces. Such methods are known as lapped transforms and are however beyond the scope of the current treatise.

In general, still, I would advise using perfect reconstruction methods in all speech processing applications (except coding applications where lapped transforms are preferred). The system is then deterministic in the sense that all modifications are due to the main processing algorithm and windowing will never cause surprising side-effects.

## 3.5 Signal energy, loudness and decibel

### 3.5.1 Signal energy

By signal energy, we usually mean the variance of the signal, which is the average squared deviation from the mean $Energy(x) = var(x) = E[(x-\mu)^2]$, where $\mu = E[x]$ is the average of the signal $x$. The variance is a measure of energy if we interpret $x$ as the displacement of a pendulum.

Since the amplitude of an oscillating signal varies through the period of the oscillation, it does not usually make sense to estimate the instantaneous energy, but only averaged over some *window*. Observe however that the windowing function reduces the average energy (it multiplies the signal by a quantity smaller than unity), which introduces a bias that should be corrected if an estimate of the absolute energy is required. Usually, however, the bias is consistent throughout a dataset and can be ignored.

Typical alternative energy estimates:

- Energy can be calculated over *spectral bands*, often called energy bands, that is, a range of frequencies of a *time-frequency transform*, such as 0 to 1000 Hz, 1000 Hz to 2000 Hz and so one for 1 kHz bands. Observe that the bands should be wide enough that they have a "large" number of frequency components within them such that the variance can be estimated. Such a representation is equivalent with an spectral envelope model of the signal.
- In a time-frequency representation, the energy of a single frequency component can be estimated over time. That is, we can take the average energy of a frequency component over several subsequent frames or windows.

Observe that since both of these representations are calculated from windowed signals, they will be similarly biased as the window itself.

### 3.5.2 Decibel

A commonly used unit for signal energy is decibel (dB). The formula to convert a signal energy value $\sigma^2$ to decibels is

$$10 \log_{10} \sigma^2.$$

Decibel is thus a logarithmic measure of energy. Trivially, we can convert this formula also to $20 \log_{10} \sigma$ such that it relates signal *magnitude* (the standard deviation) $\sigma$ to the decibel scale.

The benefit of using decibels is that signal energies have often a very large range. By taking the logarithm, we obtain a representation where for example visualizations are much easier to handle (see illustration on the right). Moreover, decibels are also closer to human perception of acoustic energy.

#### Energy normalisation, loudness, dBFS and dBov

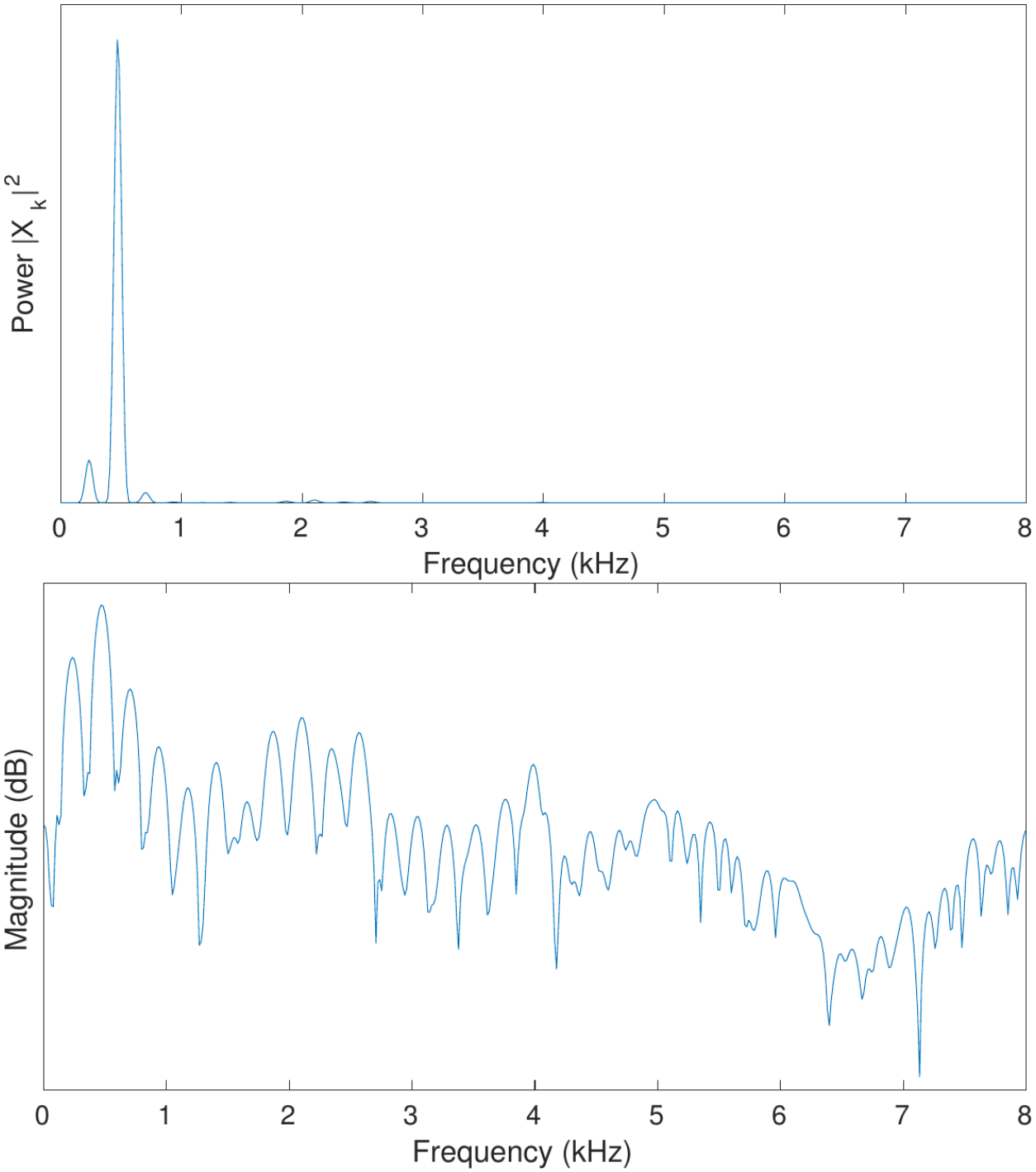
In practically all uses of acoustic data, we need to normalize the sounds such that they have approximately the same volume or at least a known volume. For example, consider a television program and advertisements. Most would feel that it is very annoying if the advertisements are much louder than the main program (see also loudness wars). We thus need to normalize the advertisement to match volume of the main program. Normalizing the average energy of the advertisement to match that of the main program is one crude way of doing that. Observe however that perception of energy is different across frequency ranges such that energy and the perceived loudness are not the same thing. To measure loudness we therefore need to model subjective perception. This is an involved subject and not discussed further here. Practical applications however still need some normalization to avoid fundamental problems such as clipping.

The energy measures decibel to overload, *dBov* and decibel to full-scale, *dBFS,* are related to the dynamic range of a signal storage or transmission format. Suppose for example that the maximum amplitude that a digital representation in which a signal is represented is $x_{ov}$. If we would try to represent a larger amplitude than that, then the signal would be clipped (distorted). dBov is a measure of how much below the maximum amplitude (how much below clipping) a signal is. Suppose $P_0$ is the energy of the maximum-amplitude square wave. Then the dBov of a signal with energy $P$ is defined as

$$L_{\text{ov}} = 10 \log_{10} \left( \frac{P}{P_0} \right).$$

Since the energy of a sinusoid with maximum amplitude is $\sqrt{\frac{1}{2}}$ of the maximum-amplitude square wave, then its dBov is $-3.01$. Observe that dBov values are always negative. dBFS is a similar

In typical cases, input speech signals are normalized to $-26$ dBov such that moderate processing of the signal is unlikely to cause clipping.




The energy (power) of a speech signal spectrum (above) and its logarithm on the decibel scale (lower).

## 3.6 Spectrogram and the STFT

We have intuitive notion of what a high or low pitch means. Pitch refers to our perception of the frequency of a tonal sound. The Fourier *spectrum* of a signal reveals such frequency content. This makes the spectrum an intuitively pleasing domain to work in, because we can visually examine signals.

In practice, we work with discrete-time signals, such that the corresponding time-frequency transform is the discrete Fourier transform. It maps a length $N$ signal $x_n$ into a complex valued frequency domain representation $X_k$ of $N$ coefficients as

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{kn}{N}}.$$

For real-valued inputs, positive and negative frequency components are complex conjugates of each other, such that we retain $N$ unique units of information. However, since spectra are complex-valued vectors, it is difficult to visualize them as such. A first solution would be to plot the magnitude spectrum $|X_k|$ or power spectrum $|X_k|^2$. Due to large differences in the range of different frequencies, unfortunately these representations do not easily show relevant information.

The log-spectrum $20 \log_{10} \|X_k\|$ is the most common visualization of spectra and it gives the spectrum in decibels. It is useful because again, it gives a visualization where sounds can be easily interpreted.

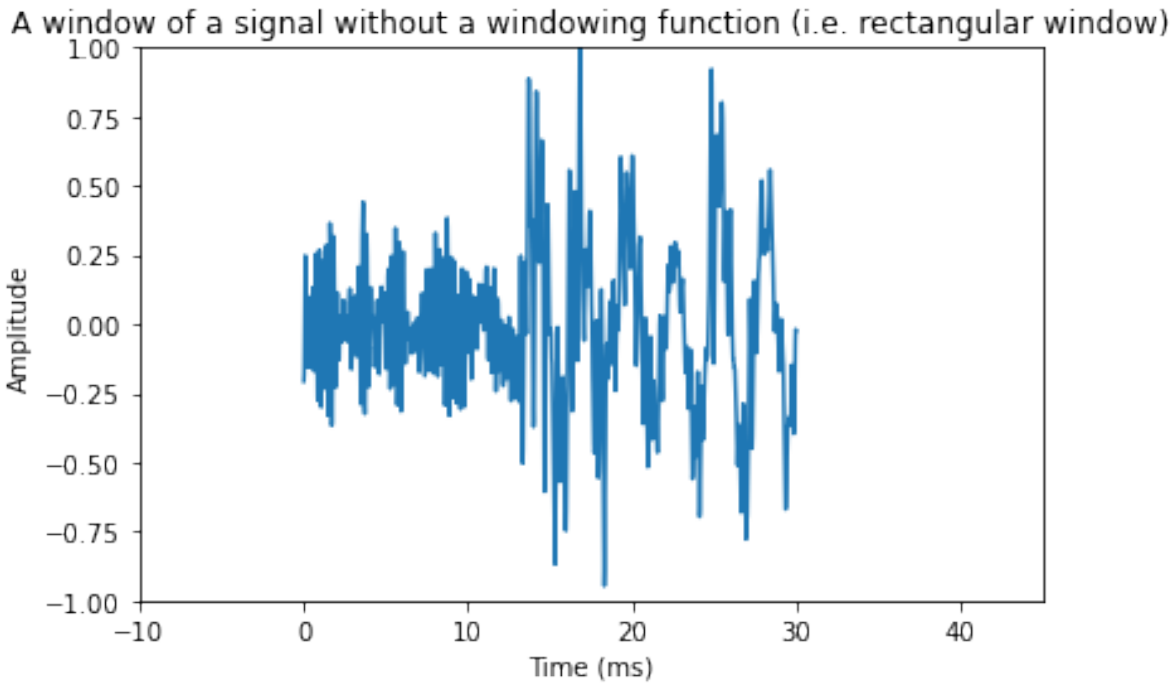
Since Fourier transforms are covered by basic works in signal processing, we will here assume that readers are familiar with the basic properties of these transforms.

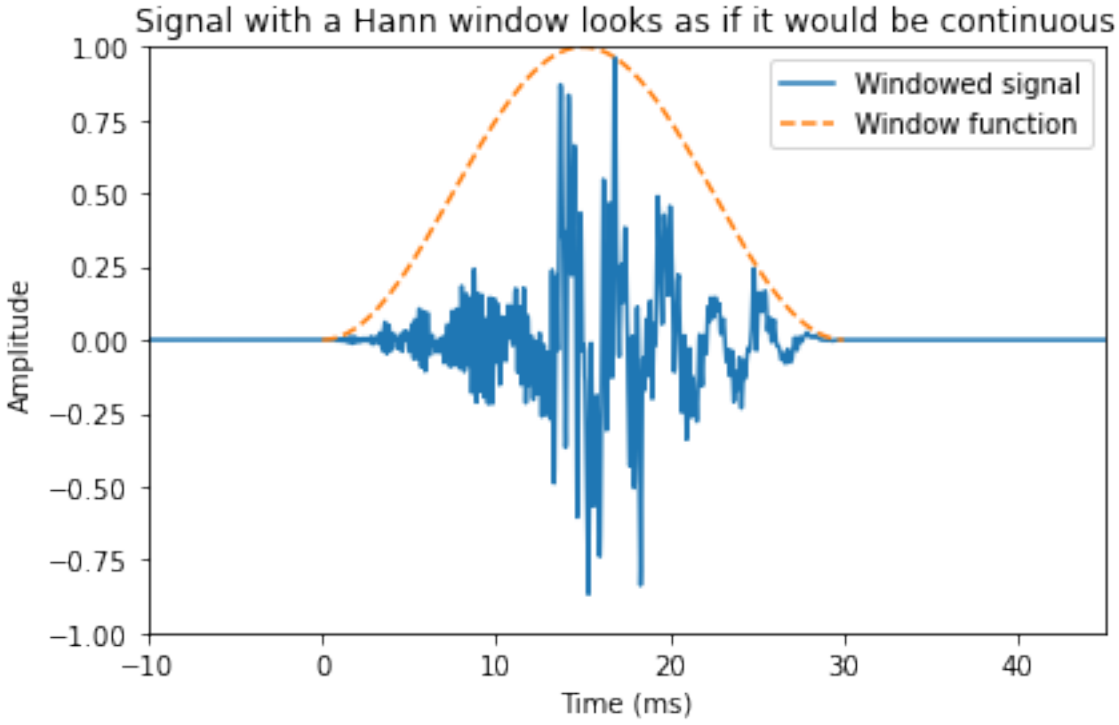
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
import scipy


# read from storage
filename = 'sounds/test.wav'
fs, data = wavfile.read(filename)

window_length_ms = 30
window_length = int(np.round(fs*window_length_ms/1000))

n = np.linspace(0.5,window_length-0.5,num=window_length)

# windowing function
windowing_fn = np.sin(np.pi*n/window_length)**2 # sine-window


windowpos = np.random.randint(int((len(data)-window_length)))

datawin = data[windowpos:(windowpos+window_length)]
datawin = datawin/np.max(np.abs(datawin)) # normalize

spectrum = scipy.fft.rfft(datawin*windowing_fn)
f = np.linspace(0.,fs/2,num=len(spectrum))


plt.plot(n*1000/fs,datawin)
plt.xlabel('Time (ms)')
plt.ylabel('Amplitude')
plt.title('A window of a signal without a windowing function (i.e. rectangular window)
 ↪')
```

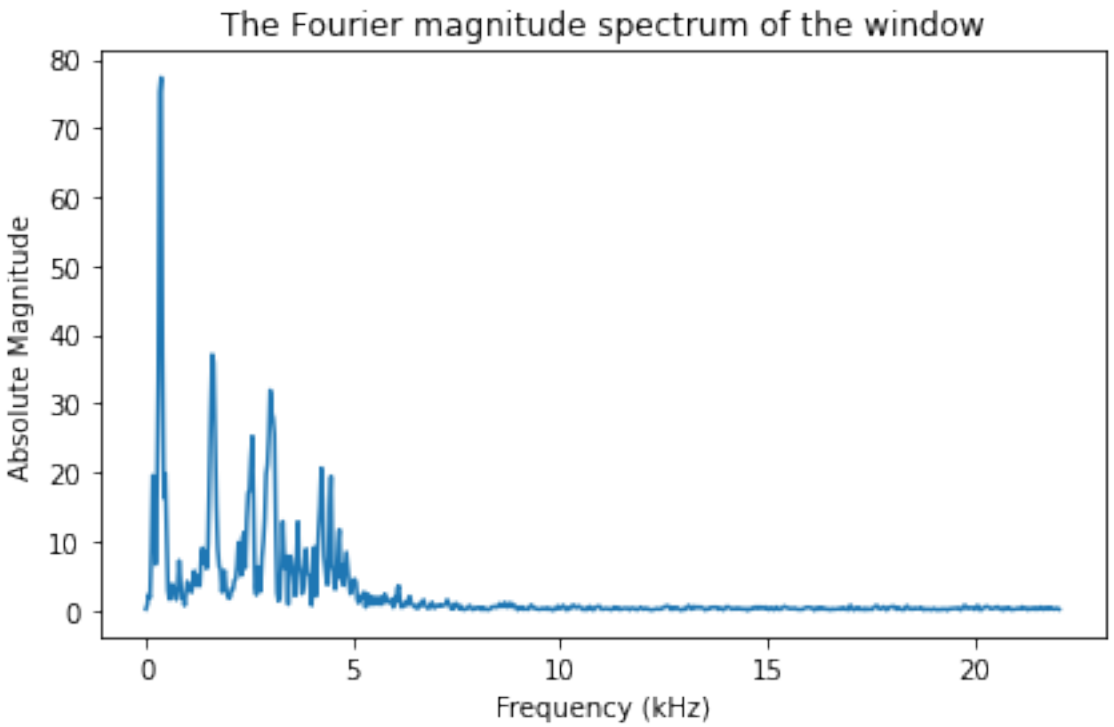
(continues on next page)

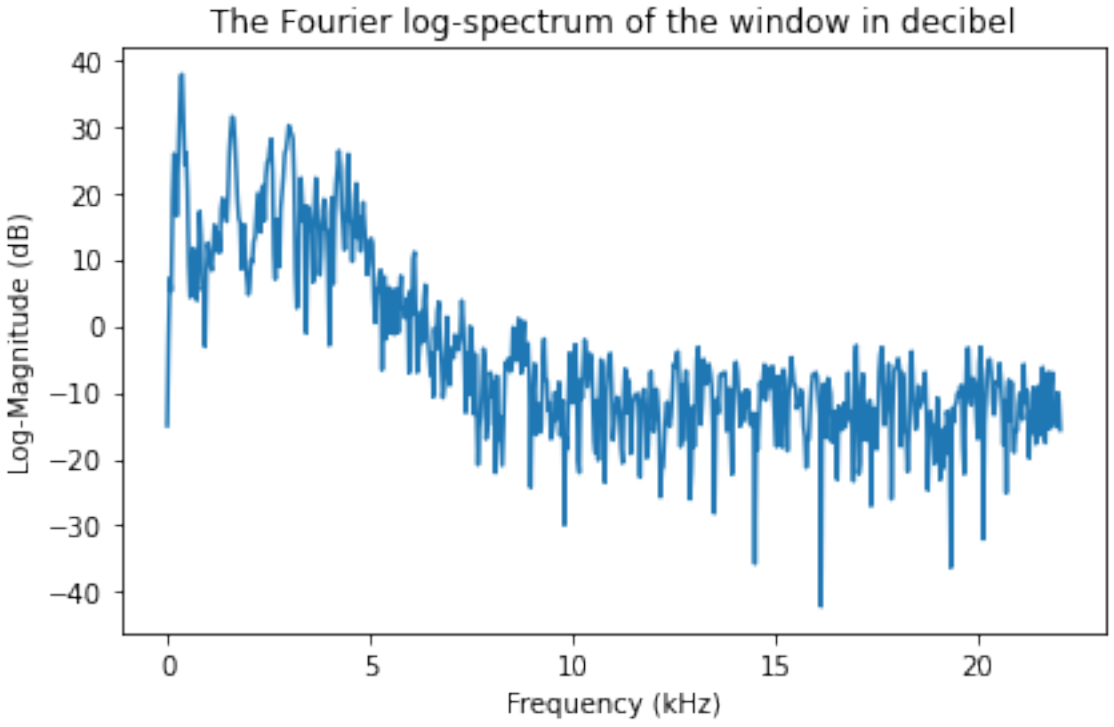
(continued from previous page)

```
plt.axis([-10.,45.,-1.,1.])
plt.tight_layout()
plt.show()

#nx = np.concatenate(([-1000,0.],n,[window_length,window_length+1000]))
#datax = np.concatenate(([0.,0.],datawin,[0.,0.]))
#plt.plot(nx*1000/fs,datax)
#plt.xlabel('Time (ms)')
#plt.ylabel('Amplitude')
#plt.title('Signal with a rectangular window looks as if it had a discontinuity at␣
 ↪the borders')
#plt.axis([-10.,45.,-1.,1.])
#plt.tight_layout()
#plt.show()

nx = np.concatenate(([-1000,0.],n,[window_length,window_length+1000]))
datax = np.concatenate(([0.,0.],datawin*windowing_fn,[0.,0.]))
plt.plot(nx*1000/fs,datax,label='Windowed signal')
plt.plot(n*1000/fs,windowing_fn,'--',label='Window function')
plt.legend()
plt.xlabel('Time (ms)')
plt.ylabel('Amplitude')
plt.title('Signal with a Hann window looks as if it would be continuous')
plt.axis([-10.,45.,-1.,1.])
plt.tight_layout()
plt.show()

plt.plot(f/1000,np.abs(spectrum))
plt.xlabel('Frequency (kHz)')
plt.ylabel('Absolute Magnitude')
plt.title('The Fourier magnitude spectrum of the window')
plt.tight_layout()
plt.show()

plt.plot(f/1000,20.*np.log10(np.abs(spectrum)))
plt.xlabel('Frequency (kHz)')
plt.ylabel('Log-Magnitude (dB)')
plt.title('The Fourier log-spectrum of the window in decibel')
plt.tight_layout()
plt.show()
```


A window of a signal without a windowing function (i.e. rectangular window)
Amplitude
Time (ms)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10
0
10
20
30
40


Signal with a Hann window looks as if it would be continuous
Windowed signal
Window function
Amplitude
Time (ms)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10
0
10
20
30
40




```
interactive(children=(FloatSlider(value=1.4048072562358276, description='position_s
↪', layout=Layout(width='760…
```

Speech signals are however non-stationary signals. If we transform a spoken sentence to the frequency domain, we obtain a spectrum which is an average of all phonemes in the sentence, whereas often we would like to see the spectrum of each

individual phoneme separately.

By splitting the signal into shorter segments, we can focus on signal properties at a particular point in time. Such segmentation was already discussed in the windowing section.

By windowing and taking the discrete Fourier transform (DFT) of each window, we obtain the short-time Fourier transform (STFT) of the signal. Specifically, for an input signal $x_n$ and window $w_n$, the transform is defined as

$$STFT\{x_n\}(h,k) = X(h,k) = \sum_{n=0}^{N-1} x_{n+h} w_n e^{-i2\pi \frac{kn}{N}}.$$

The STFT is one of the most frequently used tools in speech analysis and processing. It describes the evolution of frequency components over time. Like the spectrum itself, one of the benefits of STFTs is that its parameters have a physical and intuitive interpretation.

A further parallel with a spectrum is that the output of the STFT is complex-valued, though where the spectrum is a vector, the STFT output is a matrix. As a consequence, we cannot directly visualize the complex-valued output. Instead, STFTs are usually visualized using their log-spectra, $20\log_{10}(X(h,k))$. Such 2 dimensional log-spectra can then be visualized with a heat-map known as a spectrogram.

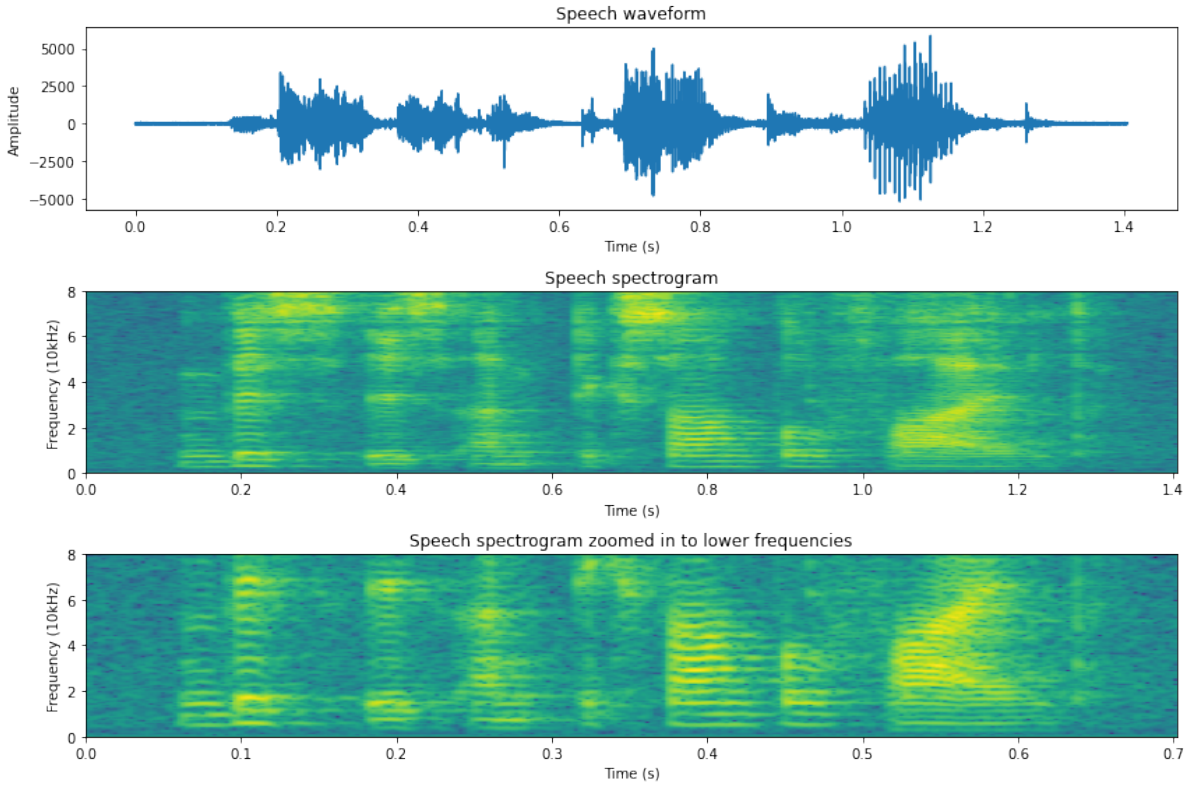
```
import numpy as np
from scipy.io import wavfile
import IPython.display as ipd
from scipy import signal

# read from storage
filename = 'sounds/test.wav'
fs, data = wavfile.read(filename)

# resample to 16kHz for better visualization
data = scipy.signal.resample(data, len(data)*8000//fs)
fs = 16000

ipd.display(ipd.Audio(data,rate=fs))

# window parameters in milliseconds
window_length_ms = 30
window_step_ms = 5

window_step = int(np.round(fs*window_step_ms/1000))
window_length = int(np.round(fs*window_length_ms/1000))
window_count = int(np.floor((data.shape[0]-window_length)/window_step)+1)

# windowing function
windowing_fn = np.sin(np.pi*np.linspace(0.5,window_length-0.5,num=window_length)/
 ↪window_length)**2 # Hann-window

spectrogram_matrix = np.zeros([window_length,window_count],dtype=complex)
for window_ix in range(window_count):
    data_window = np.multiply(windowing_fn,data[window_ix*window_step+np.
 ↪arange(window_length)])
    spectrogram_matrix[:,window_ix] = np.fft.fft(data_window)

fft_length = int((window_length+1)/2)

t = np.arange(0.,len(data),1.)/fs
plt.figure(figsize=(12,8))
plt.subplot(311)
```

(continues on next page)

(continued from previous page)

```
plt.plot(t,data)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Speech waveform')
plt.subplot(312)
plt.imshow(20*np.log10(0.2+np.abs(spectrogram_matrix[range(fft_length),:])),origin=
 ↪'lower',aspect='auto',extent=[0.,len(data)/fs,0.,fs/2000])
#plt.xticks(np.arange(0.,len(data)/fs,0.5));
plt.xlabel('Time (s)')
#plt.yticks(np.arange(0,fft_length,fft_length*10000/fs));
plt.ylabel('Frequency (10kHz)');
plt.title('Speech spectrogram')
plt.subplot(313)
plt.imshow(20*np.log10(0.2+np.abs(spectrogram_matrix[range(fft_length//2),:])),origin=
 ↪'lower',aspect='auto',extent=[0.,.5*len(data)/fs,0.,fs/2000])
plt.xlabel('Time (s)')
#plt.yticks(np.arange(0,fft_length,fft_length*10000/fs));
plt.ylabel('Frequency (10kHz)');
plt.title('Speech spectrogram zoomed in to lower frequencies')
plt.tight_layout()
plt.show()
```

```
<IPython.lib.display.Audio object>
```




When looking at speech in a spectrogram, many important features of the signal can be clearly observed:

- Horizontal lines in a comb-structure correspond to the fundamental frequency.
- Vertical lines correspond to abrupt sounds, which are often characterized as transients. Typical transients in speech

are stop consonants.

- Areas which have a lot of energy in the high frequencies (appears as a lighter colour), correspond to noisy sounds like fricatives.

```
from ipywidgets import *
import IPython.display as ipd
from ipywidgets import interactive
import numpy as np
import scipy
from scipy.io import wavfile
from scipy import signal
import matplotlib.pyplot as plt
%matplotlib inline

filename = 'sounds/test.wav'
fs, data = wavfile.read(filename)
data = data.astype(np.int16)
data = data[:]
data_length = int(len(data))


def update(window_length_ms=32.0, window_step_ms=16.0):
    ipd.clear_output(wait=True)
    window_length = int(window_length_ms*fs/1000.)
    window_step = int(window_step_ms*fs/1000.)
    window_count = (data_length - window_length)//window_step + 1
    dft_length = (window_length + 1)//2

    # sine window
    window_function = np.sin(np.pi*np.arange(.5/window_length,1,1/window_length))**2

    windows = np.zeros([window_length, window_count])
    for k in range(window_count):
        ix = k*window_step + np.arange(0,window_length,1)
        windows[:,k] = data[ix]*window_function

    X = scipy.fft.rfft(windows.T,n=2*window_length)

    fig = plt.figure(figsize=(12, 4))
    ax = fig.subplots(nrows=1,ncols=1)
    ax.imshow(20.*np.log10(np.abs(X.T)),origin='lower')
    plt.axis('auto')
    plt.show()
    fig.canvas.draw()



interactive_plot = interactive(update, window_length_ms=(2.0, 300.0, 0.5), window_
↪step_ms=(1.,50.,0.01))
output = interactive_plot.children[-1]
output.layout.height = '350px'
interactive_plot
```

```
interactive(children=(FloatSlider(value=32.0, description='window_length_ms',␣
↪max=300.0, min=2.0, step=0.5), F…
```

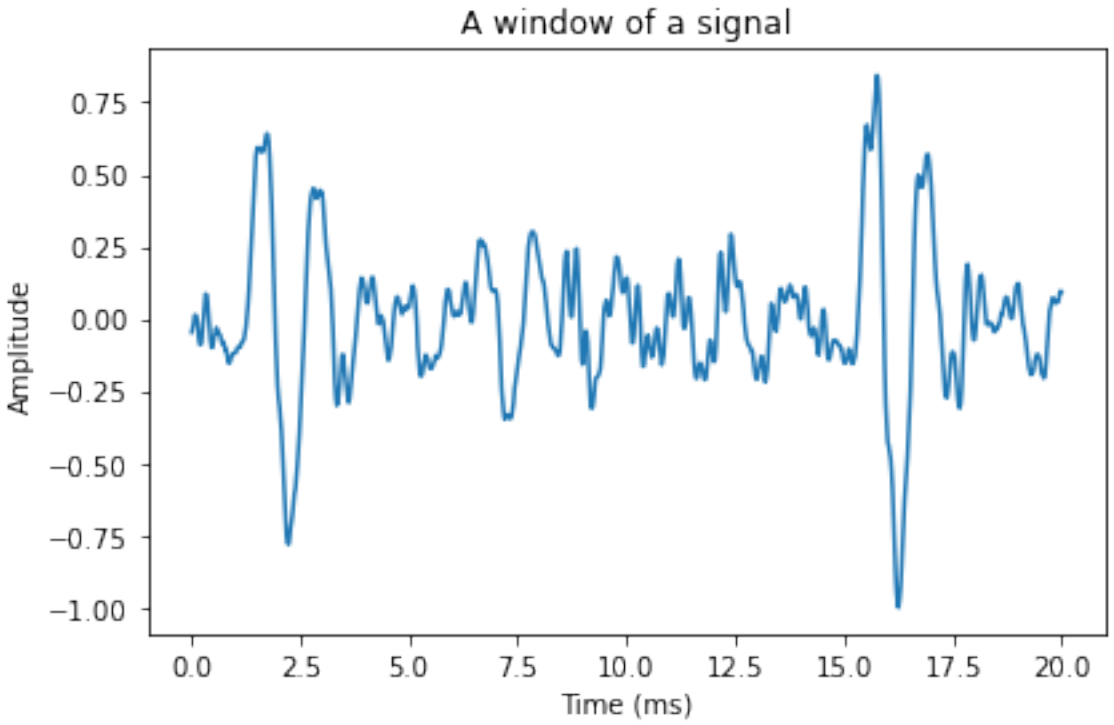
## 3.7 Autocorrelation and autocovariance

Look at the speech signal segment below. On a large scale it is hard to discern a structure, but on a small scale, the signal seems continuous. Speech signals typically have such structure that samples near in time to each other are similar in amplitude. Such structure is often called short-term temporal structure.

More specifically, samples of the signal are *correlated* with the preceding and following samples. Such structures are in statistics measured by covariance and correlation, defined for zero-mean variables x and y as

$$\text{covariance: } \sigma_{xy} = E[xy]$$
$$\text{correlation: } \rho_{xy} = \frac{E[xy]}{\sqrt{E[x^2]E[y^2]}},$$

where $E[\cdot]$ is the expectation operator.




For a speech signal $x_n$, where $n$ is the time-index, we would like to measure the correlation between two time-indices $x_n$ and $x_h$. Since the structure which we are interested in appears when $n$ and $h$ are near each other, it is better to measure the correlation between $x_n$ and $x_{n-k}$. The scalar $k$ is known as the *lag*. Furthermore, we can assume that the correlation is uniform over all $n$ within the segment. The self-correlation and -covariances, known as the *autocorrelation* and *autocovariance* are defined as

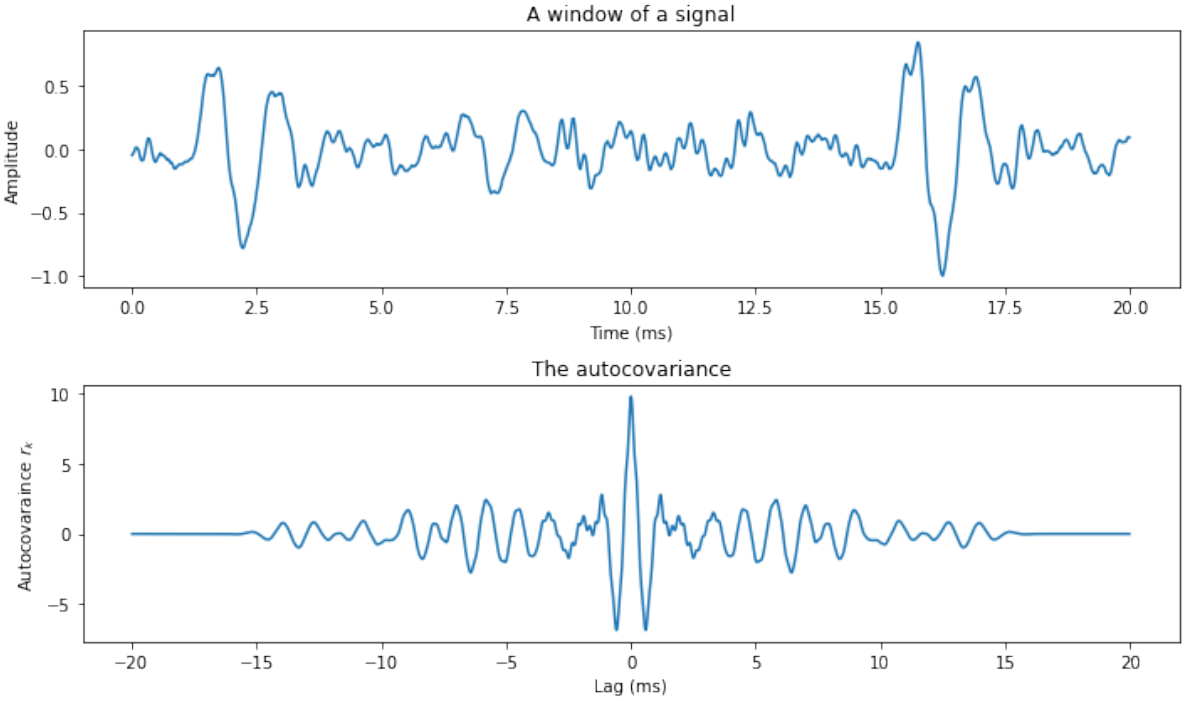
$$\text{autocovariance: } r_k = E_n[x_n x_{n-k}]$$
$$\text{autocorrelation: } c_k = \frac{E_n[x_n x_{n-k}]}{E_n[x_n^2]} = \frac{r_k}{r_0}.$$

The figure below illustrates the autocovariance of the above speech signal. We can immediately see that the short-time correlations are preserved - on a small scale, the autocovariance looks similar to the original speech signal. The oscillating structure is also accurately preserved.

Because we assume that the signal is stationary, and as a consequence of the above formulations, we can readily see that autocovarinaces and -correlations are symmetric

$$r_k = E_n[x_n x_{n-k}] = E_n[x_{n+k} x_{n+k-k}] = E_n[x_{n+k} x_n] = r_{-k}.$$

This symmetry is clearly visible in the figure below, where the curve is mirrored around lag 0.




The above formulas use the expectation operator $E[\cdot]$ to define the autocovariance and -correlation. It is an abstract tool, which needs to be replaced by a proper estimator for practical implementations. Specifically, to estimate the autocovariance from a segment of length $N$, we use

$$r_k \approx \frac{1}{N-1} \sum_{k=1}^{N-1} x_n x_{n-k}.$$

Observe that the speech signal $x_n$ has to be *windowed* before using the above formula.

We can also make an on-line estimate (a.k.a. leaky integrator) of the autocovariance for sample position $n$ with lag $k$ as

$$\hat{r}_k(n) := \alpha x_n x_{n-k} + (1-\alpha)\hat{r}_k(n-1),$$

where $\alpha \in [0,1]$ is a small positive constant which determines how rapidly the estimate converges.

It is often easier to work with vector notation instead of scalars, whereby we need the corresponding definitions for autocovariances. Suppose

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}.$$

We can then define the autocovariance matrix as

$$R_x := E[xx^T] = \begin{bmatrix} E[x_0^2] & E[x_0 x_1] & \dots & E[x_0 x_{N-1}] \\ E[x_1 x_0] & E[x_1^2] & \dots & E[x_1 x_{N-1}] \\ \vdots & \vdots & \ddots & \vdots \\ E[x_{N-1} x_0] & E[x_{N-1} x_1] & \dots & E[x_{N-1}^2] \end{bmatrix} = \begin{bmatrix} r_0 & r_1 & \dots & r_{N-1} \\ r_1 & r_0 & \dots & r_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & r_{N-1} & \dots & r_0 \end{bmatrix}.$$

Clearly $R_x$ is thus a symmetric Toeplitz matrix. Moreover, since it is a product of $x$ with itself, $R_x$ is also positive (semi-)definite.

## 3.8 The cepstrum, mel-cepstrum and mel-frequency cepstral coefficients (MFCCs)

The spectrogram is a useful representation of speech in the sense that it visualizes effectively many pertinent features of speech signals. In particular, we can observe events over time, changes in fundamental frequency and also some features of the spectral envelope. It however also comes with its drawbacks. It is not a particularly efficient representation in terms of number of coefficients; the spectrum has a large number of coefficients in comparison to the amout of information which we are after. Typically we would like to have information of the formant locations and amplitudes, which could be represented by just a handful of coefficients. Similarly, the fundamental frequency is just one piece of information, but it is hidden in a multitude of frequency components.

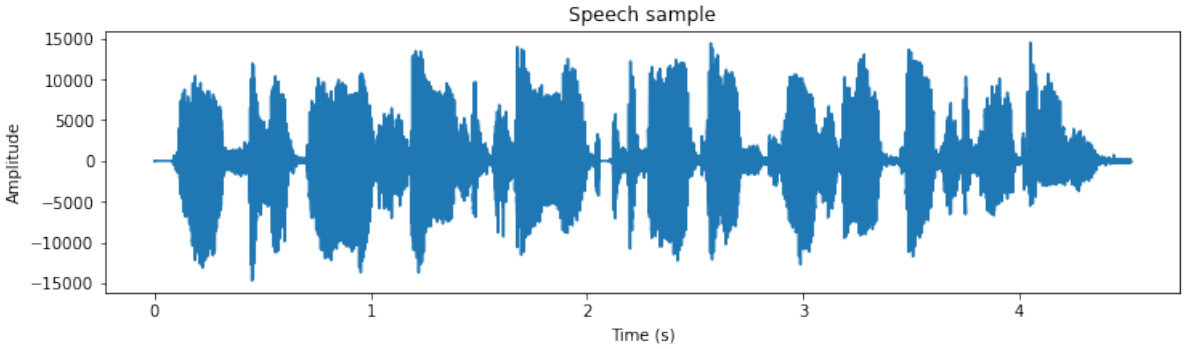
### 3.8.1 The cepstrum

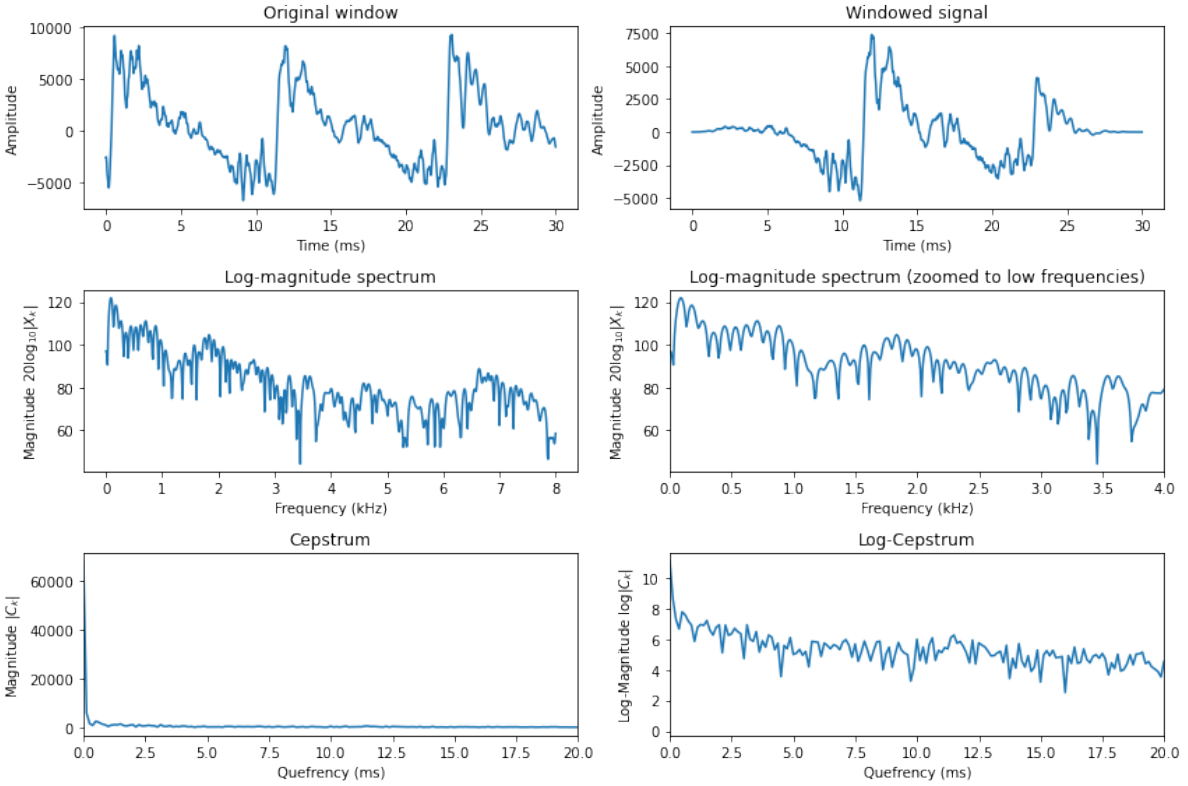
Many central properties of speech signals are clearly visible as structures in the log-spectrum, for example, the envelope is the smooth macro-structure and the fundamental frequency $F_0$ is a harmonic comb-structure. Both types of structures can be easily analyzed with a time-frequency transform. Specifically, since the structures are visible in the log-spectrum, we should take the discrete Fourier transform or discrete cosine transform of the log-spectrum. We are thus applying two time-frequency transform to the original time signal, but with the non-linear operation, logarithm of the absolute value, in between.

The algorithm is

1. Apply analysis windowing to signal
2. Apply time-frequency transform (DFT or DCT)
3. Take the logarithm of the absolute value
4. Apply second time-frequency transform

The output after the second time-frequency transform is known as the *cepstrum*, because it is backwards transform of the *spectrum*. The word-play was chosen such that it would be funny. Similarly, the x-axis in the cepstrum is known as the *quefrency*, as corresponding to *frequency*. Since the cepstrum is a result of two consecutive time-frequncy transforms, the unit in the quefrency will be *seconds*.

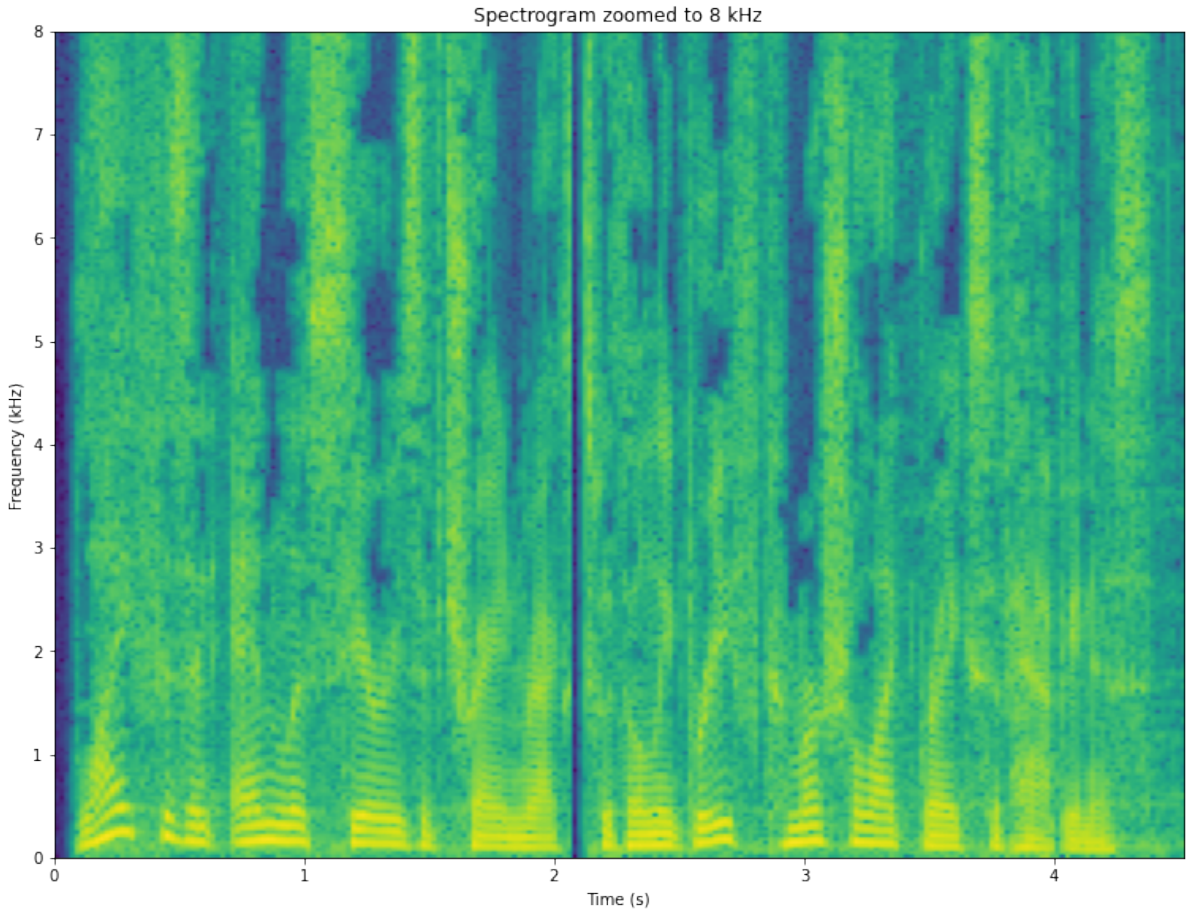


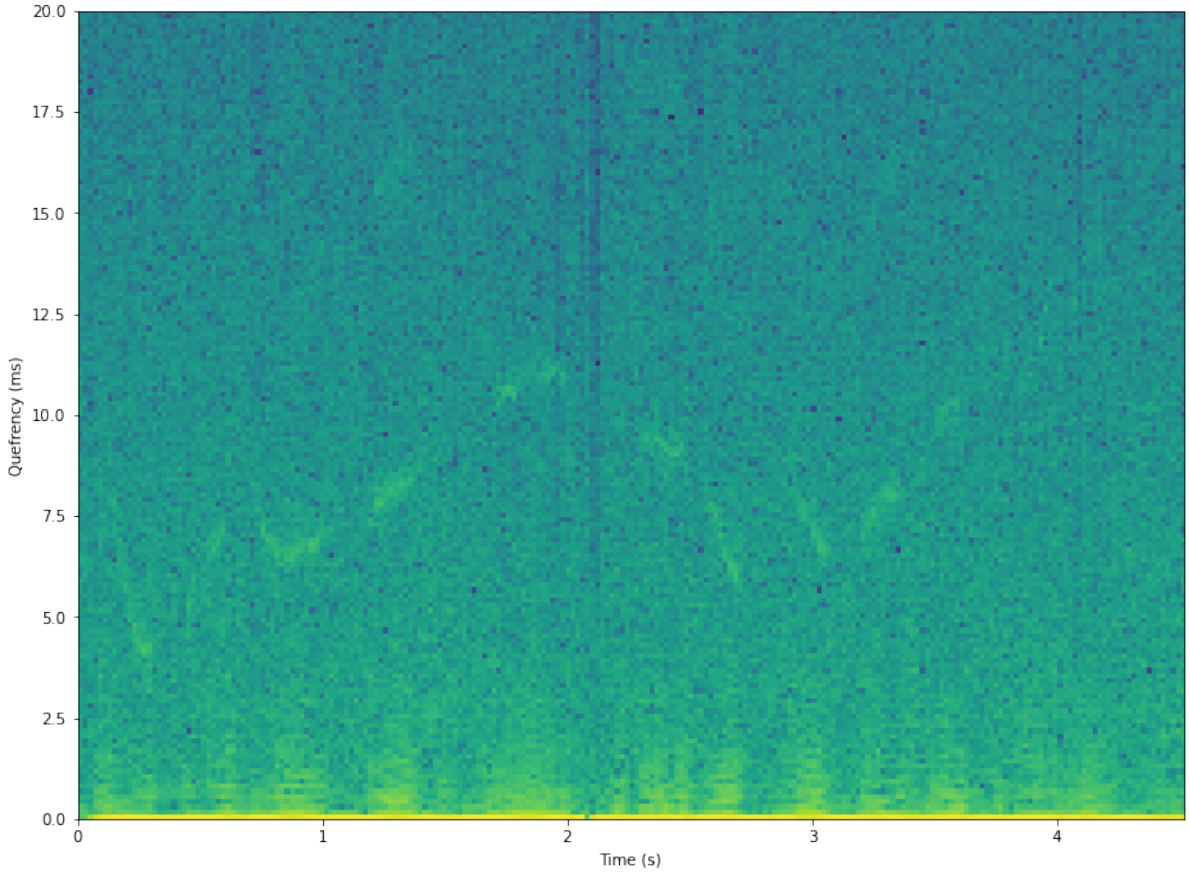



### 3.8.2 Features in the Cepstrum

The envelope of the spectrum is a smoothed version, so it should be present in the low part of the cepstrum. The interpretation of the lowest coefficients is however not intuitive. Moreover, observe that the power spectrum can sometimes be zero or very close to zero, such that the log-spectrum approaches negative infinity. Isolated zeros should not have any effect on the envelope, but in the log-domain, very large negative values have a large contribution to the cepstrum. The cepstrum, as such, is therefore not very well suited for envelope modelling.

However, the F0 is usually prominently visible as a peak in the cepstrum. Since this domain is similar to the time-domain, the cepstral peak will be visible at the same quefrency value as the period-length of the original time signal.


Spectrogram zoomed to 8 kHz
Frequency (kHz)
Time (s)
0
1
2
3
4
5
6
7
8

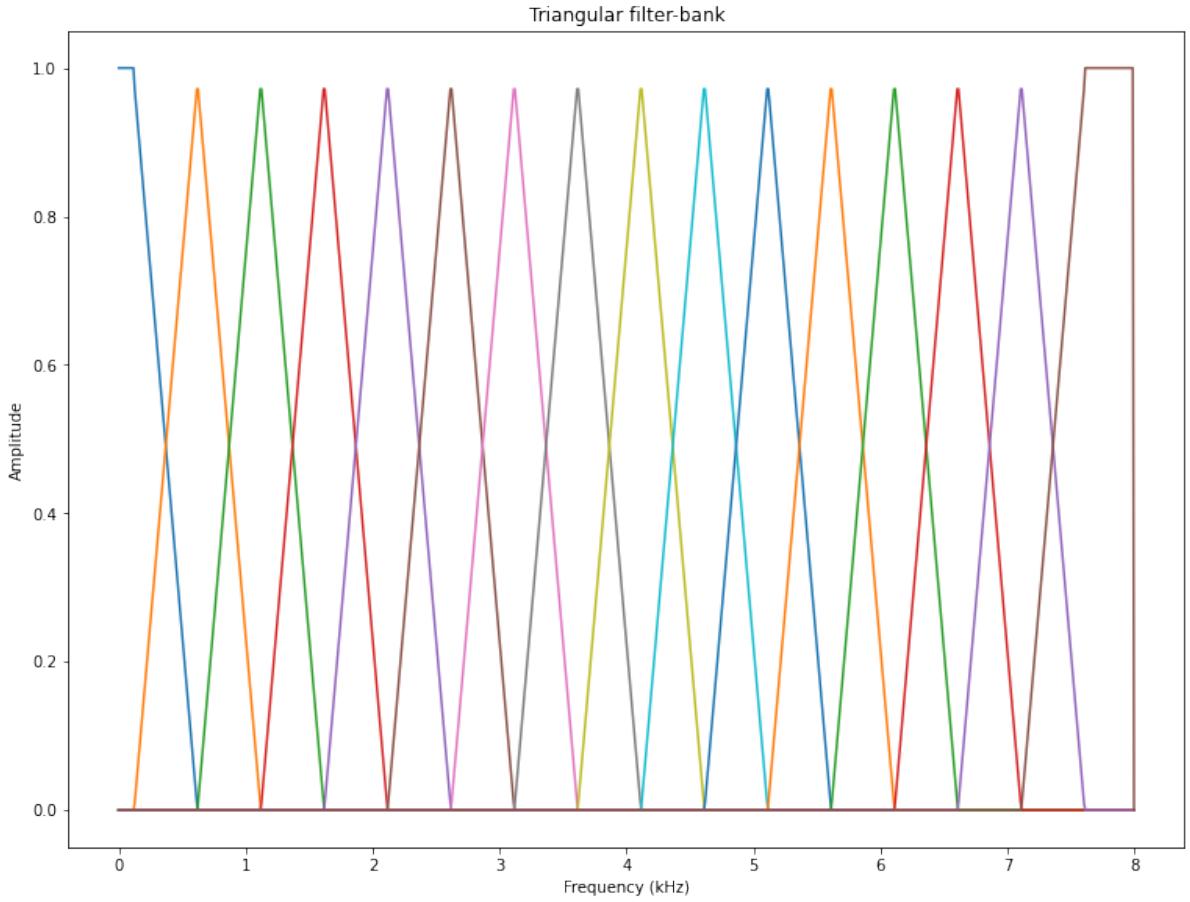


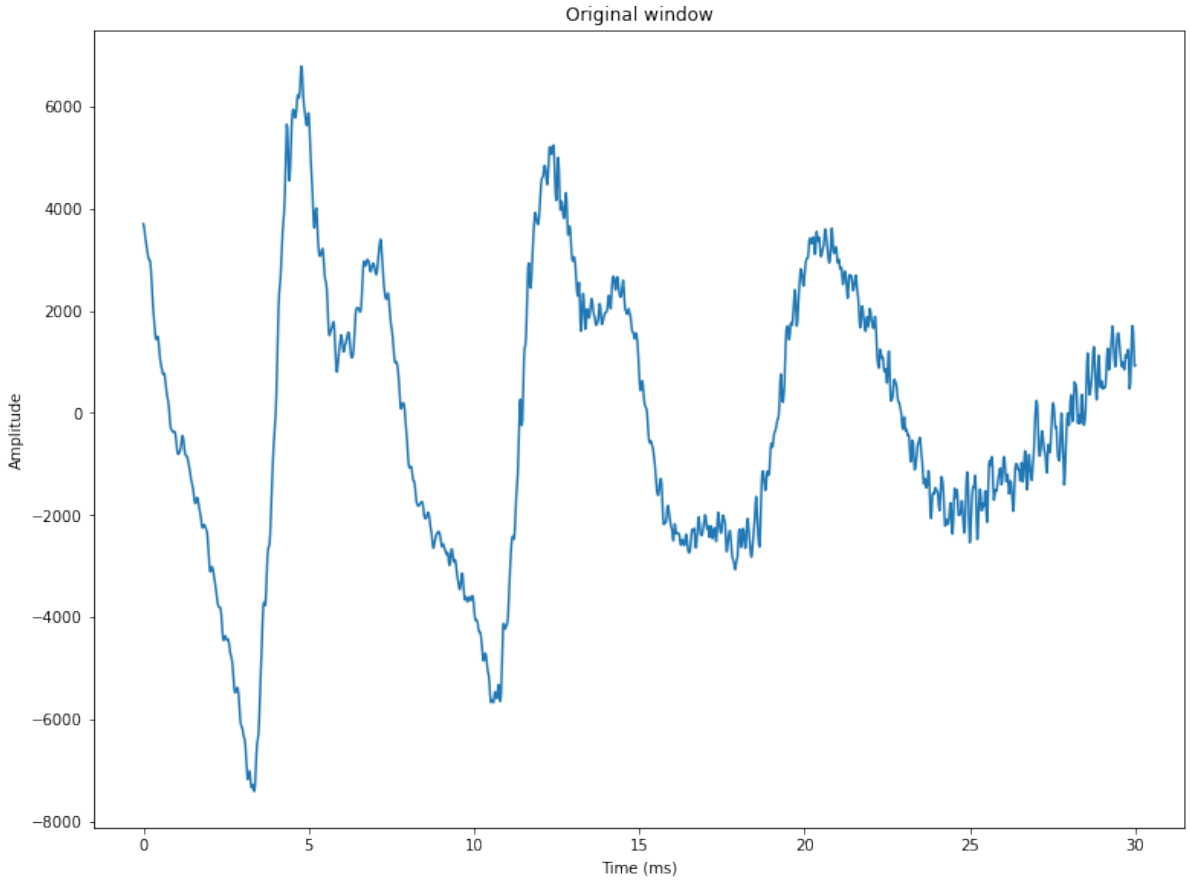

We can clearly see the a curve between quefrency 5 and 12 ms undulating up and down over time. The quefrencies $q$ can be easily converted to frequencies $f$ by $f = 1/q$ (but remember to first convert milliseconds to seconds).

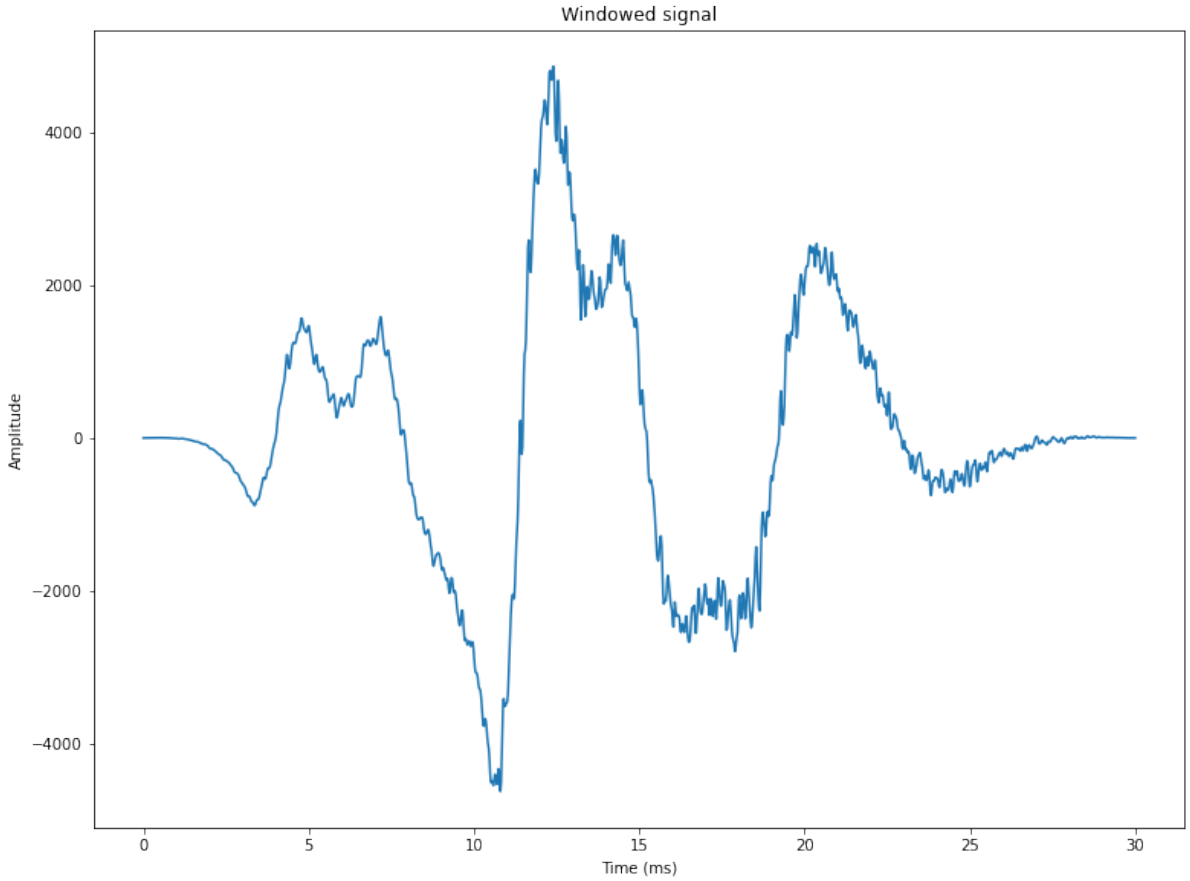
### 3.8.3 Down-sampling the log-spectrum

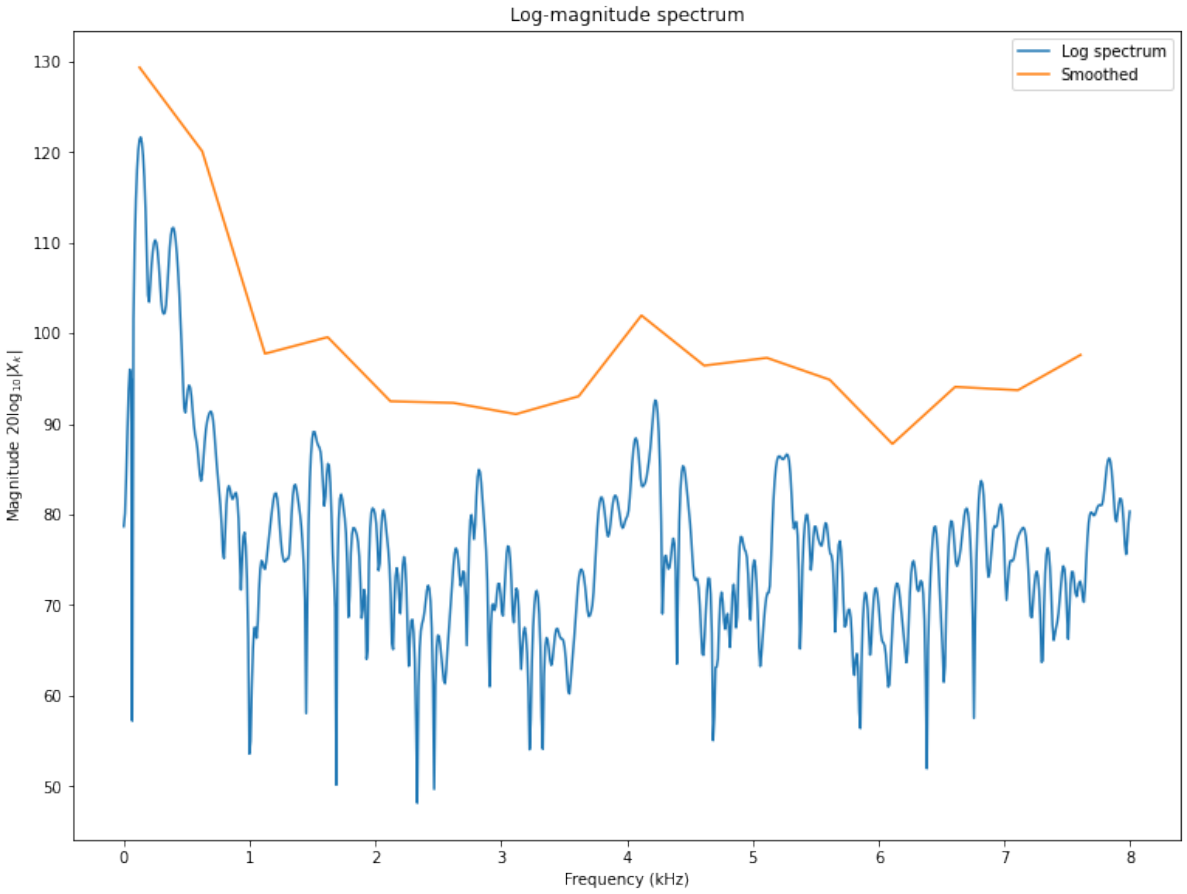
The cepstrum is good for extracting envelope and F0-information, but it is not particularly efficient in the sense that it has a large number of coefficients for a little amount of information. The envelope information is about the slowly-varying shape of the log-spectrum, so we could try to extract that by a simple downsampling. However, a problem is that the power-spectrum can sometimes have arbitrarily small values, which in the log-spectrum translate to negative near-infinite values. Any information extraction in the log-domain would therefore be susceptible for bias to negative infinite.

A solution is to apply smoothing in the power-spectrum. For example, we could use a FIR-filter $[0.5, 1, 0.5]$, or more generally, a triangular shape. In heuristic terms, we just take an average of the power around a certain frequency, such that frequencies near have a larger weight than frequencies far away. The weighting parameters are then chosen in a triangular shape. We calculate the smoothed samples at intervals corresponding to half the width of the triangle, such that the amount of samples obtained corresponds to the amount of smoothing. More rigorously, we can first apply the FIR-filter and then apply down-sampling by an appropriate amount, but the end-result is the same.


Triangular filter-bank
Amplitude
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7
8
Frequency (kHz)


Original window
Amplitude
Time (ms)
6000
4000
2000
0
−2000
−4000
−6000
−8000
0
5
10
15
20
25
30

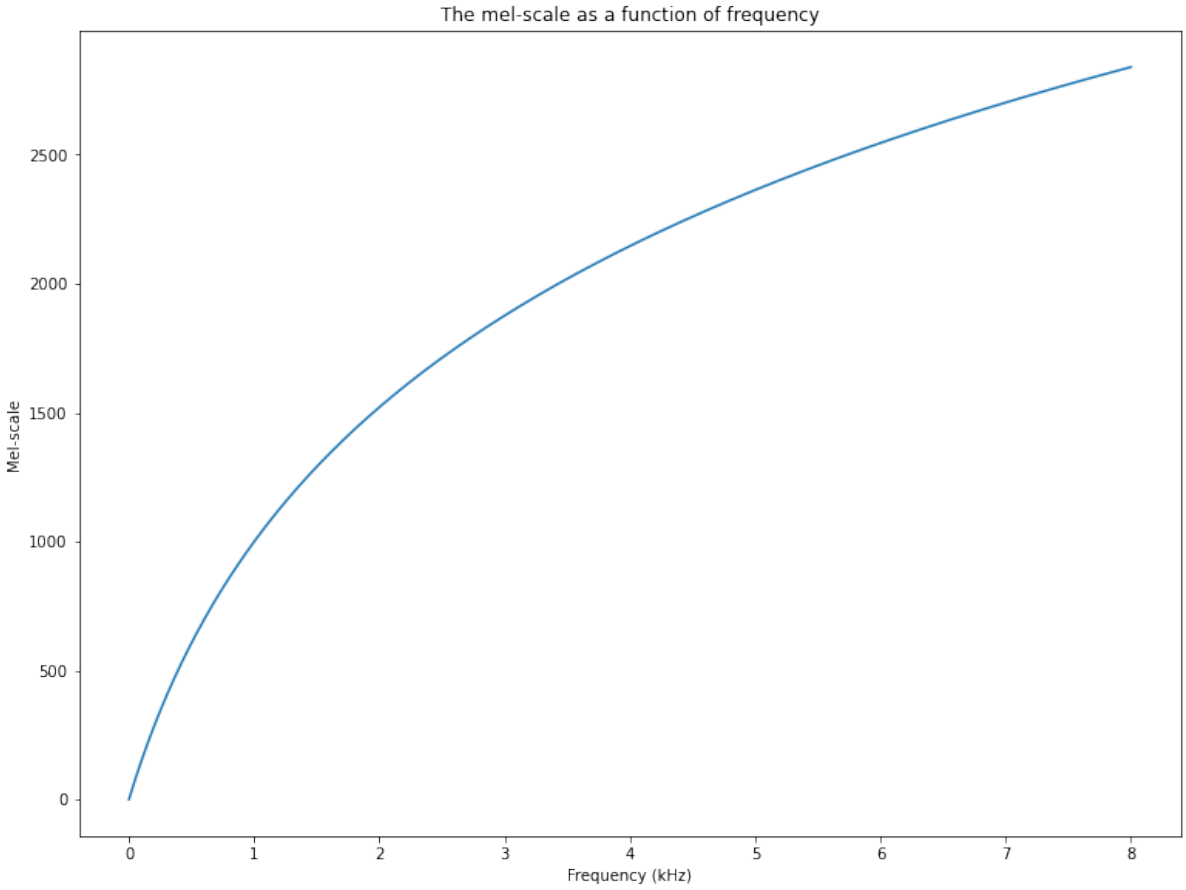

Windowed signal
Amplitude
4000
2000
0
−2000
−4000
0
5
10
15
20
25
30
Time (ms)

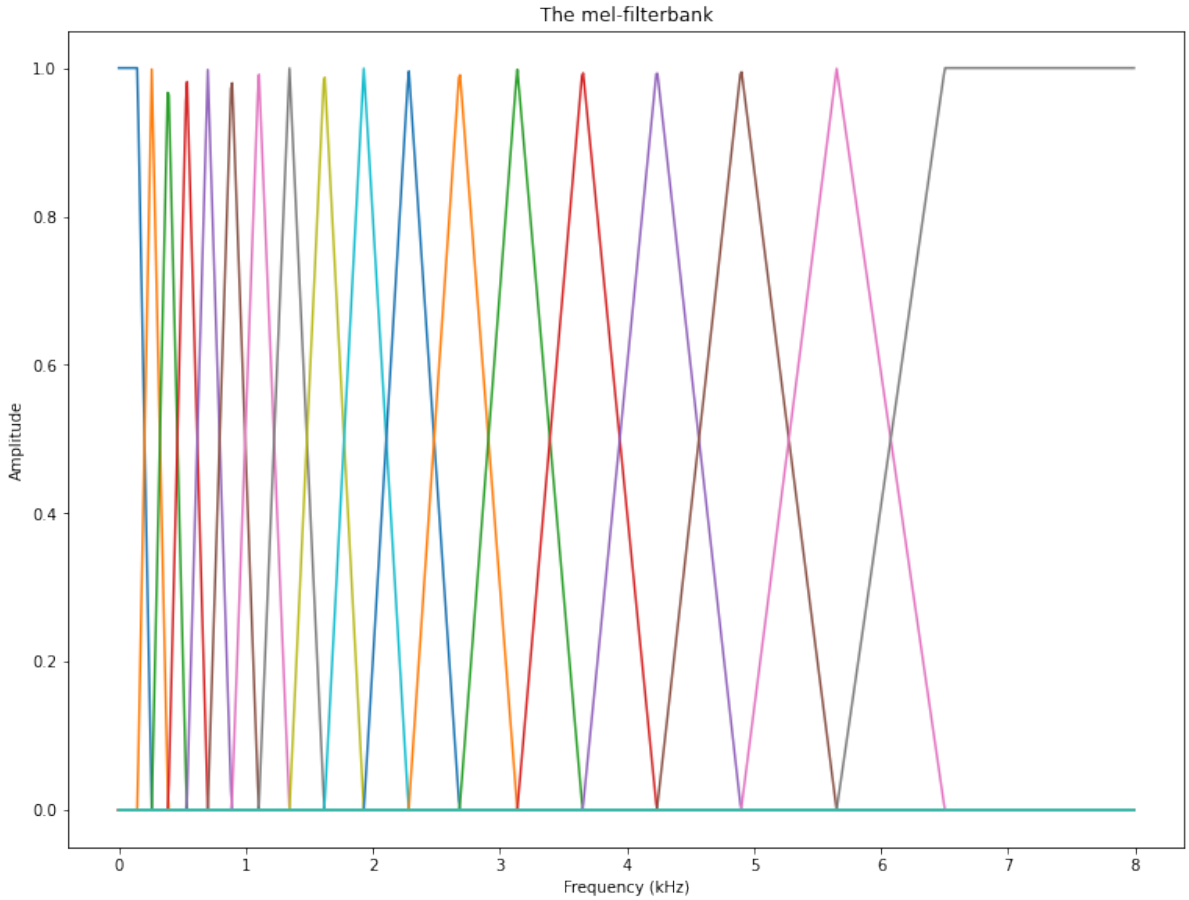


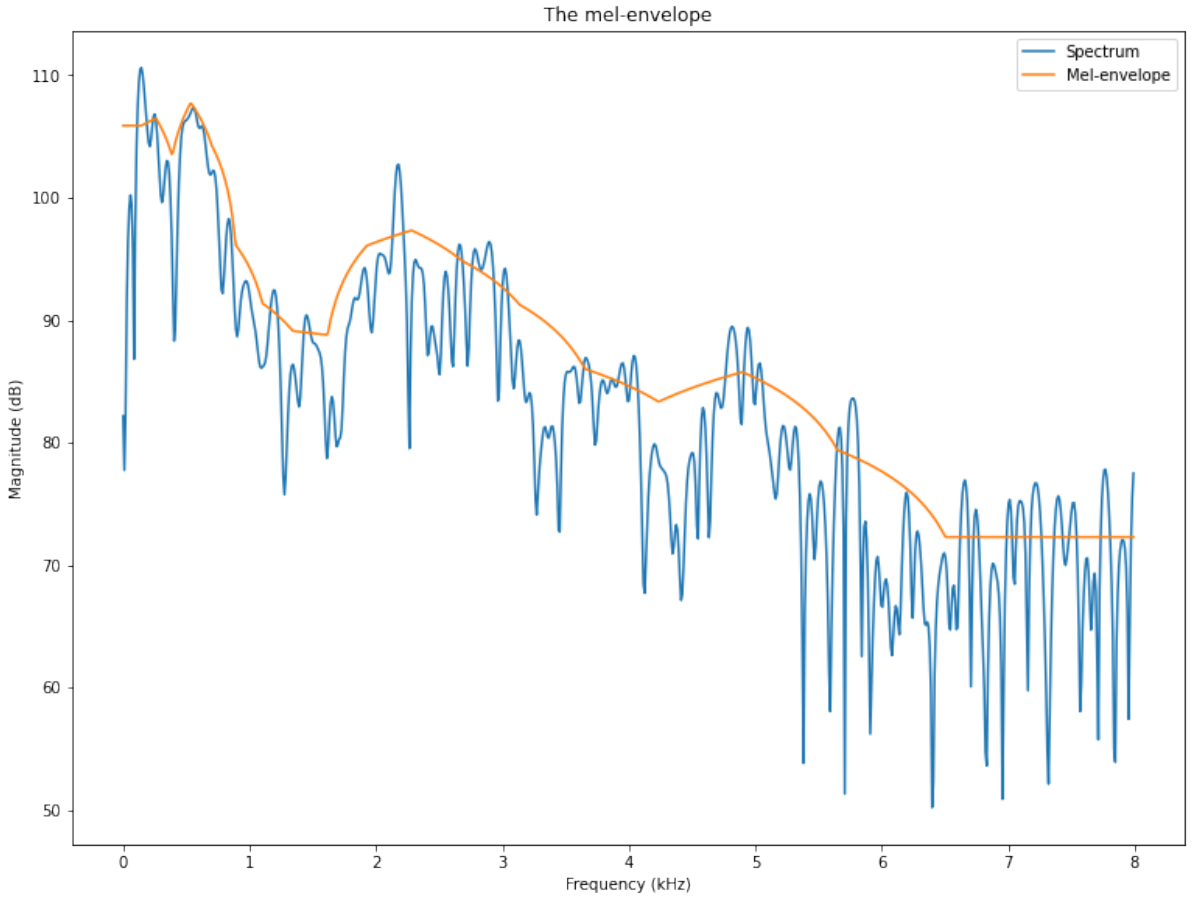

The smoothed representation clearly catches the overall shape of the spectrum, which is the envelope. It achieves this with a low number of coefficient, which means that it is a reasonably efficient model of the envelope. However, a downside is that the information does not reflect the importance of features for humans. Taking the log-transform does map magnitudes to a perceptual scale, but the frequency scale is still not mapped to a perceptual domain.

### 3.8.4 Mel-scale

The mel-scale is a scale which maps frequencies such that steps between tones align with our perception of steps. That is, for example, the step from X to X+1 mel sounds as large as the step from Y to Y+1 mel. More details in the above link. We then form a filterbank such that the triangle-centres are at the frequencies corresponding to equal distance steps on the mel scale.


The mel-scale as a function of frequency
Mel-scale
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
7
8
Frequency (kHz)


The mel-filterbank
Amplitude
Frequency (kHz)
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7
8




The mel-envelope clearly models lower frequencies accurately, which is also where the all-important formants reside. That is, accuracy is concentrated on the important part, which is good. Higher frequencies, above 6.5 kHz in particular, are poorly modelled, but there is usually not too much energy anyway, so that is ok.

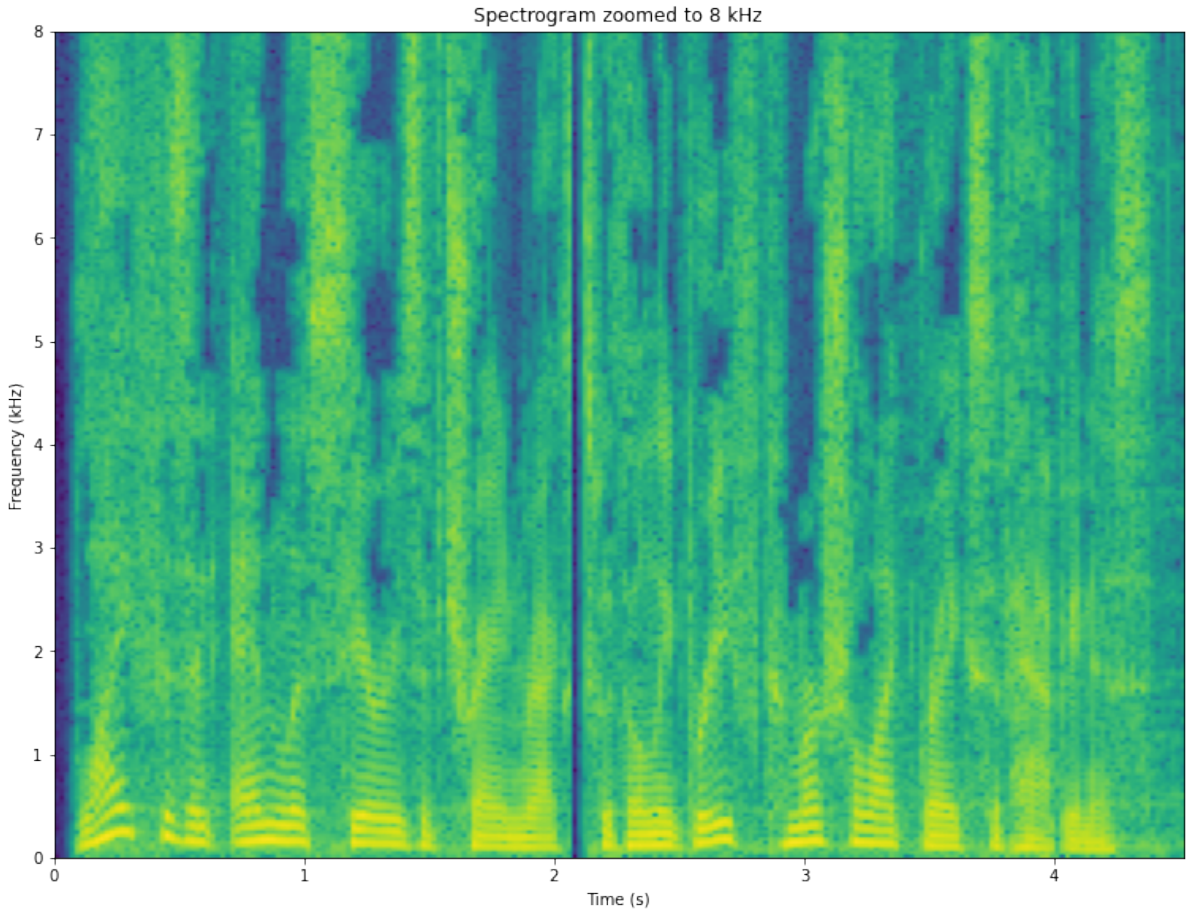
A remaining issue with the log-melspectrum is however that neighbouring samples are highly correlated. That is, information is distributed throughout the individual samples. Yet we cannot reduce accuracy more, because then we would start loosing accuracy of the formants.

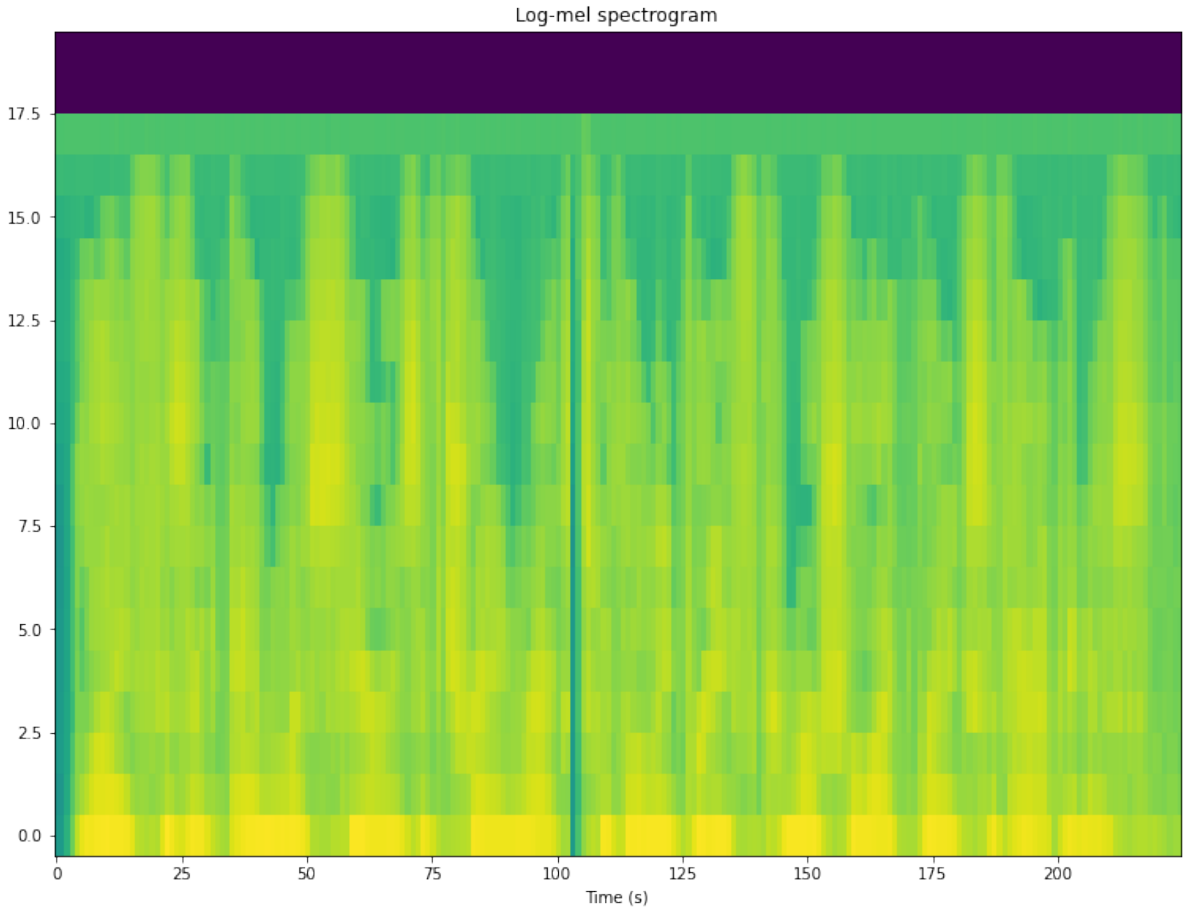
### The Mel-Frequency Cepstral Coefficients (MFCCs)

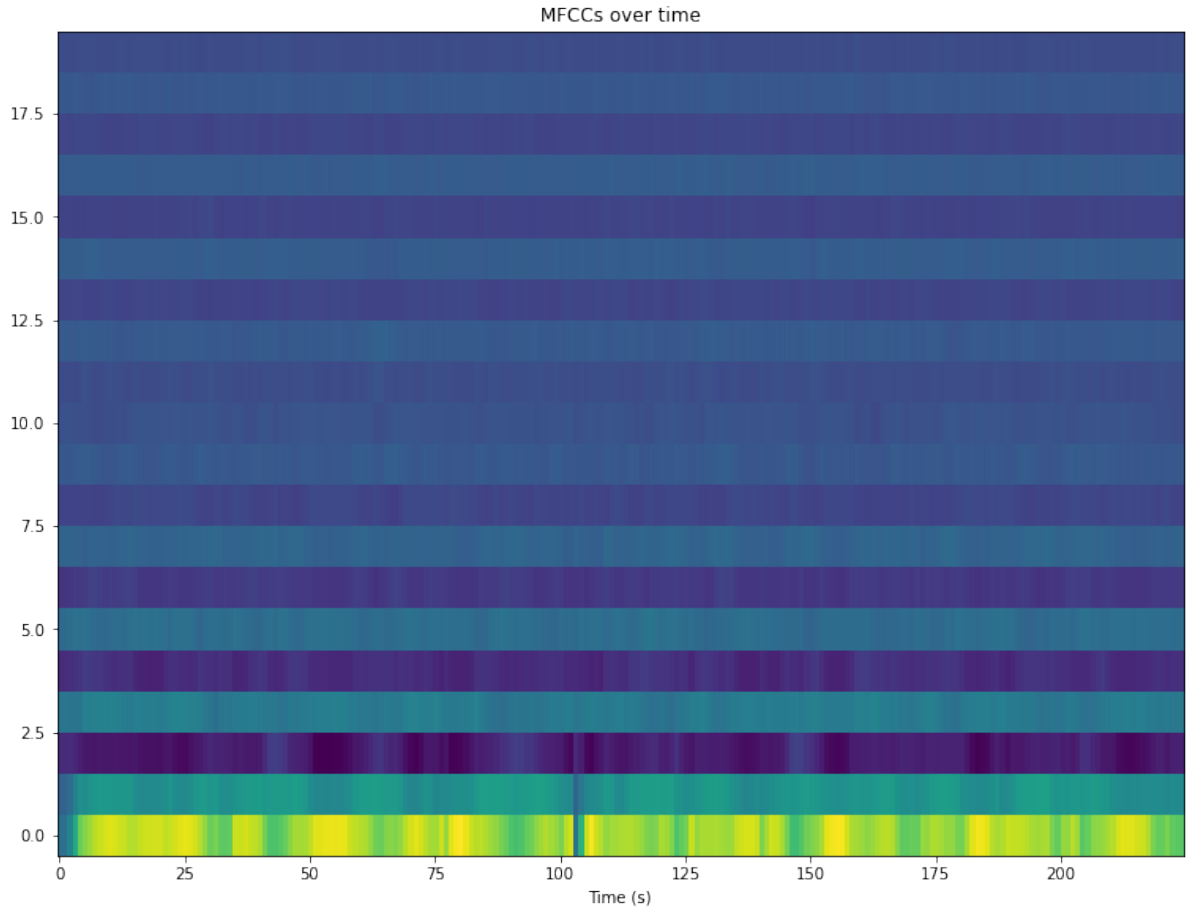
A generic operation for decorrelating sequentially correlated data is the discrete cosine transform (DCT). That is, say we have a time-signal which has correlation over time, by taking the DCT, we obtain the spectrum of the signal, where samples are reasonably uncorrelated (at least when the input vector is a stationary state system and long).

Similarly, we can thus take the DCT of the log-mel spectrum, which is known as the Mel-Frequency Cepstral coefficient (MFCC) representation. It has the mel-frequency mapping, then takes the logarithm and finally the DCT.

The MFCC is an abstract domain, which is not easy to interpret visually. However, since it is designed to correspond to resemble perception in both magnitude and frequency axis, and to be roughly uncorrelated, it is efficient for computation. Observe that this is a perceptual argument, such that the computer simulates how humans perform. Observe that simulating the human is not always appropriate, computers can sometimes do better than humans, but often it focuses attention to the relevant sources of information.


Spectrogram zoomed to 8 kHz
Frequency (kHz)
Time (s)
0
1
2
3
4
5
6
7
8


Log-mel spectrogram
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
25
50
75
100
125
150
175
200
Time (s)




The envelope is clearly visible in the log-mel spectrogram as expected. The MFCCs behave also as expected - not much is visible. Would need to normalize it over time to boost subtle changes to become visible. Such normalization (pre-whitening) is in fact a standard step for all variables which are used as input to machine learning.

### 3.8.5 Conclusion

We have discussed the cepstrum and how it can be used to extract F0 information from a speech signal. We also showed how to extract envelope information, in particular with frequencies mapped to the mel-scale to correspond with perception of pitch. Finally, we showed how the log-mel spectrum can be decorrelated to obtain the mel-frequency cepstral coefficients, the MFCCs. In fact, the main point of this whole exercise was to introduce the MFCCs, because it is the most used analysis method for all of speech and audio, especially as a front-end for machine learning methods. It is so common that if nothing is mentioned, then the assumption is that everyone will use MFCCs in machine learning.

The reason that MFCCs are used so often is that it works well. It is somehow capable of capturing essential information from speech and audio, even if we do not entirely understand why. There are many particular choices, which could be changed, but which seem to have either no effect at all or reduces the usefulness of the outcome. For example, the mel-scale is not a well-motivated choice. Pitch-perception is a particular thing, whereas the more generic perceptual models would be models of the hair-cells in the inner ear, approximated by, for example, the equivalent rectangular bandwidth (ERB) scale. The MFCCs also have a historical momentum; they have been used so long already, that we would need a very good reason to move to some other model. There is benefit from having a model which everyone are familiar with.

## 3.9 Linear prediction

### 3.9.1 Definition

Speech is a continuous signal, which means that consecutive samples of the signal are correlated (see figure on the right). In particular, if we know a previous sample $x_{n-1}$, we can make a *prediction* of the current sample, $\hat{x}_n = x_{n-1}$, such that $\hat{x}_n \approx x_n$. By using more previous samples we have more information, which should help us make a better prediction. Specifically, we can define a predictor which uses $M$ previous samples to predict the current sample $x_n$ as

$$\hat{x}_n = -\sum_{k=1}^{M} a_k x_{n-k}.$$

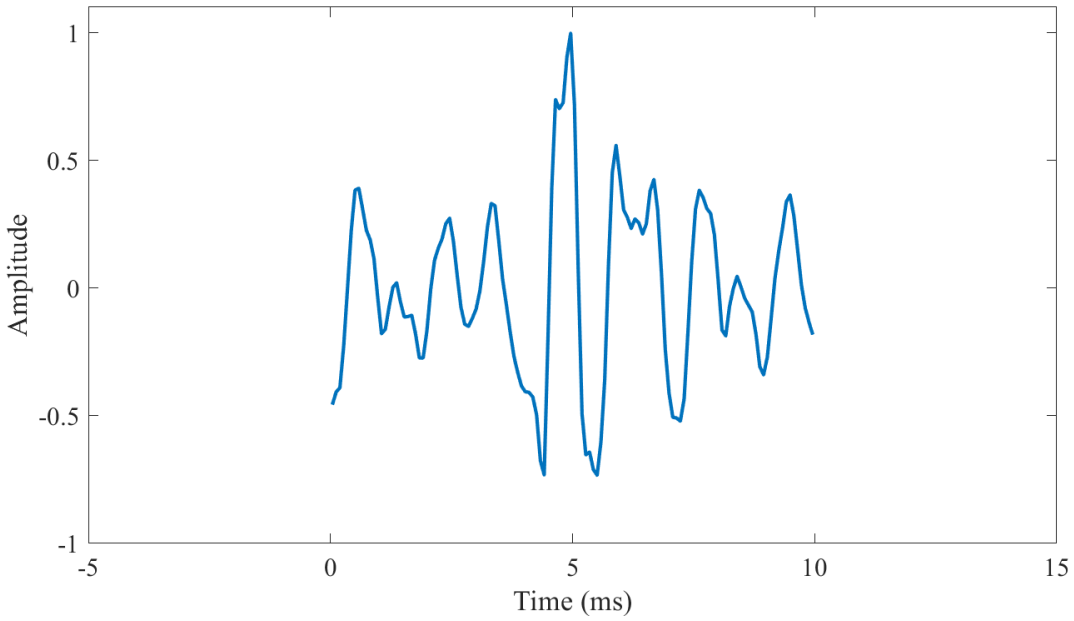
This is a *linear predictor* because it takes a linearly weighted sum of past components to predict the current one.

The error of the prediction, also known as the *prediction residual* is

$$e_n = x_n - \hat{x}_n = x_n + \sum_{k=1}^{M} a_k x_{n-k} = \sum_{k=0}^{M} a_k x_{n-k},$$

where $a_0 = 1$. This explains why the definition $\hat{x}_n$ included a minus sign; when we calculate the residual, the double negative disappears and we can collate everything into one summation.

A short segment of speech. Notice how consecutive samples are mostly near each other, which means that consecutive samples are correlated.



### 3.9.2 Vector notation

Using vector notation, we can make the expressions more compact

$$e = Xa$$

where

$$e = \begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ e_{N-1} \end{bmatrix}, \qquad X = \begin{bmatrix} x_0 & x_{-1} & \dots & x_M \\ x_1 & x_0 & \dots & x_{M-1} \\ \vdots & \vdots & & \vdots \\ x_{N-1} & x_{N-2} & \dots & x_{N-M} \end{bmatrix}, \qquad a = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{bmatrix}.$$

Here we calculated the residual for a length $N$ frame of the signal.

### 3.9.3 Parameter estimation

Vector $a$ holds the unknown coefficients of the predictor. To find the best possible predictor, we can minimize the minimum mean-square error (MMSE). The square error is the 2-norm of the residual, $\|e\|^2 = e^T e$ . The mean of that error is defined as the expectation

$$E\left[|e|^2\right] = E\left[a^T X^T X a\right] = a^T E\left[X^T X\right] a = a^T R_x a,$$

where $R_x = E\left[X^T X\right]$ and $E\left[\cdot\right]$ is the expectation operator. Note that, as shown in the *autocorrelation section*, the matrix $R_x$, can be usually assumed to have a symmetric Toeplitz structure.

If we would directly minimize the mean-square error $E\left[\|e\|^2\right]$, then clearly we would obtain the trivial solution $a = 0$, which is not particularly useful. However that solution contradicts with the requirement that the first coefficient is unity, $a_0 = 1$. In vector notation we can equivalently write

$$a_0 - 1 = u^T a - 1 = 0, \qquad \text{where } u = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The standard method for quadratic minimization with constraints is to use a Langrange multiplier, λ, such that the objective function is

$$\eta(a, \lambda) = a^T R_x a - 2\lambda\left(a^T u - 1\right).$$

This function can be heuristically interpreted such that λ is a free parameter. Since our objective is to minimize $a^T R_x a$ if $a^T u - 1$ is non-zero, then the objective function can become arbitrarily large. To allow any value for λ, the constraint must therefore be zero.

The objective function is then minimized by setting its derivative with respect to $a$ to zero

$$0 = \frac{\partial}{\partial a}\eta(a, \lambda) = \frac{\partial}{\partial a}\left[a^T R_x a - 2\lambda\left(a^T u - 1\right)\right] = 2R_x a - 2\lambda u.$$

It follows that the optimal predictor coefficients are found by solving

$$R_x a = \lambda u.$$

Since $R_x$, is symmetric and Toeplitz, the above system of equations can be efficiently solved using the Levinson-Durbin algorithm with algorithmic complexity $O(M^2)$. However, note that with direct solution we obtain $a' := \frac{1}{\lambda} a = R_x^{-1} u$ that is, instead of $a$ we get $a$ scaled with λ. However, since we know that $a_0 = 1$, we can find $a$ by $a = \lambda a' = \frac{a'}{a'_0}$.

### 3.9.4 Spectral properties

Linear prediction is usually used to predict the current sample of a time-domain signal $x_n$. The usefulness of linear prediction however becomes evident by studying its Fourier spectrum Specifically, since $e = Xa$, the corresponding Z-domain representation is

$$E(z) = X(z)A(z) \qquad \Rightarrow \qquad X(z) = \frac{E(z)}{A(z)},$$

where $E(z)$, $X(z)$, and $A(z)$, are the Z-transforms of $e_n$, $x_n$ and $a_n$, respectively. The residual $E(z)$ is white-noise, whereby the inverse $A(z)^{-1}$, must follow the shape of $X(z)$.

In other words, the linear predictor models the macro-shape or *envelope* of the spectrum.

### 3.9.5 Physiological interpretation and model order

Linear prediction has a surprising connection with physical modelling of speech production. Namely, a linear predictive model is equivalent with a *tube-model of the vocal tract* (see figure on the right). A useful consequence is that from the acoustic properties of such a tube-model, we can derive a relationship between the physical length of the vocal tract $L$ and the number of parameters $M$ of the corresponding linear predictor as

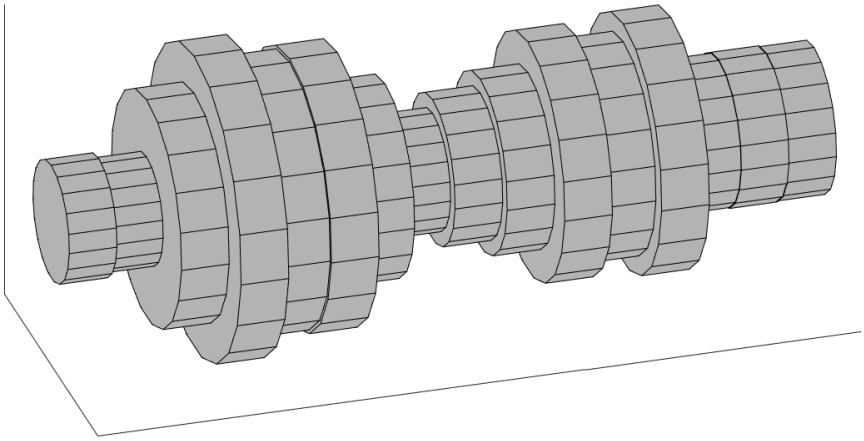
$$M = \frac{2f_s L}{c},$$

where $f_s$ is the sampling frequency and $c$ is the speed of sound. With an air-temperature of 35 C, the speed of sound is $c$=350m/s. The mean length of vocal tracts for females and males are approximately 14.1 and 16.9 cm. We can then choose to overestimate $L$=0.17m. At a sampling frequency of 16kHz, this gives $M \approx 17$ . The linear predictor will catch also features of the glottal oscillation and lip radiation, such that a useful approximation is $M \approx$ round $\left(1.25\frac{f_s}{1000}\right)$ . For different sampling rates we then get the number of parameters $M$ as

| $f_s$ | M |
|---|---|
| 8 kHz | 10 |
| 12.8 kHz | 16 |
| 16 kHz | 20 |

Observe however that even if a tube-model is equivalent with a linear predictor, the relationship is non-linear and highly sensitive to small errors. Moreover, when estimating linear predictive models from speech, in addition to features of the vocal tract, we will also capture features of glottal oscillation and lip-radiation It is therefore very difficult to estimate meaningful tube-model parameters from speech. A related sub-field of speech analysis is glottal inverse filtering, which attempts to estimate the glottal source from the acoustic signal. A necessary step in such inverse filtering is to estimate the acoustic effect of the vocal tract, that is, it is necessary to estimate the tube model.

A tube model of the vocal tract consisting of constant-radius tube-segments



### 3.9.6 Uses in speech coding

Linear prediction has been highly influential especially in early speech coders. In fact, the dominant speech coding method is code-excited linear prediction (CELP), which is based on linear prediction.

### 3.9.7 Alternative representations (advanced topic)

Suppose scalars $a_{m,k}$, are the coefficients of an $M$th order linear predictor. Coefficients of consecutive orders $M$ and $M+1$ are then related as

$$a_{M+1,k} = a_{M,k} + \gamma_{M+1} a_{M,M+1-k},$$

where the real valued scalar $\gamma_M \in (-1, +1)$ is the $M$th reflection coefficient. This formulation is the basis for the Levinson-Durbin algorithm which can be used to solve the linear predictive coefficients. In a physical sense, reflection coefficients describe the amount of the acoustic wave which is reflected back in each junction of the tube-model. In other words, there is a relationship between the *cross-sectional areas* $S_k$ of each tube-segment and the reflection coefficients as

$$\gamma_k = \frac{S_k - S_{k+1}}{S_k + S_{k+1}}.$$

Furthermore, the logarithmic ratio of cross-sectional areas, also known as the *log-area ratios*, are defined as

$$A_k = \log \frac{S_k}{S_{k+1}} = \log \frac{1 - \gamma_k}{1 + \gamma_k}.$$

This form has been used in coding of linear predictive models, but is today mostly of historical interest.

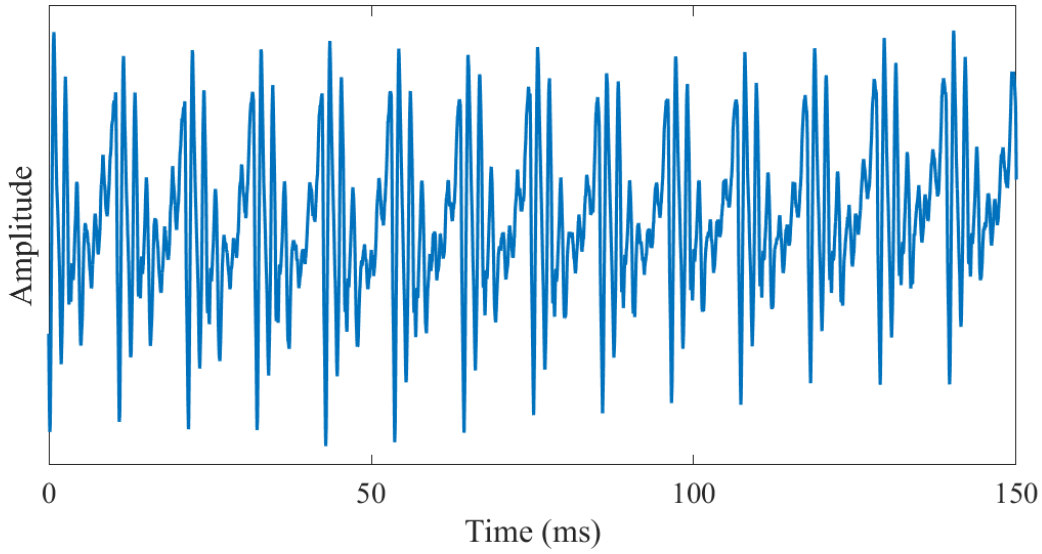
## 3.10 Fundamental frequency (F0)

The fundamental frequency of a speech signal, often denoted by F0 or $F_0$, refers to the approximate frequency of the (quasi-)periodic structure of voiced speech signals. The oscillation originates from the vocal folds, which oscillate in the airflow when appropriately tensed. The fundamental frequency is defined as the average number of oscillations per second and expressed in Hertz. Since the oscillation originates from an organic structure, it is not exactly periodic but contains significant fluctuations. In particular, amount of variation in period length and amplitude are known respectively as *jitter* and *shimmer*. Moreover, the F0 is typically not stationary, but changes constantly within a sentence. In fact, the F0 can be used for expressive purposes to signify, for example, emphasis and questions.

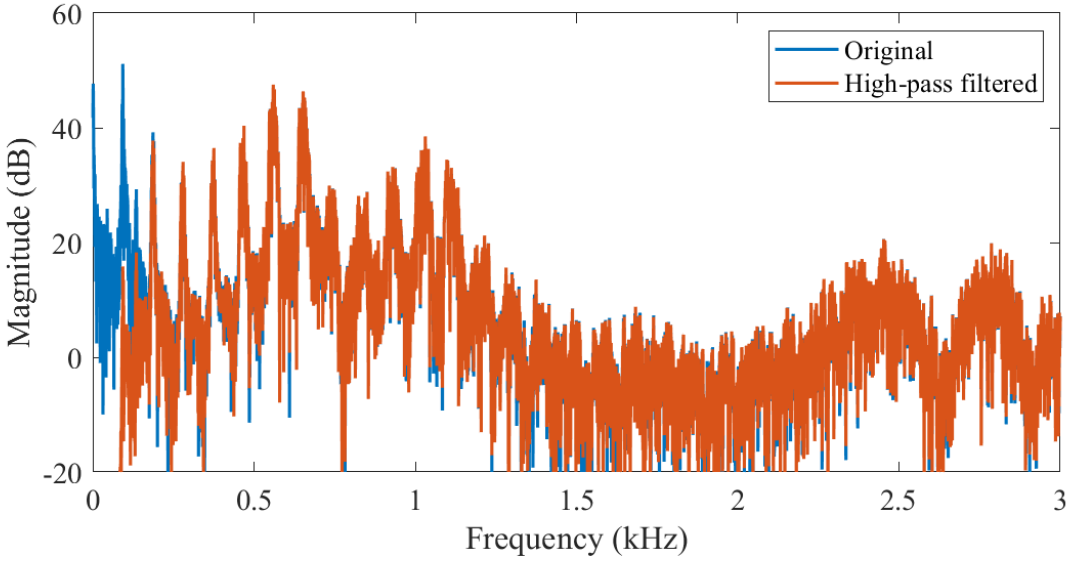
Typically fundamental frequencies lie roughly in the range *80* to *450 Hz*, where males have lower voices than females and children. The F0 of an individual speaker depends primarily on the length of the vocal folds, which is in turn correlated with overall body size. Cultural and stylistic aspects of speech naturally have also a large impact.

The fundamental frequency is closely related to *pitch*, which is defined as our perception of fundamental frequency. That is, the F0 describes the actual physical phenomenon, whereas pitch describes how our ears and brains interpret the signal, in terms of periodicity. For example, a voice signal could have an F0 of 100 Hz. If we then apply a high-pass filter to remove all signal components below 450 Hz, then that would remove the actual fundamental frequency. The lowest remaining periodic component would be 500 Hz, which correspond to the fifth harmonic of the original F0. However, a human listener would then typically still perceive a pitch of 100 Hz, even if it does not exist anymore. The brain somehow reconstructs the fundamental from the upper harmonics. This well-known phenomenon is however still not completely understood.

A speech signal with a fundamental frequency of approximately F0=93Hz.




The spectrum of a speech signal with a fundamental frequency of approximately F0=93Hz (original) and a high-pass filtered version of it such that the fundamental frequency has been removed (high-pass filtered).




A speech signal with a fundamental frequency of approximately F0=93Hz

```
<IPython.lib.display.Audio object>
```

A high-pass filtered version of it such that the fundamental frequency has been removed

```
<IPython.lib.display.Audio object>
```

If $F_0$ is the fundamental frequency, then the length of a single period in seconds is

$$T = \frac{1}{F_0}.$$

The speech waveform thus repeats itself after every $T$ seconds.

A simple way of modelling the fundamental frequency is to repeat the signal after a delay of $T$ seconds. If a signal is sampled with a sampling rate of $F_s$, then the signal repeats after a delay of $L$ samples where

$$L = F_s T = \frac{F_s}{F_0}.$$

A signal $x_n$ then approximately repeats itself such that

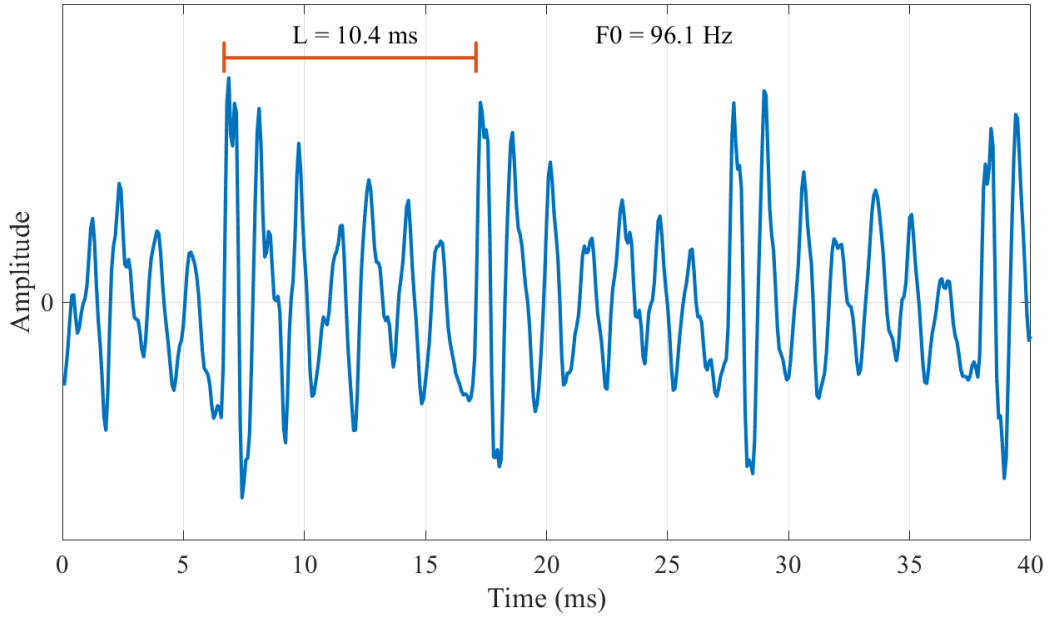
$$x_n \approx x\_n - L \approx x\_n - 2L \approx x\_n - 3L.$$

In the Z-domain this can be modelled by an IIR-filter as

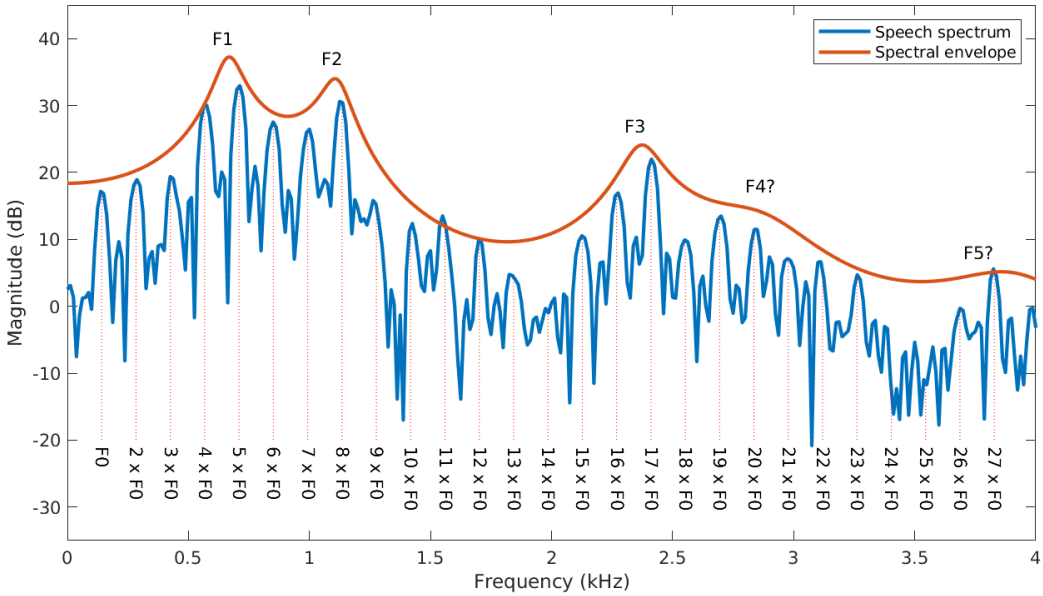
$$B(z) = 1 - \gamma_L z^{-L},$$

where the scalar $0 \leq \gamma_L \leq 1$ scales with the accuracy of the period. The Z-transform of the signal $x_n$ can then be written as $X(z) = B^{-1}(z)E(z)$, where $E(z)$ is the Z-transform of a single period.

Segment of a speech signal, with the period length $L$, and fundamental frequency $F_0 = 1/L$.

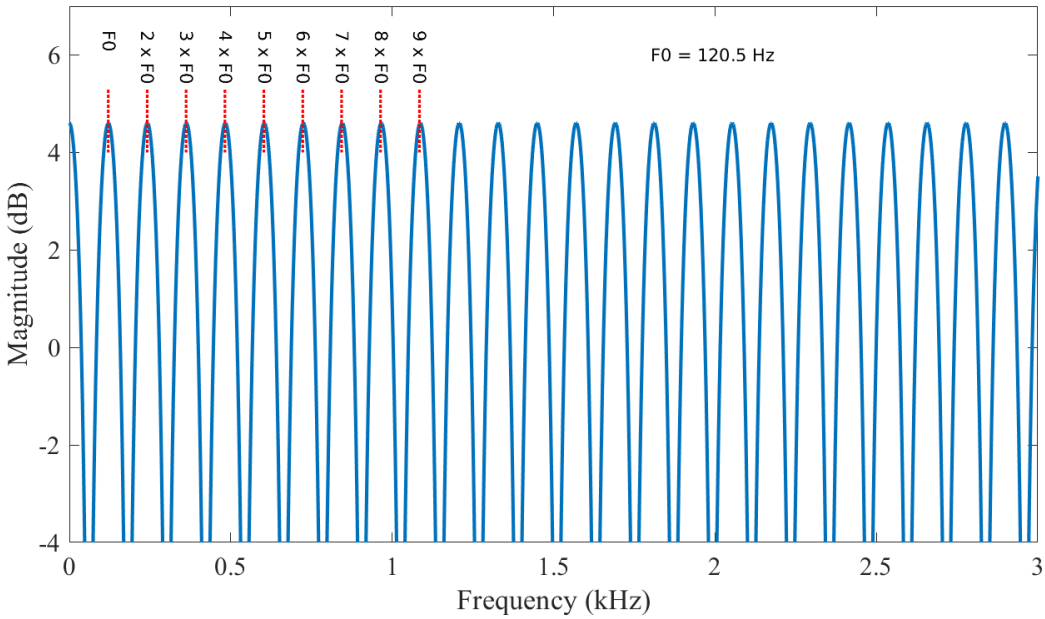



Spectrum of speech signal with the fundamental frequency $F_0$ and harmonics $kF_0$, as well as the formants *F1*, *F2*, *F3*… Notice how the harmonics form a regular comb-structure.




The magnitude spectrum of $B^{-1}(z)$, has then a periodic comb-structure. That is, the magnitude spectrum has peaks at $k\,F_0$ , for integer $k$. For a discussion about the fundamental frequency in the cepstral domain, see Cepstrum and MFCC.

Spectrum of fundamental frequency model $B^{-1}(z)$, showing the characteristic comb-structure with harmonic peaks appearing at integer multiples of $F0$.



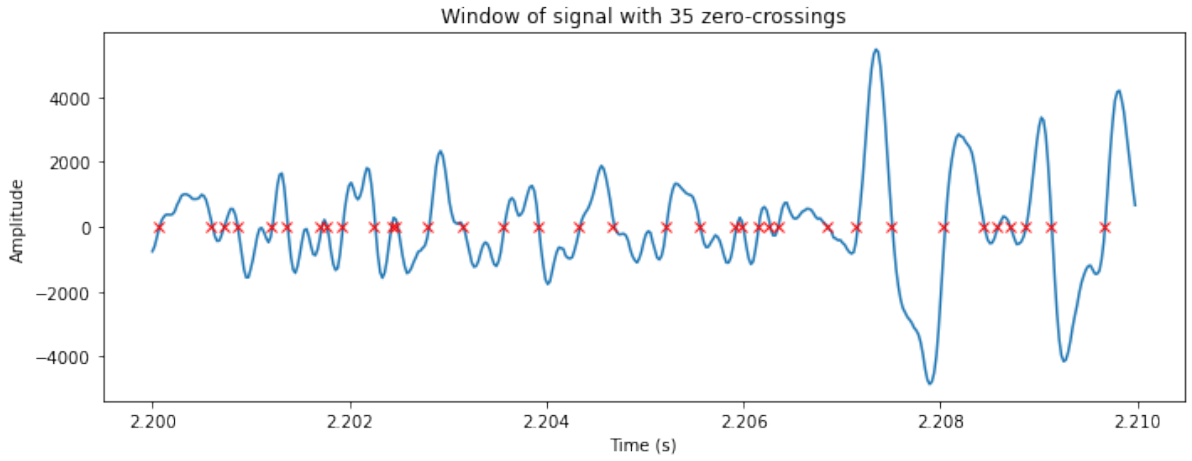

## 3.11 Zero-crossing rate

By looking at different speech and audio waveforms, we can see that depending on the content, they vary a lot in their smoothness. For example, voiced speech sounds are more smooth than unvoiced ones. Smoothness is thus a informative characteristic of the signal.

A very simple way for measuring smoothness of a signal is to calculate the number of zero-crossing within a segment of that signal. A voice signal oscillates slowly - for example, a 100 Hz signal will cross zero 100 per second - whereas an unvoiced fricative can have 3000 zero crossing per second. An implementation of the zero-crossing for a signal $x_h$ at window $k$ is

$$ZCR_k = \sum_{h=kM}^{kM+N} |\text{sign}(x_h) - \text{sign}(x_{h-1})| \,,$$

where $M$ is the step between analysis windows and $N$ the analysis window length.

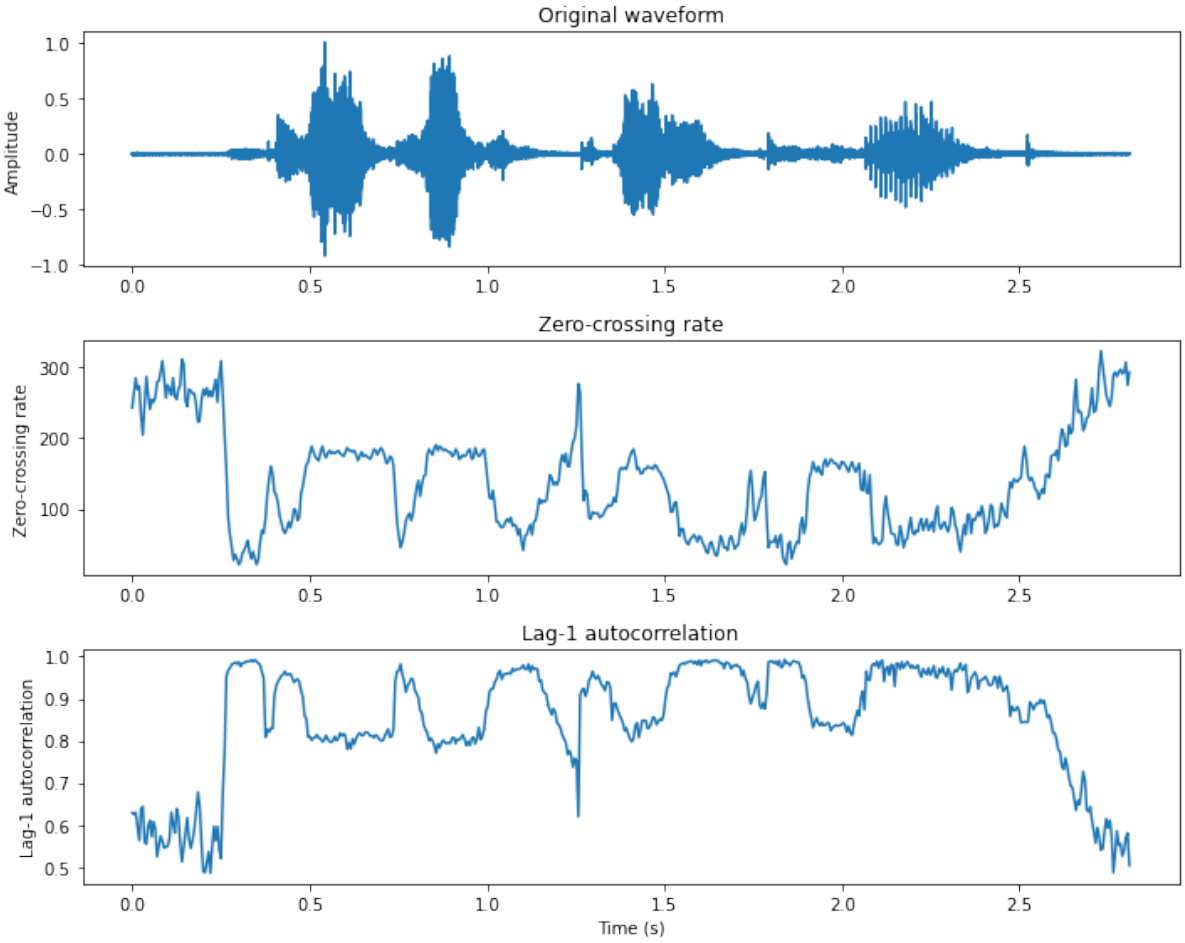


```
interactive(children=(FloatSlider(value=1.4048072562358276, description='position_s
↪', layout=Layout(width='760…
```

To calculate of the zero-crossing rate of a signal you need to compare the sign of each pair of consecutive samples. In other words, for a length $N$ signal you need $O(N)$ operations. Such calculations are also extremely simple to implement, which makes the zero-crossing rate an attractive measure for low-complexity applications. However, there are also many drawbacks with the zero-crossing rate:

- The number of zero-crossings in a segment is an integer number. A continuous-valued measure would allow more detailed analysis.
- Measure is applicable only on longer segments of the signal, since short segments might not have any or just a few zero crossings.
- To make the measure consistent, we must assume that the signal is zero-mean. You should therefore subtract the mean of each segment before calculating the zero-crossings rate.

An alternative to the zero-crossing rate is to calculate the *autocorrelation* at lag-1. It can be estimated also from short segments, it is continuous-valued and arithmetic complexity is also $O(N)$.



## 3.12 Deltas and Delta-deltas

In recognition tasks, such as phoneme recognition or voice activity detection, a classic input feature are the mel-frequency cepstral coefficients (MFCCs). They describes the instantaneous, spectral envelope shape of the speech signal. However, speech signals are time-variant signals and in a constant flux. Though we describe speech in linguistics as concatenated sequences of phonemes, the acoustical signal is more accurately described as a sequence of transitions between phonemes.

The same observation applies to other features of speech like the fundamental frequency (F0), which describes an instantaneous value. However, it is often more informative to analyse the overall shape of the F0 track, than the absolute value. For example emphasis in a sentence is often encoded with a distinct high-low contrast in F0 and questions have in many languages a characteristic low-high F0 contour.

A common method for extracting information about such transitions is to determine the first difference of signal features, known as the *delta* of a feature. Specifically, for a feature $f_k$, at time-instant $k$, the corresponding delta is defined as

$$\Delta_k = f_k - f_{k-1}.$$

The second difference, known as the delta-delta, is correspondingly

$$\Delta\Delta_k = \Delta_k - \Delta_{k-1}.$$

Common short-hand notations for the deltas and delta-deltas are, respectively, $\Delta$ and $\Delta\Delta$ -features. Features in a recognition engine are then typically appended by their $\Delta$ and $\Delta\Delta$ -features to triple the number of features with a very small computational overhead.

A trivial observation/interpretation of the delta and delta-delta features is that they approximate first and second derivatives of the signal. As estimates of the derivatives, they are not particularly accurate, but their simplicity probably makes up for that. The issue with accuracy is that differentiators tend to amplify white noise, whereas the desired signal remains unchanged. Consequently, the output is more noisy than the original signal. Differentiation is applied twice in the delta-delta feature such that issues with noise are also accumulated.

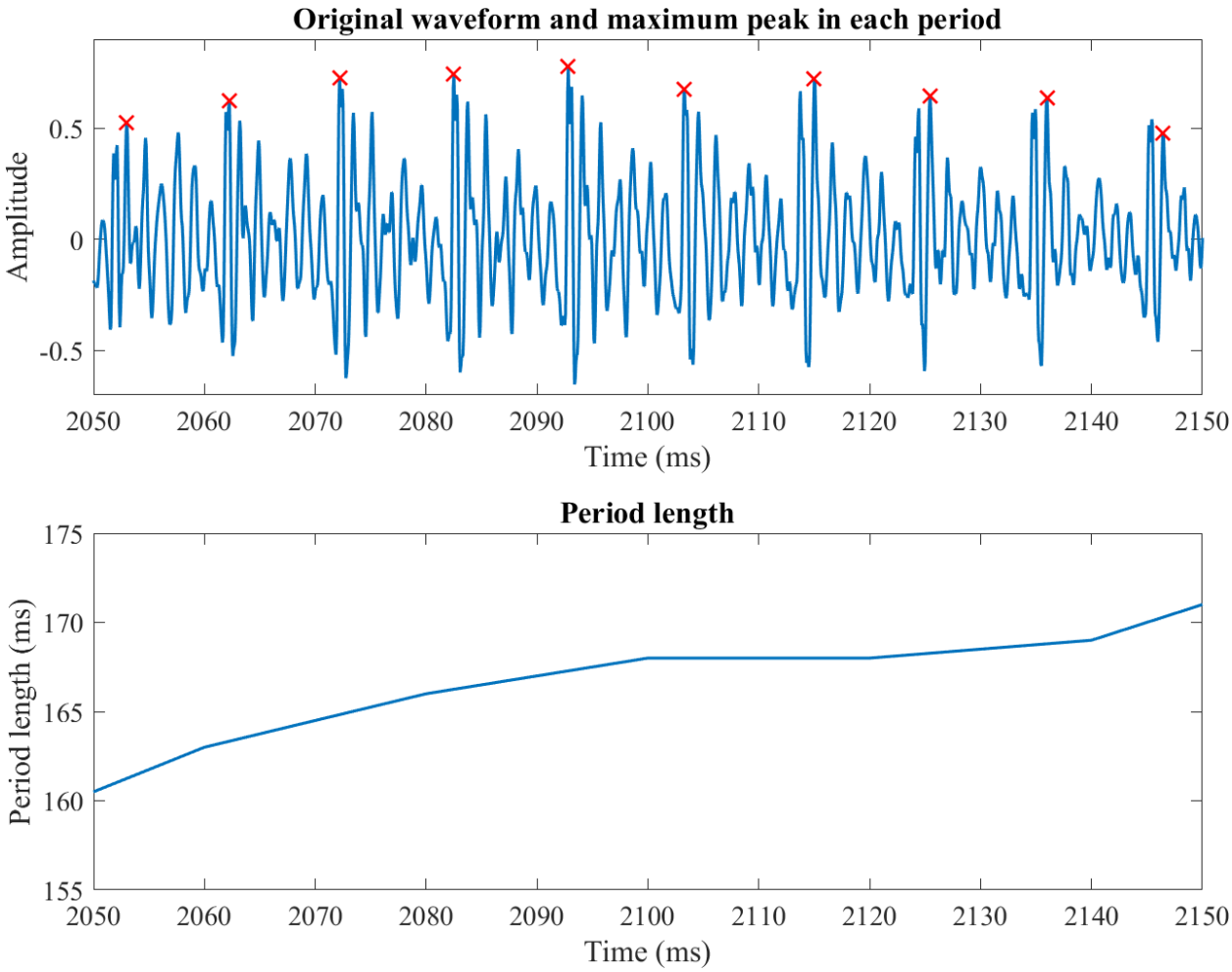
Note that the delta-features are linear transforms of the input features, such that if they are combined with a linear layer in a subsequent neural network, then in principle, the two consecutive linear layers are redundant. However, using delta-features can still provide a benefit in convergence.

In any case, delta- and delta-delta features are a classic component of machine learning algorithms. They are successful because they are very simple to calculate and provide often a clear benefit over the instantaneous features.

## 3.13 Pitch-Synchoronous Overlap-Add (PSOLA)

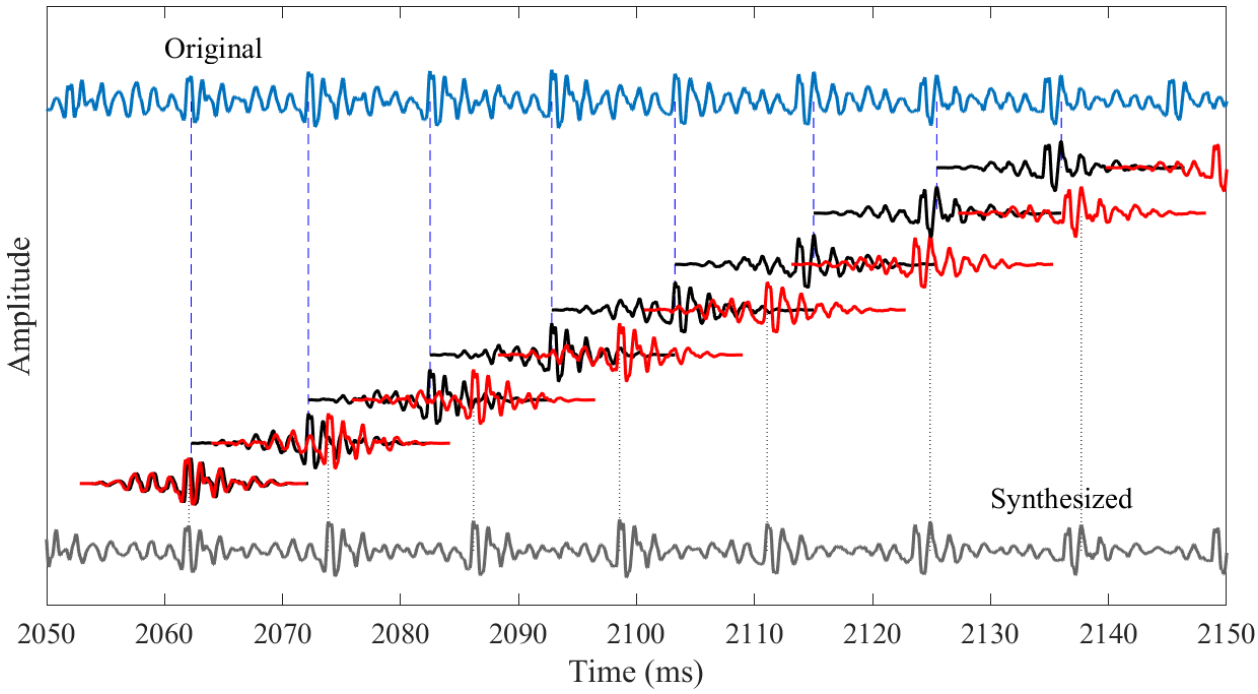
Many speech applications require the ability to modify the fundamental frequency. For a classic but marginal application, think of the auto-tune function often used in post-processing of singing voices. With such tools it is possible to change the fundamental frequency of a speaker's or singer's voice without changing the phoneme or timbre of the sound. One of the more popular tools developed for this purpose is pitch-synchronous overlap-add (PSOLA). Like the name suggests, it is closely related to the overlap-add method used in the *short-time Fourier transform* algorithm. It allows changing the pitch of a speech sound without modifying or with only minor influence on other characteristics of the signal, such as vowel-identity. In addition to auto-tune, an important application of PSOLA is speech synthesis, where we want to be able generate speech with any reasonable pitch contour. Voice conversion is another application, where the objective is to convert the speech of one person, such that it sounds like speech of another person.

Illustration of the PSOLA process; find period length and maximum peak in each period.




The basic idea of PSOLA is to decompose the signal into individual pitch-periods, such that we can move the pitch-periods to change the effective length of those periods. That is, the fundamental frequency of a signal is expressed as a periodic structure of the time-signal. If we cut the signal into segment corresponding to the length of such periodic structures, then we can shift their positions as desired and then add them back together, like in the overlap-add process (see STFT). Since short-term correlations in the signal are not changed, that is, signal inside the windows/segments is not changed, then the spectral envelope of the signal is not changed.

PSOLA analysis windowing, time-shift and synthesis windowing




To illustrate the principle, consider the following basic algorithm:

1. Estimate the fundamental frequency contour of a speech sample.
2. Find pitch periods of the speech sample, for example by identifying the largest peak in each period.
3. Extract windows of the speech signal covering *two* pitch periods. Apply a *windowing function* with perfect reconstruction. (Observe: Perfect reconstruction should apply for each period, so we construct half-length windows for each period. Conversely, the left and right parts of windows can be of different length.)
4. Shift windows to match the desired pitch-period length.

In the sound examples on the right, the pitch period lengths are adjusted by a fixed multiplier to increase or decrease the fundamental frequency. Observe that the implementation is not perfectly tuned such that the output sound has some audible distortions.

Sound examples with varying multiplier on distance between pitch periods

```
Multiplier 0.6
```

```
<IPython.lib.display.Audio object>
```

```
Multiplier 0.8
```

```
<IPython.lib.display.Audio object>
```

```
Multiplier 0.9
```

```
<IPython.lib.display.Audio object>
```

```
Multiplier 1.0 (original)
```

```
<IPython.lib.display.Audio object>
```

```
Multiplier 1.1
```

```
<IPython.lib.display.Audio object>
```

```
Multiplier 1.2
```

```
<IPython.lib.display.Audio object>
```

```
Multiplier 1.4
```

```
<IPython.lib.display.Audio object>
```

## 3.14 Jitter and shimmer

The speech production system is not a rigid, mechanical machine, but composed of an assortment of soft-tissue components. Therefore, although parts of a speech signal might seem stationary, there are always small fluctuations in it, as vocal fold oscillation is not exactly periodic. Variations in signal frequency and amplitude are called jitter and shimmer, respectively. Jitter and shimmer are acoustic characteristics of voice signals, and they are caused by irregular vocal fold vibration. They are perceived as roughness, breathiness, or hoarseness in a speaker's voice. All natural speech contains some level of jitter and shimmer, but measuring them is a common way to detect voice pathologies. Personal habits such as smoking or alcohol consumption might increase the level of jitter and shimmer in voice. However, many other factors can have an effect as well, such as loudness of voice, language, or gender. As jitter and shimmer represent individual voice characteristics that humans might use to recognize familiar voices, these measures could even be useful for speaker recognition systems.

There are several different ways to measure jitter and shimmer. For instance, when detecting voice disorders, they are measured as percentages of the average period, where values above certain thresholds are potentially related to pathological voices. Jitter and shimmer are most clearly detected from long, sustained vowels.

A commonly used jitter value is the absolute jitter. This measure expresses the average absolute difference between consecutive periods.

$$Jitter(absolute) = \frac{1}{N-1}\sum_{i=1}^{N-1}\|T_i - T_{i+1}\|$$

where Ti are the extracted F0 period lengths and N is the number of extracted F0 periods.

When this is divided by the average period, another common measure, relative jitter, is obtained.

$$Jitter(relative) = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}\|T_i - T_{i+1}\|}{\frac{1}{N}\sum_{i=1}^{N} T_i}$$

where Ti are the extracted F0 period lengths and N is the number of extracted F0 periods.

A commonly used shimmer value, here Shimmer(dB), expresses the average absolute base-10 logarithm of the difference between the amplitudes of consecutive periods multiplied by 20.

$$Shimmer(dB) = \frac{1}{N-1}\sum_{i=1}^{N-1}\|20\log(A_{i+1}/A_i)\|$$

where Ai are the extracted peak-to-peak amplitude data and N is the number of extracted fundamental frequency periods.

Relative shimmer expresses the average absolute difference between the amplitudes of consecutive periods divided by the average amplitude.

$$Shimmer(relative) = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}\|A_i - A_{i+1}\|}{\frac{1}{N}\sum_{i=1}^{N} A_i}$$

where Ai are the extracted peak-to-peak amplitude data and N is the number of extracted fundamental frequency periods.

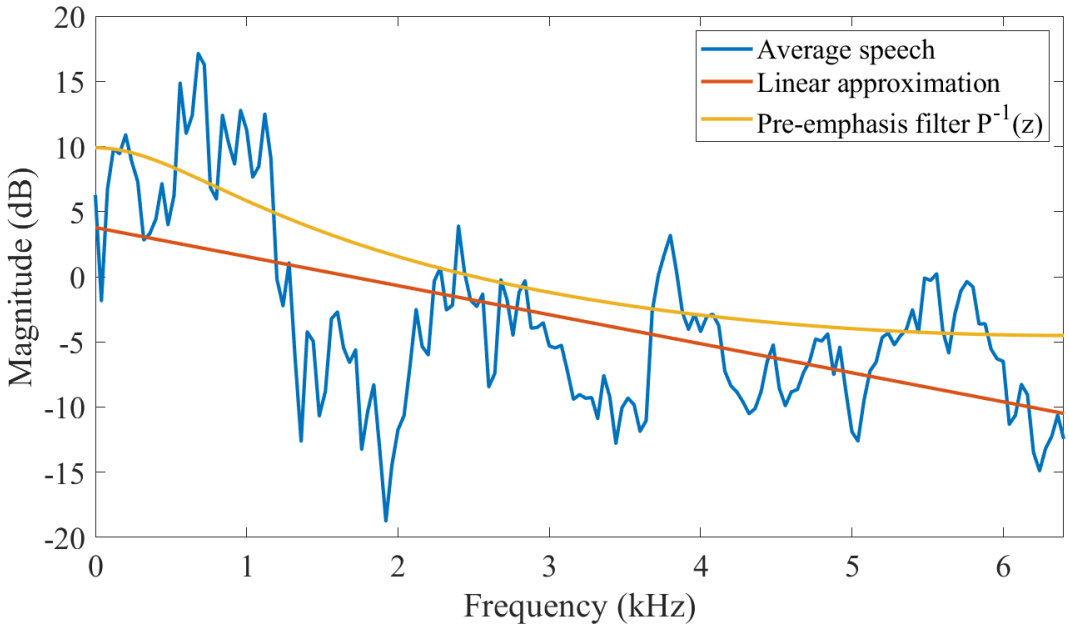
CHAPTER

# FOUR

# PRE-PROCESSING

1. *Pre-emphasis*
2. Noise gate (Wikipedia)
3. Dynamic Range Compression (Wikipedia)
4. Voice activity detection (VAD)
5. Vocal tract length normalization
6. *Speech enhancement*

## 4.1 Pre-emphasis



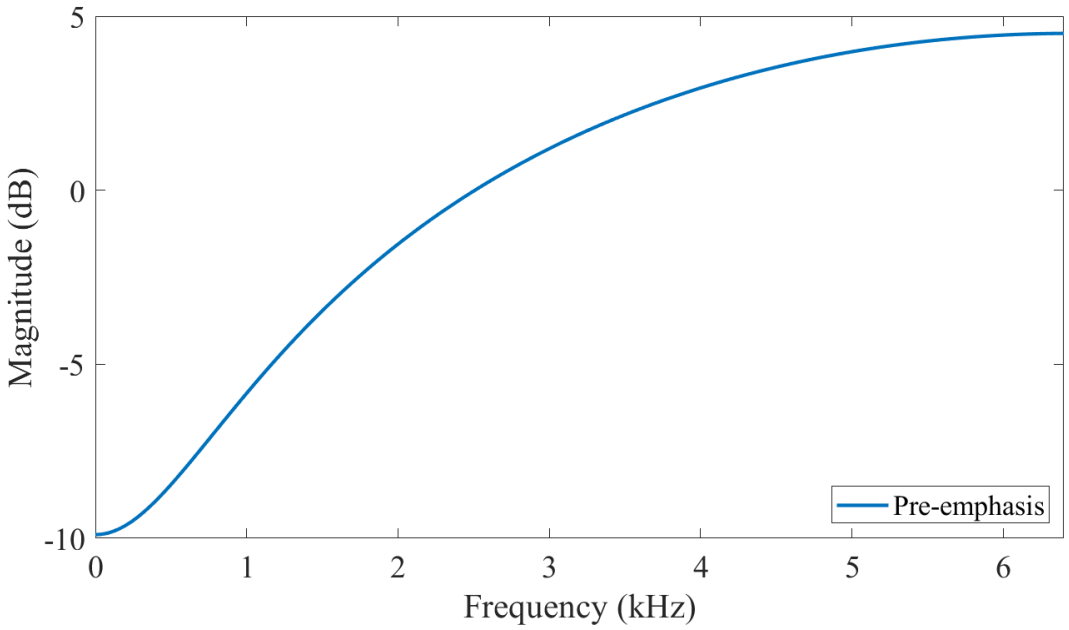

The figure above illustrates the average magnitude spectrum of a speech signal. We observe that a majority of the energy is concentrated in the lower end of the spectrum. In fact, as a linear approximation, we see that in this particular example, the energy drops at a rate of 2.2 dB/kHz. The exact rate of decrease varies for each speaker and depending on several factors. A safe and often used assumption is that energy drops at roughly 2 dB/kHz.

This rapid reduction in energy leads to practical problems in implementations. For example, if we would implement a discrete Fourier transform with fixed-point arithmetic, then the accuracy would be very different in different parts of the

spectrum. Typically the spectrum at 6kHz is 15dB lower than at 0Hz. On a linear scale 15dB corresponds to a factor of 6. In other words, on a 16-bit CPU, if we use the full range of a signed 15-bit representation for the lowest frequencies, than we use effectively only 12-bit range for frequency components at 6kHz.

A common pre-processing tool used to compensate for the average spectral shape is *pre-emphasis*, which emphasises higher frequencies. Typically, pre-emphasis is applied as a time-domain FIR filter with one free parameter, for example, in speech coding at a sampling rate of 8kHz or 12.8kHz, we use the pre-emphasis filter $P(z) = 1 - 0.68z^{-1}$ [Bäckström *et al.*, 2017]. The spectrum of this filter is illustrated below. After applying the filter, the spectrum is more flat and we can apply fixed-point arithmetic with a lower accuracy and thus better optimize CPU consumption.

There are numerous different ways of tuning pre-emphasis. Firstly, though the average spectrum is decaying, unvoiced fricatives have typically *more* energy at the high frequencies. Excessive pre-emphasis would therefore cause problems for fricatives. Pre-emphasis also has an effect on both perceptual and statistical modelling as well as estimation of linear predictive models. The best amount of pre-emphasis is therefore very much dependent on the application and implementation details.




## 4.2 References

CHAPTER

FIVE

# MODELLING TOOLS IN SPEECH PROCESSING

1. *Source and perceptual modelling*
2. *Linear regression*
3. *Sub-space models*
4. Vector quantization (VQ)
5. Gaussian mixture model (GMM)
6. *Neural networks*
7. *Non-negative Matrix and Tensor Factorization*
8. Hidden Markov Models (Wikipedia)

## 5.1 Source modelling and perceptual modelling

Speech processing applications use predominantly two types of modelling, models related to perception and to the source. They are always separate models on a meta-level, where we describe the motivations why we use them. In practical applications, they however have often complicated interactions, such that it becomes difficult to separate them.

- *Source models* characterize the objective properties of a signal. A source model can for example describe the statistical distribution of speech signals and their characteristics. One such model would be a model of the fundamental frequency; voiced signals have a fundamental frequency and we can specify the range where fundamental frequencies of speech signals can lie. Other obvious characteristics of speech signals we can model include the intensity of speech and how it can change over time, as well as the spectral envelope, what shapes are possible and how they can change over time.
- *Perceptual models* characterize *what* human listeners *can hear* and *how* much they appreciate different qualities. Perceptual models thus try to predict how humans evaluate quality. To construct perceptual models we need to ask human listeners questions like "Can you hear distortions in this signal?" or "How much is this signal distorted?". Based on the responses of human listeners, we can then make an analysis algorithm, which predicts the answer based on an analysis of the input signal.

In other words, source models explain objectively "*what the world is like*", where as perceptual models are evaluation models estimate subjective preference, "*how good the world is*".

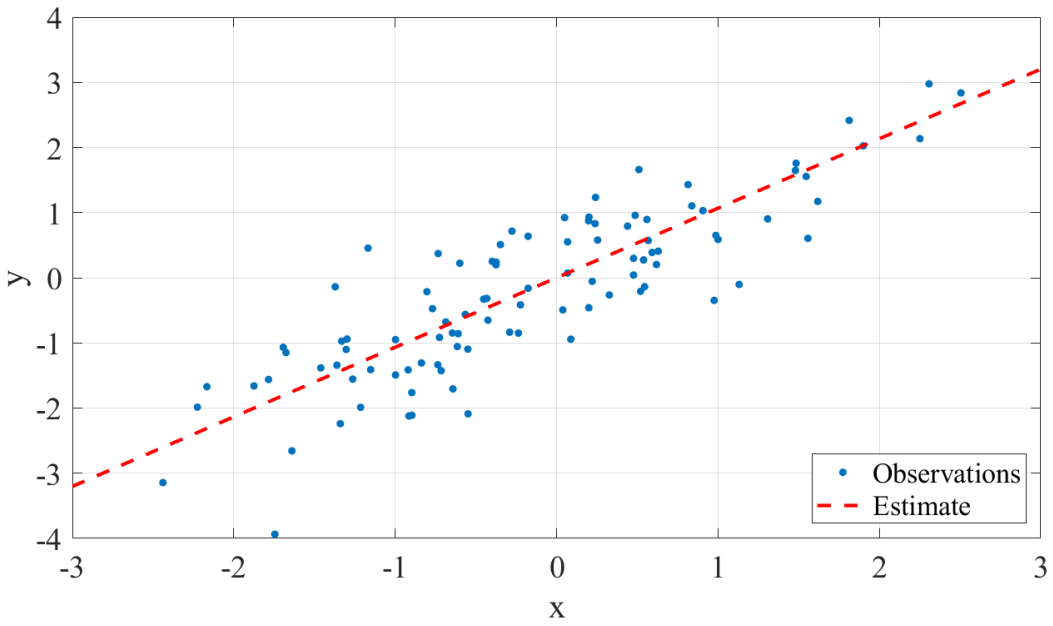
It is not however always so clear which model is active in which situation. For example, consider a speech recognizer, to which we feed speech with a loud background noise. We must decide whether the speech recognizer would evaluate speech as a human would evaluate, or whether we want to recognize speech as best we can. It is potentially possible that the speech recognizer could tolerate noise better than humans, such that it recognizes speech also when a human would fail. The model is then objectively evaluating speech content. However, if we want for example that a humanoid robot behaves like a human, then it should not understand speech in situations where a human would not.

As another example, consider a noise attenuation task, where our objective is to restore the original speech from a corrupted sample. We can then remove noise and only evaluate objectively how close we are to the original. This is an unambiguous objetive task, where perception plays no role. More similar to the original is better. However, an alternative approach is to consider all possible sounds $y$, and given a noisy observation $v$, compute the likelihood of all possible inputs. Suppose then that our output is $x$, we can then compute the perceptual distortion between all possible inputs $d(x, y)$, and assign weighting to them according to their likelihoods $P(y|v)$. Finally, we can minimize the expected distortion min $E[d(x, y)|P(y|v)]$. We thus take into account all possible true inputs, calculate the perceptual distortion between the true inputs and the output, and weight them according to how likely they are. Now, for the same task as before, we have a perceptual criterion with which we can choose the best output. Clearly the latter is more complicated, but it is also better motivated, so we have to choose which one we want to use. The first one is based on source modelling only (model of speech and noise), while the latter is a combination of source (likelihood of different speech signals, given a noisy observation) and perceptual modelling (perceptual distortion measure).

## 5.2 Linear regression

### 5.2.1 Problem definition

In speech processing and elsewhere, a frequently appearing task is to make a prediction of an unknown vector $y$ from available observation vectors $x$. Specifically, we want to have an estimate $\hat{y} = f(x)$ such that $\hat{y} \approx y$. In particular, we will focus on *linear estimates* where $\hat{y} = f(x) := A^T x$, and where $A$ is a matrix of parameters. The figure on the right illustrates a linear model, where the input sample pairs *(x,y)* are modelled by a linear model $\hat{y} \approx ax$.



### 5.2.2 The minimum mean square estimate (MMSE)

Suppose we want to minimise the squared error of our estimate on average. The estimation error is $e = y - \hat{y}$ and the squared error is the *L_{2}*-norm of the error, that is, $\|e\|^2 = e^T e$ and its mean can be written as the expectation $E\left[\|e\|^2\right] = E\left[\|y - \hat{y}\|^2\right] = E\left[\|y - A^T x\|^2\right]$. Formally, the minimum mean square problem can then be written as

$$\min_A E\left[\|y - A^T x\|^2\right].$$

This can in generally not be directly implemented because we have the abstract expectation-operation in the middle.

*(Advanced derivation begins)* To get a computational model, first note that the error expectation can be written in terms of the mean of a sample of vector *e_{k}* as

$$E\left[\|e\|^2\right] \approx \frac{1}{N}\sum_{k=1}^{N}\|e_k\|^2 = \frac{1}{N}\text{tr}(E^T E),$$

where $E = [e_1, e_2, \dots, e_N]$ and *tr()* is the matrix trace. To minimize the error energy expectation, we can then set its derivative to zero

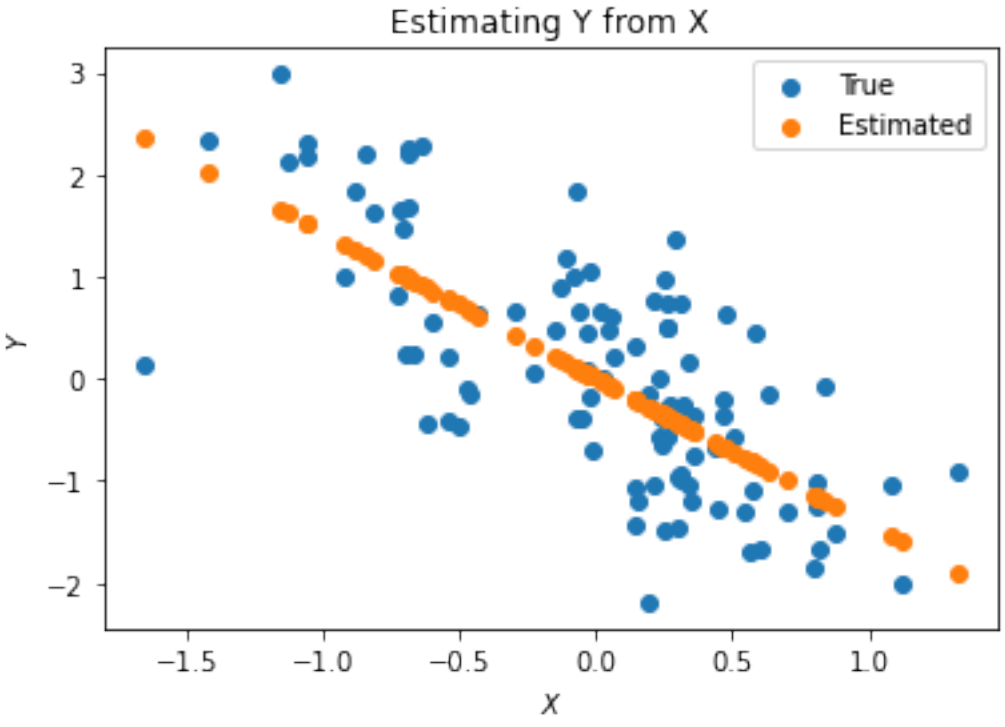
$$0 = \frac{\partial}{\partial A}\frac{1}{N}\text{tr}(E^T E) = \frac{1}{N}\frac{\partial}{\partial A}\text{tr}((Y - A^T X)^T (Y - A^T X)) = \frac{1}{N}(Y - A^T X)X^T$$

where the observation matrix is $X = [x_1, x_2, \dots, x_N]$ and the desired output matrix is $Y = [y_1, y_2, \dots, y_N]$. *(End of advanced derivation)*

It follows that the optimal weight matrix *A* can be solved as

$$\boxed{A = (XX^T)^{-1}XY^T = X^\dagger Y^T},$$

where the superscript † denotes the Moore-Penrose pseudo-inverse.



### Estimates with a mean parameter

Suppose that instead of an estimate $\hat{y} = A^T x$ , we want to include a mean vector in the estimate as $\hat{y} = A^T x + \mu$ . While it is possible to derive all of the above equations for this modified model, it is easier to rewrite the model into a similar form as above with

$$\hat{y} = A^T x + \mu = \begin{bmatrix} \mu^T \\ A^T \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} := A'^T x'.$$

That is, we can extend *x* by a single 1, (the observation *X* similarly with a row of constant 1s), and extend *A* to include the mean vector. With this modifications, the above Moore-Penrose pseudo-inverse can again be used to solve the modified model.

### Estimates with linear equality constraints

*(Advanced derivation begins)*

In practical situations we often have also linear constraints, such as $C^T A = B$ , which is equivalent with $C^T A - B = 0$. The modified programming task is then

$$\min_A E\left[\left\|y - A^T x\right\|^2\right] \quad \text{such that} \quad C^T A - B = 0.$$

For simplicity, let us consider only scalar estimation, where instead of vector *y*, as well as matrices *A, B* and *C*, respectively, we have scalar $\theta$ as well as vector *a, b* and *c *and the optimization problem is

$$\min_a E\left[\left\|\theta - a^T x\right\|^2\right] \quad \text{such that} \quad c^T a - b = 0.$$

Such constraints can be included into the objective function using the method of Lagrange multipliers such that the modified objective function is

$$\eta(a, g) = E\left[ \left|\theta - a^T x\right|^2 - 2\left[g^T\left(c^T a - b\right)\right]\right].$$

A heuristic explanation of this objective function is based on the fact the *g* is a free parameter. Since its value can be anything, then $c^T a - b$ must be zero, because otherwise the output value of the objective function could be anything. That is, when optimizing with respect to *a*, we find the minimum of the mean square error, while simultaneously satisfying the constraint.

The objective function can further be rewritten as

$$\begin{aligned}
\eta(a, g) &= E\left[\left(\theta - a^T x\right)\left(\theta - a^T x\right)^T\right] - 2\gamma\left(c^T a - \beta\right) \\
&= E\left[\theta^2 - 2\theta x^T a + a^T x x^T a\right] - 2\gamma\left(c^T a - \beta\right) \\
&= \sigma_\theta^2 + a^T R_x^T a - 2\gamma\left(c^T a - \beta\right) \\
&= \begin{bmatrix} a^T & \gamma \end{bmatrix} \begin{bmatrix} R_x & -c \\ -c^T & 0 \end{bmatrix} \begin{bmatrix} a \\ \gamma \end{bmatrix} - 2\begin{bmatrix} 0 & \beta^T \end{bmatrix} \begin{bmatrix} a \\ \gamma \end{bmatrix} + \sigma_\theta^2 \\
&= \begin{bmatrix} a^T & \gamma - \beta \end{bmatrix} \begin{bmatrix} R_x & -c \\ -c^T & 0 \end{bmatrix} \begin{bmatrix} a \\ \gamma - \beta \end{bmatrix} + \text{constant} \\
&:= (a' - \mu)^T R' (a' - \mu) + \text{constant}.
\end{aligned}$$

where $R_x = E[xx^T]$ and "*constant*" refers to a constant which does not depend on *a* or $\gamma$.

In other words, with a straightforward approach, equality constraints can in general also be merged into a quadratic form. We can therefore reduce constrained problems to unconstrained problems which can be easily solved.

*Inequality* constraints can be reduced to quadratic forms with similar steps as follows. Suppose we have a task min $f(x)$ such that $x \geq b$. We would then first solve the unconstrained problem min $f(x)$ and check whether the constraint is satisfied. If not, then the inequality constraint is "active", that is, we must have $x=b$. We can then rewrite the inequality constraint as an equality constraint and solve the problem as above.

In the case that we have multiple constraints over multiple dimensions, we can keep merging them one by one until we have an entire unconstrained programming problem.

*(Advanced derivation ends)*

### 5.2.3 Some applications

- Linear prediction in modelling of the spectral envelope of speech
- Noise attenuation with Wiener filtering

### 5.2.4 Discussion

Linear regression is just about the simplest thing you can do to model data. If that works then it's perfect! Especially for estimating low-dimensional from high-dimensional data, linear estimates can be very useful. In any case, it is always a good approach to start modelling with the simplest possible model, which usually is a linear model. If nothing else, that gives a good baseline. The first figure on this page demonstrates a case where a linear model does do a decent job at modelling the data.

Naturally there are plenty of situations where linear models are insufficient, such as when the data

- follows non-linear relationships
- has discontinuities or when the data
- contains multiple classes with different properties.

Moreover, in many cases we are not interested in modelling the average signal, but to recreate a signal which contains all the complexity of the original signal. Say, if we want to synthesize speech, then "average speech" can sound dull. Instead, we would like to reproduce all the colorfulness and expressiveness of a natural speaker. A model of the statistical distribution of the signal can then be more appropriate, such as the Gaussian mixture model (GMM).

Another related class of models are *Sub-space models*, where the input signal is modeled in a lower-dimensional space such that dimensions related to background noise are cancelled and the desired speech signal is retained.

## 5.3 Sub-space models

In many cases, we can assume that signals are low-dimensional in the sense that a high-dimensional observation $y \in \mathbb{R}^{N \times 1}$ can be completely explained by a low-dimensional representation $x \in \mathbb{R}^{M \times 1}$ such that with a matrix $A \in \mathbb{R}^{N \times M}$ we have $y = Ax$ with $N\ M$. This signal thus spans only a $M$-dimensional *sub-space* of the whole $N$-dimensional space.

### 5.3.1 Application with known sub-space

This representation comes in handy for example if we assume that we have only a noisy observation of $y$. The desired signal lies in a sub-space, so all the other dimensions have only noise in them and we can remove them. We thus only need a mapping from the whole space to the sub-space. It turns out that such a mapping is a projection to the sub-space spanned by matrix $A$. In fact, the minimum mean square error (see *Linear regression*) solution is exactly the Moore-Penrose pseudo-inverse. However, the downside with this approach is that here the matrix A needs to be known in advance such that the pseudo-inverse can be formed.

### 5.3.2 Estimation of unknown sub-space

In practical cases it is rather unusual that we would have access to the matrix $A$, but instead, it must be estimated from available data. A typical approach is based on a singular value decomposition (SVD) or the eigenvalue decomposition. In short, we first estimate the covariance matrix of the signal and then decompose it into uncorrelated components with the singular value decomposition. Often, a small set of singular values make up most of the energy of the whole signal. Thus if we discard the smallest singular values, we do not loose much of the energy, but have a signal of a much lower dimensionality. The singular value decomposition thus takes the role of the sub-space mapping matrix $A$, and we can apply the model as described above.

### 5.3.3 Discussion

Sub-space models are theoretically appealing models, since their analysis is straightforward. In terms of speech signals, the difficulty lies in finding a representation which is actually low-rank. In other words, it is not immediately clear in which domain we can apply analysis such that speech signals can efficiently modelled by low-rank models.

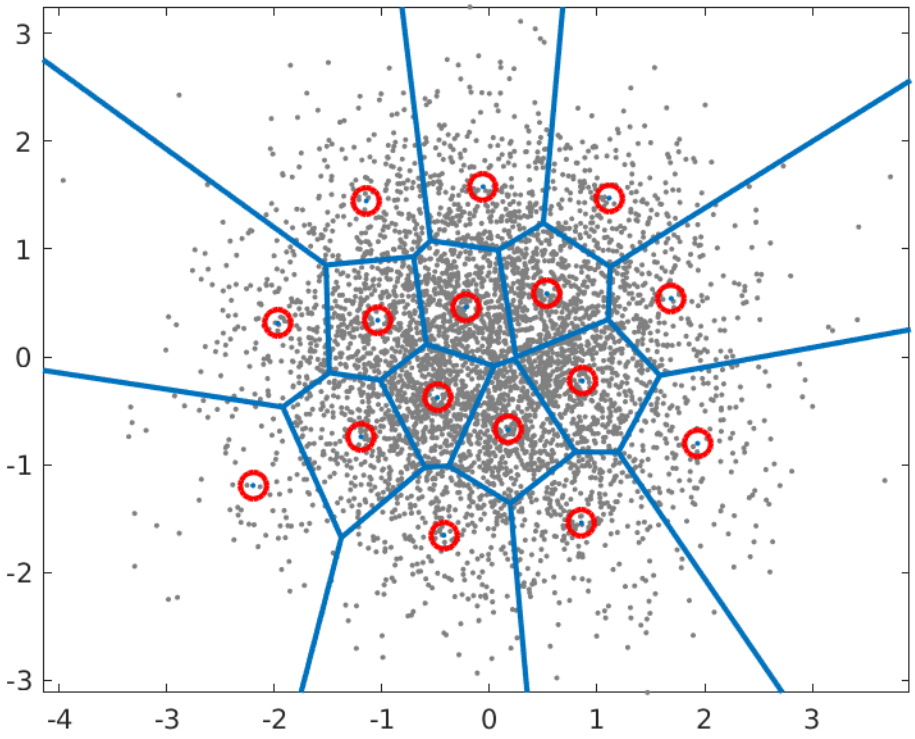
## 5.4 Vector quantization (VQ)

Suppose you have recorded sounds at different locations and want to categorize them into similar groups. In other words, you have a stochastic vector $x$ which you want to characterize with a simple description. For example, categories could correspond to office, street, hallway and cafeteria. A classic way for this task is to choose template vectors $c_k$, which represents a typical sound in each environment $k$. To categorize the sounds, you then find that template vector which is closest to your recording $x$. In mathematical notation, you search for a $k^*$ by

$$k^* = \arg\min_k \|x - c_k\|^2.$$

The above expression thus calculates the squared error between $x$ and each of the vectors $c_k$ and chooses the index $k$ of the vector with the smallest error. The vectors $c_k$ then represent a codebook and the vector $x$ is quantized to $c_{k^*}$. This is the basic idea behind *vector quantization,* which is also known as *k-means*.

A illustration of a simple vector codebook is shown on the right. The input data is a Gaussian distribution shown with grey dots and the codebook vectors $c_k$ with red circles. For each input vector we thus search for the nearest codebook vector and the borders of the regions where input vectors are assigned to a particular codebook vector are illustrated with blue lines. These regions are known as Voronoi regions and the blue lines are the decision-boundaries between codebook vectors.




Example of a codebook for a 2D Gaussian with 16 code vectors.

### 5.4.1 Metric for codebook quality

Suppose then that you have a large collection of vectors $x_h$, and you want to find out how well this codebook represents the input data. The expectation of the squared error is approximately the mean over your data, such that

$$E_h\left[\min_k \|x_h - c_k\|^2\right] \approx \frac{1}{N}\sum_{h=1}^{N} \min_k \|x_h - c_k\|^2,$$

where $E[]$ is the expectation operator and $N$ is the number of input vectors $x_h$. Above, we thus find the codebook vector which is closest to $x_h$, find its squared error and take the expectation over all possible inputs. This is approximately equal to the mean of those squared errors over a set of input vectors.

To find the best set of codebook vectors $c_k$, we then need to minimize the mean squared error as

$$\{c_k^*\} := \arg\min_{\{c_k\}} E_h\left[\min_k \|x_h - c_k\|^2\right]$$

or more specifically, for a dataset as

$$\{c_k^*\} := \arg\min_{\{c_k\}} \sum_{h=1}^{N} \min_k \|x_h - c_k\|^2.$$

Unfortunately we do not have an analytic solution for this optimization problem, but have to use numerical, iterative methods.

## 5.4.2 Codebook optimization

### Expectation maximization (EM)

Classical methods for finding the best codebook are derivatives of expectation maximization (EM), which is based on two alternating steps:

*Expectation Maximation (EM)* algorithm:

1. For every vector $x_h$ in a large database, find the best codebook vector $c_k$.
2. For every codebook vector $c_k$;
    1. Find all vectors $x_h$ assigned to that codevector.
    2. Calculate mean of those vectors.
    3. Assign the mean as a new value for the codevector.
3. If converged then stop, otherwise go to 1.

This algorithm is guaranteed to give a codebook at every step which is *not worse* than the previous codebook. That is, at each iteration will improve until it finds a local minimum, where it stops changing. The reason is that each step in the iteration finds a partial best-solution. In the first step, we find the best matching codebook vectors for each data vectors $x_h$. In the second step, we find the within-category mean. That is, the new mean is more accurate than the previous codevector in that it reduces the average squared error. If the mean is equal to the previous codevector, then there is no improvement.

As noted above, this algorithm is the basis to most vector quantization codebook optimization algorithms. There are a multiple reasons why this simple algorithm is usually not sufficient alone. Most importantly, the above algorithm is slow to converge to a stable solution *and* it often finds a local minimum instead of a global minimum.

To improve performance, we can apply several heuristic approaches. For example, we can start with a small codebook $\{c_k\}_{k=1}^{K}$ of $K$ elements and optimize it with the EM algorithm. We then split the codebook into two, offset by a small delta $d$, such that $\|d\| < \epsilon$ and make the new codebook $\{\hat{c}_k\}_{k=1}^{2K} := \{c_k,\, c_k + d\}_{k=1}^{K}$ of $2K$ elements. We then rerun the EM algorithm on the new codebook. The codebook thus doubles in size at every iteration and we continue until we have the desired codebook size.

The advantage of this approach is that it focuses attention to the big bulk of datapoints $x_k$, and ignores outliers. The outcome is then expected to be more stable and the likelihood of converging to a local minimum is smaller. The downside is that with this approach it is then more difficult to find small separated islands. That is, because the initial codebook is near the center of the whole mass of datapoints, adding a small delta to the codebook vectors keeps the new codevectors near the center-of-mass.

Conversely, we can start with a large codebook, say treat the whole input database $x_k$ as a codebook. We can then iteratively merge pairs of points which are close to each other, until the codebook is reduced to the desired size. Needless to say, this will be a slow process if the database is large, but will be very efficient in finding separated islands of points.

In any case, optimization of vector codebooks is a difficult task and we have no practical algorithms which would be guaranteed to find the global optimum. Like in many other machine learning problems, optimizing the codebook is very much about learning to know your data. You should first use one algorithm and then analyse the output to find out what can be improved, and keep repeating this optimization and analyse process until the output is sufficiently good.

#### Optimization with machine learning platforms

A modern approach to modelling is *machine learning*, where complex phenomena are modelled with neural networks. Typically they are trained with gradient-descent type methods, where parameters are iteratively nudged towards the minimum, by following the steepest gradient. Since such gradients can be automatically derived on machine learning platforms (using the chain rule), they can be applied on very complex models. Consequently, they have become very popular and succesful.

The same type of training can be readily applied to vector quantizers as well. However, there is a practical problem with this approach. Estimation of the gradients of the parameters with the chain-rule requires that *all* intermediate gradients are non-zero. Quantizers are however piece-wise constant such that their gradients are uniformly zero, thus disabling the chain rule and gradient descent for all parameters which lie behind the quantizer in the computational graph. A practical solution is known as pass-through, where gradients are passed unchanged through the quantizer. This approximation is simple to implement and provides often adequate performance.

While this approach converges slower than the EM-algorithm, it is often beneficial since it allows optimization of an entire system to a single goal. That is, if we have several modules in a system, it is beneficial if their joint interaction is taken into account in the optimization. If modules are optimized independently, it is difficult to anticpate all interactions and the system performance can remain far from optimal.

### 5.4.3 Algorithmic complexity

Vector quantization is manageable for relatively small codebooks of, say, $K = 32$ codevectors. That corresponds to 5 bits of information. For many applications, that does not give sufficient accuracy - the mean squared error is too large. For example, the linear predictive models in speech coding could be quantized with 30 bits, which corresponds to $K = 2^{30} \approx 10^9$ codevectors. To find the best codevector for a vector $x$ of length $N = 16$, we would then need to calculate the distance between every codebook vector and $x$, which amounts to approximately $16 \times 10^9 = 1.6 \times 10^{10}$ operations. That is infeasible in on-line applications on mobile devices. Instead, we need to find a simpler method which retains the best aspects of the algorithm, but reduces algorithmic complexity.

A heuristic approach is to use successive codebooks, where at each iteration, we quantize the error of the last iteration. That is, let's say that on the first iteration we have 8 bits, corresponding to a codebook $c_k$ of $K = 256$ vectors. We find the best matching codevector $c_{k^*}$ and calculate the residual $x' := x - c_{k^*}$ . In the second stage, we would then find the best matching vector for $x'$ from a second codebook $c'_k$. We can add as many layers of codebooks as we want until the desired number of bits has been consumed. This approach is known as a *multi-stage vector quantizer*.

Where ordinary vector quantization can find the optimal solution, split vector quantization generally does not give a global optimum. It does give good solutions, though, but with an algorithmic complexity which very much lower than ordinary vector quantization. For example, in the above example of 30 bits, we could assign three consecutive layers of codebooks with 10 bits / $K = 1024$ each, such that the overall complexity is $3 \times 16 \times 2^{10} \approx 5 \times 10^4$, which gives an improvement with a factor of $3.5 \times 10^5$. Given that the reduction in accuracy is manageable, this is a major improvement in complexity.

### 5.4.4 Applications

Probably the most important application where vector quantization is used in speech processing, is speech coding with Code-excited linear prediction (CELP), where

- linear predictive coefficients (LPC) are transformed to line spectral frequencies (LSFs), which are often encoded with multi-stage vector quantizers.
- gains (signal energy) of the residual and long term prediction are jointly encoded with a single stage vector quantizer.

Other typical applications include

- In optimization of Gaussian mixture models (GMMs), it is useful to use vector quantization to find a first-guess of the means of each mixture.

### 5.4.5 Discussion

The benefit of vector quantization is that it is a simple algorithm which gives high accuracy. In fact, for quantizing complicated data, vector quantization is (in theory) optimal in fixed-rate coding applications. It is simple in the sense that an experienced engineer can implement it in a matter of hours. Downsides with vector quantization include

- Complexity; for accurate quantization you need prohibitively large codebooks. The method therefore does not scale up nicely to big problems.
- Difficult optimization;
  - Training data; The amount of data needed to optimize a vector codebook is large. Each codebook vector must be assigned to a large number of data vectors, such that calculation of the mean (in the EM algorithm) is meaningful.
  - Convergence; we have no assurance that optimization algorithms find the global optimum and we have no assurance that local minima are "good enough".
- Lack of flexibility; the codebook has a fixed size. If we would like to use codebooks of different sizes, for example, if we want to transmit data with a variable bit-rate, then we have to optimize and store a large codebook for *every possible bitrate*.
- Blindness to inherent structures; this model describes data with a codebook, without any deeper understanding of what the data looks like within each category. For example, say we have two classes, speech and non-speech. Even if speech is very flexible, the non-speech class is much, much larger. Speech is a very small subset of all possible sounds. Therefore, the within-class variance will be much larger in the non-speech class. Consequently, the accuracy in the non-speech class would be much lower.As a consequence, we would be tempted to increase the number of codevectors such that we get uniform accuracy in both classes. But then we loose the correspondence between codevectors and natural descriptions of the signal.

## 5.5 Gaussian mixture model (GMM)

### 5.5.1 Motivation

Where approaches such as *linear regression* and *sub-space models* are based on reducing the dimensionality of a signal to capture the essential information in a signal, in many cases we want to model the full range of possible signals. For that purpose we can design models the *statistical distribution* of the signal. For example, it is possible to model a signal as a Gaussian process, where every observation has a (multivariate) Gaussian (normal) distribution.

Speech signals however feature much more structure than simple Gaussian processes. For example voiced signals are very different from unvoiced signals, and within both voiced and unvoiced signals we have a multitude of distinct groups of utterances whose statistical characteristics are clearly different. Modelling them all with a Gaussian process would ignore such structures and the model would therefore be inefficient.

*Mixture models* is a type of models, where we assume that the signal under study consists of several distinct classes, where each class has its own unique statistical model. That is, for example the statistics of voiced sounds is clearly different from those of unvoiced sounds. We model each class with its own distribution and their joint distribution is the weighted sum of the class distributions. The weights of each distribution correspond to the frequency with which they appear in the signal. So if unvoiced signals would in some hypothetical language constitute 30% of all speech sounds, then the weight of the unvoiced class would be 0.3.

The most typical mixture model structure uses Gaussian (normal) distributions for each of the classes, so that the whole model is known as a *Gaussian mixture model* (GMM). Depending on application the class-distributions can obviously take other forms than Gaussian, for example a Beta mixture model could be used if the individual classes follow the Beta distribution. In this document we however focus on Gaussian mixture models because it is most common among mixture models and demonstrates application in an accessible way.

### 5.5.2 Model definition

The multivariate normal distribution for a variable $x$ is defined as

$$f(x;\Sigma,\mu) = \frac{1}{\sqrt{(2\pi)^N \|\Sigma\|}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right],$$

where $\Sigma$ and $\mu$ are the covariance and mean of the process, respectively, with $N$ dimensions. In other words, this is the familiar Gaussian process for vectors $x$.

Suppose then that we have $K$ classes in the signal, where each class has its own covariance and mean $\Sigma_k$ and $\mu_k$. The *Gaussian mixture model* is then defined as

$$\boxed{f(x) = \sum_{k=1}^{K} \alpha_k f(x;\Sigma_k,\mu_k),}$$

where the weights $\alpha_k$ add up to unity $\sum_{k=1}^{K} \alpha_k = 1$.

### 5.5.3 Applications

- In recognition/classification applications, we can, for example, model a system which has two distinct states (like speech and noise) and train a GMM with mixture components matching those states. When receiving a microphone signal, we can then determine the likelihood of each mixture component and thus obtain the likelihood that the signal is speech or noise.
- In transmission applications, our objective is to model the signal such that we can transmit likely signals with a small amount of bits and unlikely signals with a large number of bits. If we train a GMM on a speech database, we can determine which signals are speech-like, such that those can be transmitted with a low number of bits.

## 5.6 Neural networks

### 5.6.1 Introduction

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model (function). Artificial neuron is a basic building block of every artificial neural network. Its design and functionalities are derived from observation of a biological neuron that is basic building block of biological neural networks (systems) which includes the brain, spinal cord and peripheral ganglia.

Computational modelling using neural networks was started in the 1940's and has gone through several waves of innovation and subsequent decline. The recent boom of deep neural networks (DNNs) - which roughly means that the network has more layers of non-linearities than previous models - happened probably due to the increase in available computational power and availability of large data-sets. It has fuelled a flurry of incremental innovation in all directions, and in many modelling tasks, deep neural networks currently give much better results than any competing method.

Despite its successes, application of DNNs in speech processing does not come without its fair share of problems. For example,

- Training DNNs for a specific task on a particular set of data often does not increase our understanding of the problem. It is a black box. How are we to know whether the model is reliable? A trained speech recognizer on a language A does not teach much about speech recognition for language B (though the process of designing a recognizers does teach us about languages).

- Training of DNNs is sensitive to the data on which it is trained on. Models can for example be susceptible to hidden biases, such that performance degrades for particular under-represented groups of people.
- A trained DNN is "a solution" to a particular problem, but it does not directly give us information about how good of a solution it is. For example, if a data-set represents a circle in the 2D-plane, then it is possible to accurately model that data-set with a neural network where the non-linearities are sigmoids. The neural network just has to be large enough and it can do the job. However, the network is then several orders of magnitude more complex than the equation of the circle. That is, though training of the network was successful, and the model is relatively accurate, it gives no indication if the complexity of the network is of similar scale as the complexity of the problem.

To combat such problems, a recent trend in model design has been to return to classical design paradigms, where models are based on thorough understanding of the problem. The parameters of such models are then trained using methods from machine learning.

### 5.6.2 Network structures

Neural networks are, in principle, built from simple building blocks, where the most common type of building block is

$$y = f(Ax + b)$$

where $x$ is an input vector, matrix $A$ and vector $b$ are constants, $f$ is a non-linear function such as the element-wise sigmoid, and $y$ is the output. This is often referred to as a *layer*. Layers can then be stacked after each other such that, for example, a three layer network would be

$$\begin{aligned} y_1 &= f(A_1 x + b_1) \\ y_2 &= f(A_2 y_1 + b_3) \\ y_{out} &= f(A_3 y_2 + b_3). \end{aligned}$$

#### Deep Neural Networks (DNNs)

Deep Neural Network (DNNs) are an artificial neural network (ANN) with multiple layers between the input and output layers. Many experts define deep neural networks as networks that have an input layer, an output layer and at least one hidden layer in between. Each layer performs specific types of sorting and ordering in a process that some refer to as "feature hierarchy." One of the key uses of these sophisticated neural networks is dealing with unlabeled or unstructured data. The phrase "deep learning" is also used to describe these deep neural networks, as deep learning represents a specific form of machine learning where technologies using aspects of artificial intelligence seek to classify and order information in ways that go beyond simple input/output protocols.

#### Basics of Neural Networks

Neurons: It forms the basic structure of a neural network. When we get the information, we process it and then we generate an output. Similarly, a neuron receives an input, processes it and generates an output which is either sent to other neurons for further processing or it is the final output.

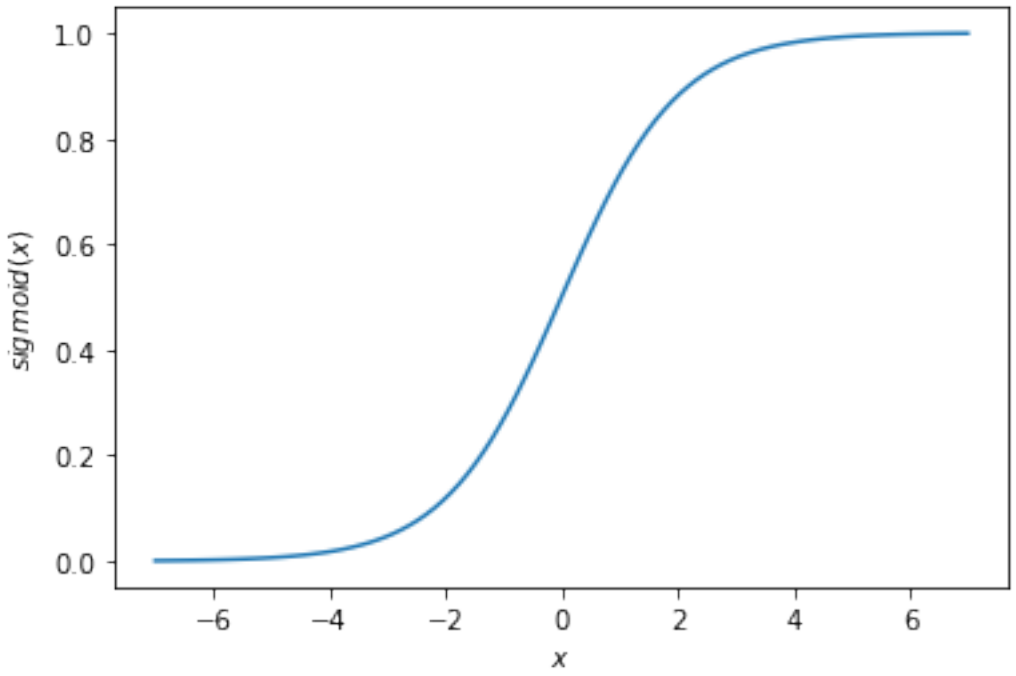
Weights: When an input enters a neuron, it is multiplied by a weight. Initially, the weights are initialized and they are updated during the model training process. When the training is over, the neural network assigns a higher weight value to the input it considers more important as compared to the ones which are considered less important.

Bias: In addition to the weights, another linear component is applied to the input, called as the bias. It is added to the result of weight multiplication to the input. The bias is basically added to change the range of the weight multiplied input.

### Activation Functions:

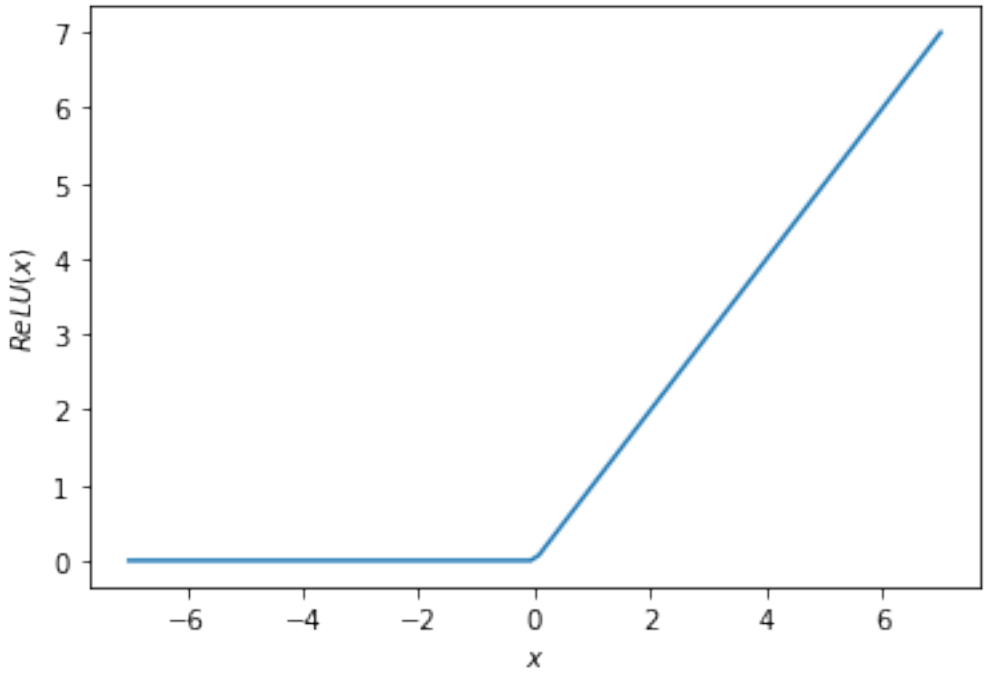
1. Sigmoid: It allows a reduction in extreme or atypical values in valid data without eliminating them: it converts independent variables of almost infinite range into simple probabilities between 0 and 1. Most of its output will be very close to the extremes of 0 or 1. $\$sigmoid(x) = \frac{1}{1+e^{-x}}\$$

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-7,7,100)
plt.plot(x, 1./(1. + np.exp(-x)))
plt.xlabel('$x$')
plt.ylabel('$sigmoid(x)$')
plt.show()
```




1. ReLU(Rectified Linear Unit): It has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. The operation of ReLU is closer to the way our biological neurons work. $\$ReLU(x) = \max(x, 0) = \begin{cases} x, & x > 0 \\ 0, & \text{otherwise.} \end{cases}\$$

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-7,7,100)
plt.plot(x, (x>0)*x)
plt.xlabel('$x$')
plt.ylabel('$ReLU(x)$')
plt.show()
```




1. Softmax: A modification of the regular max over a vector, such that it has a continuous derivative everywhere. That is, it maps the output to the range $[0, 1]$ and simultaenously ensures that the total sum is 1. The output of Softmax is therefore a probability distribution. $\$SoftMax(\mathbf{x_k}) = \frac{e^{x_k}}{\sum_{j=1}^{K} e^{x_j}}.\$$

1. Input / Output / Hidden Layer : The input layer receives the input and is the first layer of the network. The output layer is the one which generates the output and is the final layer of the network. The processing layers are the hidden layers within the network. These hidden layers are the ones which perform specific tasks on the incoming data and pass on the output generated by them to the next layer. The input and output layers are the ones visible to us, while are the intermediate layers are hidden.

2. MLP (Multi Layer perceptron): In the simplest network, we would have an input layer, a hidden layer and an output layer. Each layer has multiple neurons and all the neurons in each layer are connected to all the neurons in the next layer. These networks can also be called as fully connected networks.

3. Cost or loss function: When we train a network, its main objective is to to predict the output as close as possible to the actual value. Hence, the cost/loss function is used to measure this accuracy. The cost or loss function penalizes the network when it makes errors. The main objective while running the network is to increase the prediction accuracy and to reduce the error, thus minimizing the cost function.

4. Gradient Descent: It is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.

5. Learning Rate: The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process. The learning rate may be the most important hyperparameter when configuring neural network. Therefore, it is important to know how to investigate the effects of the learning rate on model performance and to build an intuition about the dynamics of the learning rate on model behavior.

6. Backpropagation: When we define a neural network, we assign random weights and bias values to our nodes. Once we have received the output for a single iteration, we can calculate the error of the network. This error is then fed back to the network along with the gradient of the cost function to update the weights of the network. These weights are then updated so that the errors in the subsequent iterations is reduced. This updating of weights using

the gradient of the cost function is known as back-propagation. In back-propagation the movement of the network is backwards, the error along with the gradient flows back from the out layer through the hidden layers and the weights are updated.

7. Batches: While training a neural network, instead of sending the entire input in one go, we divide in input into several chunks of equal size randomly. Training the data on batches makes the model more generalized as compared to the model built when the entire data set is fed to the network in one go.
8. Epochs: An epoch is is a single training iteration of all batches in both forward and back propagation. Thus, 1 epoch is a single forward and backward pass of the entire input data. Although it is highly likely that more number of epochs would show higher accuracy of the network, it would also take longer for the network to converge. You should also take into account that if the number of epochs are too high, the network might over-fit.
9. Dropout: It is a regularization technique to prevent over-fitting of the network. While training, a number of neurons in the hidden layer are randomly dropped.
10. Batch Normalization: It normalizes the input layer by adjusting and scaling the activations. For example, if we have features from 0 to 1 and some from 1 to 1000, we should normalize them to speed up learning.

Classification of Neural Networks

1. **Shallow neural network**: It has only one hidden layer between the input and output.
2. **Deep neural network**: it has more than one layer.

Types of Deep Learning Networks

- Feed forward neural networks: They are artificial neural networks where the connections between units do not form a cycle. Feedforward neural networks were the first type of artificial neural network invented and are simpler than their counterpart, recurrent neural networks (see below ). They are called feedforward because information only travels forward in the network. Firstly, it goes through the input nodes, then through the hidden nodes and finally through the output nodes.

### Convolutional neural networks

CNN was first proposed by [1]. It has first been successfully used by [2] for handwritten digit classification problem. It is currently the most popular neural network model being used for image classification problem. The advantages of CNNs over DNNs include CNNs are highly optimized for processing 2D and 3D images, and are effective to learn and extract abstractions of 2D features. In addition to these, CNNs have significantly fewer parameters than a fully connected network of similar size.

Figure 4.1. shows the overall architecture of CNNs. CNNs consists mainly of two parts: feature extractors and classifier. In the feature extraction module, each layer of the network receives the output from its immediate previous layer as its input and passes its output as the input to the next layer. The CNN architecture mainly consists of three types of layers: convolution, pooling, and classification.

Figure 4.1. An overall architecture of the Convolutional Neural Network.

The main layers in Convolutional Neural Networks are:

- Convolutional layer: A “filter” passes over the image, scanning a few pixels at a time and creating a feature map that predicts the class to which each feature belongs. Thus, in this layer, feature maps from previous layers are convolved with learnable kernels. The output of the kernels goes through a linear or non-linear activation function, such as sigmoid, hyperbolic tangent, Softmax, rectified linear, and identity functions) to form the output feature maps. Each of the output feature maps can be combined with more than one input feature map.
- Pooling layer: It reduces the amount of information in each feature obtained in the convolutional layer while maintaining the most important information (there are usually several rounds of convolution and pooling).

- Fully connected input layer (flatten): It takes the output of the previous layers, "flattens" them and turns them into a single vector that can be an input for the next stage.
- The first fully connected layer: It takes the inputs from the feature analysis and applies weights to predict the correct label.
- Fully connected output layer: It gives the final probabilities for each label.

Higher-level features are derived from features propagated from lower level layers. As the features propagate to the highest layer or level, the dimensions of features are reduced based on the size of the kernel for the convolution and pooling operations respectively. However, the number of feature maps usually increase for representing better features of the input images for ensuring classification accuracy. The output of the last layer of the CNN is used as the input to a fully connected network. Feed-forward neural networks are used as the classification layer since they provide better performance.

Figure 4.2 Popular CNN architectures. The Figure is taken from https://www.aismartz.com/blog/cnn-architectures/.

Applications of CNNs:

- Image Processing and Computer Vision: CNNs have been successfully used in different image classification tasks [7–11].
- Speech Processing: CNNs have been successfully used in different speech processing applications such as speech enhancement [8] and audio tagging [9].
- Medical Imaging: CNNs have also been widely used in different medical image processing including classification, detection, and segmentation tasks [10].

Training Techniques

1. Sub-Sampling Layer or Pooling Layer: Two different techniques have been used for the implementation of deep networks in the sub-sampling or pooling layer: average and max-pooling. While the average Pooling calculate the average value for each patch on the feature map, the max pooling calculate the maximum value for each patch of the feature map.
2. Padding: It adds extra layer of zeros across the images so that the output image has the same size as the input.
3. Data Augmentation: It is the addition of new data derived from the given data. This might prove to be beneficial for prediction. It includes rotation, shearing, zooming, cCropping, flipping and changing the brightness level.

Note that the performance of CNNs depends heavily on multiple hyperparameters: number of layers, number of feature maps in each layer, the use of dropouts, batch normalization, etc. Thus, it's important that you should first fine-tune the model hyperparameters by conducting lots of experiments. Once you find the right set of hyperparameters, you need to train the model for a number of epochs.

### Recurrent neural networks (RNNs)

Topics to be covered:

- Basics of RNNs
- Vanishing and exploding gradient problem
- Long short-term memory (LSTM)

**Basics of RNNs**

The standard feedforward neural networks (i.e,, DNNs and CNNs) are function generators associating appropriate output to input. However, certain types of data are serial in nature. A recurrent neural network (RNN) processes sequences such as stock prices, sentences one element at a time while retaining a memory (called a state) of what has come previously in the sequence. Recurrent means the output at the current time step becomes the input to the next time step. At each element of the sequence, the model considers not just the current input, but what it remembers about the preceding

elements. This memory allows the network to learn long-term dependencies in a sequence which means it can take the entire context into account when making a prediction such as predicting the next word in a sentence. A RNN is designed to mimic the human way of processing sequences: we consider the entire sentence when we form a response instead of words by themselves. For example, consider the following sentence:

"The concert was boring for the first few minutes but then was terribly exciting."

A machine learning model that considers the words in isolation would probably conclude this sentence is negative. An RNN by contrast should be able to see the words "but" and "terribly exciting" and realize that the sentence turns from negative to positive because it has looked at the entire sequence. Reading a whole sequence gives us a context for processing its meaning, a concept encoded in recurrent neural networks.

Figure 4.3 General form of RNN

The left part of Figure 4.3 shows that the input-output relation of a standard neural network is altered so that the output is fed into the input. But, the right part of Figure 4.3 shows that the scheme unwrapped through time. The input is the serial data $(x_1, ....., x_T)$ and the output is $(o_1, ....., o_T)$ . The output of a neural network is fed into the next constituent neural network in the next stage as part of the input. So the output $o_t$ depends on all the inputs $(x_1, ....., x_T)$ . It may be the case that it is not a good idea to make the output $o_1$

dependent only on $x_1$ . In fact $o_1$ itself should be produced in context.

RNNs can be used in different applications such as machine translation, speech recognition, generating image descriptions, video tagging, and language modeling.

Advantages of Recurrent Neural Network

1. An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
2. Recurrent neural network are even used with convolutional layers to extend the effective pixel neighborhood.

Disadvantages of Recurrent Neural Network

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using tanh or relu as an activation function.

**Vanishing and exploding gradient problem**

RNNs are very hard to train. Let us see the reason. The term $\frac{\partial E}{\partial h^t}$ used in backpropagation algorithm is a product of a long chain of matrices: $\left(\frac{\partial E}{\partial h^t}\right) = \left(\frac{\partial z^{l+1}}{\partial h^l}\right)\left(\frac{\partial h^{l+1}}{\partial z^{l+1}}\right) ... \left(\frac{\partial h^l}{\partial z^L}\right)\left(\frac{\partial E}{\partial h^L}\right)$, For the input $x = h^0$ , this chain is the longest. If the activation function $sigmoid(t)$ is the sigmoid function $sigmoid(t) = \frac{1}{1+e^{-t}}$ its derivative is $sigmoid'(t) = \frac{e^t}{(1+e^t)^2}$ which gets very small so that it practically vanishes except at a small interval near 0. It is one of the reasons why people ReLU activation function is prefered. But it is not a solution, as negative input values also kill the gradient and make it stay there. Even if one avoids such an outright vanishing gradient problem, the long matrix multiplication in general may make the gradient vanish or explode. This kind of problem gets even more aggravated in the case of RNNs, since RNNs normally require long chain of backpropagation not only through the layers of neural networks of constituent cells but also across the different cells. Hence, RNNs are difficult to train.

**Long short-term memory (LSTM)**

The Long Short-Term Memory (LSTM) was first proposed by Hochreiter and Schmidhuber [10] as a solution to the vanishing gradients problem. But it did not attract much attention until people realized it indeed provides a good solution to the vanishing and exploding gradient problem of RNN as described above. We will only describe the architecture of its cell. There are many variations in the cell architecture, but we present only the basics.

At the heart of an RNN is a layer made of memory cells. The most popular cell at the moment is the Long Short-Term Memory (LSTM) which maintains a cell state as well as a carry for ensuring that the signal (information in the form of a gradient) is not lost as the sequence is processed. At each time step the LSTM considers the current word, the carry,

and the cell state. The LSTM has 3 different gates and weight vectors: there is a “forget” gate for discarding irrelevant information; an “input” gate for handling the current input, and an “output” gate for producing predictions at each time step. However, as Chollet points out, it is fruitless trying to assign specific meanings to each of the elements in the cell.

Figure 4.4. Structure of LSTM cell

The structure of LSTM is shown in Figure 4.4 . The input vector is $x_t$ . The cell state denoted by $c_{t-1}$ and the hidden state $h_{t-1}$ are fed into the LSTM cell and $c_t$ and $h_t$ are fed into the next cell. Internally, it has four states: $i_t$ (input), $f_t$ (forget), $o_t$ (output), and $g_t$ . The forget state $f_t$ is obtained as a sigmoid output of a network with $x_t$ and $h_{t-1}$ fed into it as inputs. Since $f_t$ is a sigmoid output, each element in it has a value between 0 and 1. If it is close to 0, it erases $c_{t-1}$ by multiplication; if it is close to 1, it keeps $c_{t-1}$ by multiplication. Thus, $f_t$ is given the name ”forget state” because of this property, The input state $i_t$ and the output state $o_t$ are obtained by using (1) and (3), respectively. The state $g_t$ is also obtained similarly except that it is an output of the form tanh . This gives the $\pm$ sign. The product it
$otimes$ gt is then added to $c_t$ , and this gives new information to the cell state. The hidden state ht is obtained as in (6), and the cell output $y_t$ is the same as $h_t$ . The whole scheme is depicted in Figure 4.4.

$$
\begin{aligned}
i_t(input) &= \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) && (1)\\
f_t(forget) &= \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) && (2)\\
o_t(output) &= \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) && (3)\\
g_t &= tanh(W_{gx}x_t + W_{gh}h_{t-1} + b_g) && (4)\\
c_t(cell\ state) &= ft \otimes c_{t-1} + i_t \otimes g_t && (5)\\
h_t(hidden\ state) &= o_t \otimes tanh(c_t) && (6)\\
y_t(cell\ output) &= h_t && (7)
\end{aligned}
$$

### 5.6.3 References:


[1] Fukushima, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural Netw. 1988, 1, 119–130.

[2] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. Proc. IEEE 1998, 86, 2278–2324.

[3] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1106–1114.

[4] Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. arXiv 2013, arXiv:1311.2901.

[5] Simonyan, K.; Zisserman, A. deep convolutional networks for large-scale image recognition. arXiv 2014, arXiv:1409.1556.

[6] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.

[7] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770– 778.

[8] Hou, J.-C.; Wang, S.; Lai, Y.; Tsao, Y.; Chang, H.; Wang, H. Audio-Visual Speech Enhancement Using Multimodal Deep Convolutional Neural Networks. arXiv 2017, arXiv:1703.10893.

[9] Xu, Y.; Kong, Q.; Huang, Q.; Wang, W.; Plumbley, M.D. Convolutional gated recurrent neural network incorporating spatial features for audio tagging. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3461–3466.

[10] Hochreiter, S., Schmidhuber, J., Courville, A., Long short-term memory, Neural Computation 9(8):1735-80 (1997)

## 5.7 Non-negative Matrix and Tensor Factorization

### 5.7.1 Introduction

Many of the most descriptive features of speech are described by energy; for example, formants are peaks and the fundamental frequency is visible as a comb-structure in the power spectrum. A basic property of such features is that they are positive-valued. Negative values in energy are not physically realizable. However, most signal processing methods are applicable only for real-valued variables and inclusion of a non-negative constraints is cumbersome.

*Non-negative matrix factorization* (NMF or NNMF) and its tensor-valued counterparts is a family of methods which explicitly assumes that the input variables are *non-negative*, that is, they are by definition applicable to energy-signals. In some sense, NMF methods are an extension of prinicipal component analsys (PCA) -type and other *subspace methods* to positive-valued signals.

### 5.7.2 Model definition

Specifically, suppose that the power (or magnitude) spectrum of one window of a speech signal is represented as a $Nx1$ vector $v_k$, and furthermore we arrange the $K$ windows into an $NxK$ matrix $V$. The signal model we use is then

$$V \approx WH,$$

where $W$ is the $NxM$ weight matrix, $H$ is the $MxK$ model matrix and the scalar $M$ is the model order.

The idea is that $H$ is a fixed matrix corresponding to our model of the signal, viz. the source model. It describes typical types features of the data. With the weights $W$, we interpolate between the columns of $H$. In some sense, this is then a generalization of a codebook (see vector quantization), but such that we interpolate between codevectors. In addition, we require that all elements of $W$ and $H$ are non-negative, such that we ensure that $V$ is also non-negative.

Since the model order $K$ is chosen to be smaller than either $N$ or $K$, this mapping is generally an approximation. The model thus tries to catch *the relevant features of the input signal with a low number of parameters*.

The model is generally optimized by

$$\min_{W,H} \|V - WH\|_F \qquad \text{such that} \qquad W, H \geq 0.$$

Here the norm refers to the Frobenius norm, which is defined as the square root sum of squared elements. We do not have analytic solutions to the above optimization problem, but we can solve it by numerical methods, which are included in typical software libraries.

### 5.7.3 Application

A typical use of NMF type algorithms is source separation, where we find the solution of the above optimization problem and then identify those dimensions of $H$ which corresponds to the different sources. By retaining only those dimensions of $W$ which correspond to the desired source, we can thus extract the desired source signal from their mixture with the interfering other sources. For example, we might want to extract a speech signal corrupted by noise by extracting the dimensions corresponding to speech and removing those dimensions which correspond to noise.

Note however that NMF-type methods extract only the power (or magnitude) spectrum of the desired signal. In contrast, usually the input signal is a time-frequency representation which has also a phase-component. After application of NMF-estimation, we therefore need also an estimate of the phase-component of the signal. Such methods will be discussed in the speech enhancement chapter of this document.

For more information, see the Wikipedia article: Non-negative matrix factorization.

CHAPTER

SIX

# EVALUATION OF SPEECH PROCESSING METHODS

1. *Subjective quality evaluation*
2. *Objective quality evaluation*
3. *Other performance measures*
4. *Analysis of evaluation results*

## 6.1 Subjective quality evaluation

In speech processing applications where humans are the end-users, there humans are also the ultimate measure of performance and quality. *Subjective evaluation* refers to evaluation setups where human subjects measure or quantify performance and quality. From a top-level perspective the task is simple; subjects are asked to evaluate questions such as

- Does X sound good?
- How good does X sound?
- Does X or Y sound better?
- How intelligible is X?

However like always, the devil is in the details. Subjective evaluation must be designed carefully such that questions address information which is *useful* for measuring performance and such that information extracted is *reliable* and *accurate*.

Most commonly, subjective evaluation in speech and audio refers to *perceptual evaluation* of sound samples. In some cases evaluation can also involve interactive elements, such as participation in a dialogue over a telecommunication connection. In any case, perceptual evaluation refers to evaluation through the subjects' senses, which in this context refers primarily to *hearing*. In other words, in a typical experiment setup, subjects listen to sound samples and evaluate their quality.

Photo by Anthony Brolin on Unsplash

### 6.1.1 Aspects of quality

Observe that the appropriate evaluation questions are tightly linked to the application and context. For example, when designing a teleconferencing system for business applications, we are interested in entirely different types and aspects of quality than in design of hearing-aids.

From a top-level perspective, we discuss quality for example, with respect to

- *sound or speech quality*, relating to an intrinsic property of the audio signal,
- *interaction and communication quality*, as the experience of quality in terms of dynamic interaction,
- *service quality and user experience*, which is usually meant to include beyond sound and interaction quality, the whole experience of using a service or device, including responsiveness of system, network coverage, user interface, visual and tactual design of devices etc.

#### Sound and speech quality

The acoustic quality of the signal can be further described, for example, through concepts such as

- *noisiness* or the amount of noise the speech signal is perceived to have (perceptually uncorrelated noise)
- *distortion* describes how much parts of the speech signal are destroyed (perceptually correlated noise), though often also uncorrelated noises are referred to as distortions
- *intelligibility* is the level to which the meaning of the speech signal can be understood
- *listening effort* refers to the amount of work a listener has to use in listening to the signal and how much *listening fatigue* and *annoyance* a user experiences
- *pleasantness* describes how much the speech signal annoys the listener
- *resemblance* describes how close the speech signal is to the original signal
- *naturalness* describes how natural an artificial speech source sounds, often used as a last resort, when no other adjective feels suitable, then we can see that "*It sounds aunnatural*".
- *acoustic distance* or how near or far the far-end speaker is perceived to be, which is closely related to the amount of reverberation the signal has (and how much delay the communication path has)

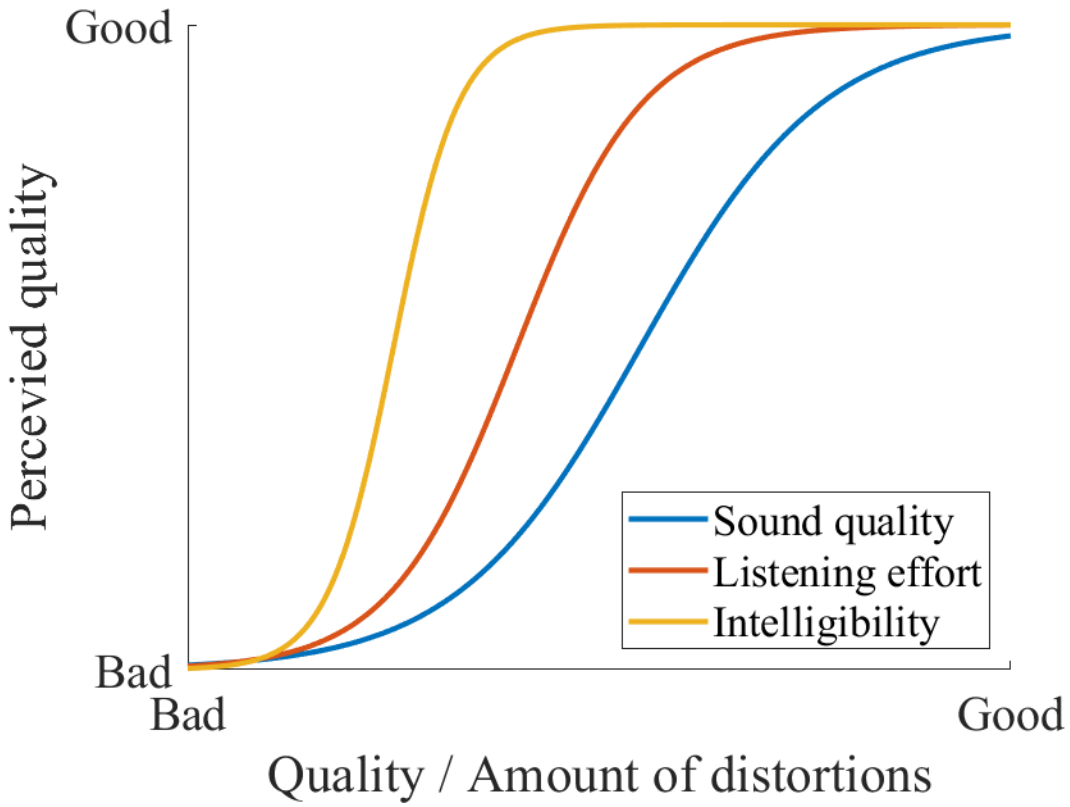
Note that we can also think of intelligibility and listening effort as separate aspects of sound quality. That is, we can perceive noisiness and distortions, but still not be annoyed by the quality at all and be able to listen to the sound effortlessly. Listening effort is usually a prerequisite for loss of intelligibility; if we have to listen carefully than it is exhausting in the long run. Quality, effort and intelligibility are, in this sense, addressing different quality levels, where intelligibility is an issue only at very bad quality, effort in the middle range whereas quality is always relevant (see figure on the right).

#### Interaction quality

The quality of interaction when using speech technology can be further described, for example, through concepts such as

- *delay* describes the delay between the acoustic event at the speakers end, to the time the event is perceived at the receiving end. It is further divided into algorithmic delay caused by processing, as well as delays caused by the transmission path.
- *echo* is the feedback loop of sound in the sound system, where a sound loudspeaker is picked by a microphone
- *presence or distance* is the feeling of proximity (and intimacy) that a user experiences in communication

- *naturalness* can also here be used to describe communication with the obvious meaning.




### 6.1.2 Choosing subjects - Naïve or expert?

When designing a subjective listening test, one of the first important choices is whether listeners are naïve or expert listeners. With naïve we refer to listeners who do not have prior experience in analytic listening nor do they have other related expertise. Expert listeners are then, obviously, subjects who are trained in the art of listening and have the ability to analytically evaluate small differences in sounds samples. Good expert listeners are typically researchers in the field who have years of experience in developing speech and audio processing and listening to sound samples. Some research labs even have their own training for expert listeners.

Expert listeners are the best listeners in the sense that they can give *accurate* and *repeatable* results. They can hear the smallest audible differences and they can grade the differences *consistently*. In other words, if an expert hears the same sounds in two tests, he or she can give them exactly the same score both times, and (in an ideal world) his or her expert colleagues will grade the sounds similarly. However, the problem is that expert listeners have skills far beyond those of average user of speech and audio products. If we want average users to enjoy our products, then we should be measuring the preferences of average users. It is uncertain whether the preferences of expert listeners align with average users. For example, expert listeners might notice a highly annoying distortion which an average user never discovers. The expert would therefore be unable to enjoy the product which is great for the average user.

Naïve listeners are therefore the best listeners in the sense that they reflect best the *preferences of the average population*. The downside is that since naïve listeners do not have experience in subjective evaluation, their answers have usually a high variance (the measurement is noisy). That is, if they hear the same samples twice, they are often unable to repeat the same grade (intra-listener variance) and the difference in grades between listeners is often high (inter-listener variance). Consequently, to get useful results from naïve listeners, you need a large number of subjects. To get statistically significant results, you often need a minimum of *10 expert* listeners, where you frequently need more than *50 naïve* listeners. The

difficulty of the task naturally can have a large impact on the required number of subjects (difficult tasks require a larger number of listeners for both expert and naïve listeners).

Note that the above definition leaves a large grey area between naïve and expert listeners. A naïve listener looses the naïve status when participating in a listening test. However, a naïve listener needs years of training to become an expert listener. This is particularly problematic in research labs, where there are plenty of young researchers available for listening tests. They are not naïve listeners any more, but they need training to become expert listeners. A typical approach is then to include the younger listeners regularly in listening tests, but remove those listeners in post-screening if their inter- or intra-listener variance is too large. That way the listeners slowly gain experience, but their errors do not get too much weight in the results. This approach however has to be very clearly monitored such that it does not lead to tampering of results (i.e. scientific misconduct). Any listeners removed in post-screening and the motivations for post-screening have to therefore be documented accurately.

### 6.1.3 Experimental design

To get accurate results from an experiment, we have to design the experiment such that it matches the performance qualities we want to quantify. For example, in an extreme case, an evaluation of the performance of noise reduction can be hampered, if a sound sample features a speaker whose voice is annoying to the listeners. The practical questions are however more nuanced. We need to consider for example:

- To which extent does the language of speech samples affects listening test results? Can naïve listeners grade accurately speech samples in a foreign language? Does the text-content of speech samples affect grading: for example, if the spoken text is politically loaded, would the grades of politically left- and right-leaning subjects give different scores?
- Does the text material, range of speakers and recording conditions reflect the target users and environments? For example, if we test a system with English-speaking listeners with speech samples in English, do the results reflect performance for Chinese users? Are all phonemes present in the material and do they appear equally often as they do in the target languages? Does the material feature background noises and room acoustics in the same proportion as real-world scenarios? Does gender of the speaker or listener play a role? Or their cultural background?
- Is the subject learning from previous sounds, such that the answers regarding the current sound are different from the previous sound? That is, if the subject hears the same sound with different distortions several times, then he already knows how the sound is supposed to sound like. Then perhaps he evaluates distortions differently, because he know how the sound is supposed to sound like.

To take into account such considerations, experiments can be designed in different ways, for example:

- We can measure *absolute* quality (How good is X?), or *relative* quality (How good is X in comparison to Y?) or we can *rank* samples ("Which one is better, A or B?" or "Order samples A, B and C from best to worst."). Clearly absolute quality is often the most important quality for users, since usually users do not have the opportunity to test products side by side. However, for example during development, two version of an algorithm could have very similar quality, such that it would be difficult to determine preference with an absolute quality measure. Relative quality measures then give more detailed information, explicitly quantifying the difference in performance. Ranking samples is usually used in competitions, for example, when a company wants to choose a supplier for a certain product, it is then useful to be able to measure the ordering of products. It is thus more refined than relative measures in terms of finding which one is better, but at the same time, it does not say how large the difference is between particular samples.
- In many applications, the target is to recover a signal after transmission or from a noisy recording. The objective is thus to obtain a signal which is as close as possible to the original signal. It can then be useful to play the original signal to subjects as a *reference*, such that they can compare performance explicitly with the target signal. While this then naturally gives listeners the opportunity to make more accurate evaluations, it is also not realistic. In real life, we generally do not have access to the original; for example, when speaking on the phone, we cannot directly hear the original signal, but can hear only the transmitted signal. We would therefore never be able to compare

performance to the target, but only absolute quality.In some cases it is also possible that some speech enhancement methods improve quality such that the output sounds better than the original non-distorted sound!

- In some experimental designs, the subject can listen to sounds many times, even in a loop. That way we make sure that the subject hears all the minute details. However, that is unrealistic since in a real scenario, like a telephone conversation, you usually can hear sounds only once. Repeated sounds are therefore available only for expert listeners.
- In choice of samples, the length of samples should be chosen with care. Longer samples reflect better real-life situations, but bring many problems, for example:
  - The ability to listeners to remember particular features of the sound, especially in comparison to other sounds, is very limited. This would reduce the accuracy of results.
  - Longer samples can have multiple different characteristics, which would warrant a different score. The listener would then have to perform a judgement; which part of the sentence or sample is more important, and which type of features are more important for quality? This can lead to ambiguous situations.

  Listening to very short samples can, in turn, make features of the sound audible which a listener could not hear in real-life setting.
- Usually we prefer to have speech samples in the same language as the listeners. With expert listeners this constraint might not be so strict.
- Speech samples should generally be *phonetically balanced* "nonsense" sentences. With phonetically balanced, we refer to sentences where all phonemes appear with the same frequency as they appear on average in that particular language. With nonsense sentences, we refer to text content which does not convey any particular, loaded or surprising meaning. For example, "An apple on the table" is a good sentence in the sense that it is grammatically correct and there is nothing strange with it. Examples of bad sentences would be "Elephants swimming in champagne", "Corporations kill babies" and "The dark scent of death and mourning".
- When playing samples to subjects, they will both *learn* more about the samples, but also experience *fatigue*. Especially for naïve listeners, the performance of subjects will therefore change during an experiment. It is very difficult to take such changes in performance into account in analysis and it is therefore usually recommended to randomize the ordering of samples separately for each listener. That way the effects of learning and fatigue will be dispersed evenly across all samples, such that they have a uniform effect on all samples.
- To measure the ability of listeners to consistently grade samples, it is common practice to include items in the test whose answers are known. For example, we can
  - repeat a sample twice, such that we measure the listeners ability to give the same grade twice,
  - have the original *reference* signal hidden among test samples (known as the *hidden reference*), such that we can measure the listeners ability to give the perfect score to the perfect sample,
  - include samples with known distortions among the test items, such that we can compare results with prior experiments which included the same known samples. Typically such known distortions include for example low-pass filtered versions of the original signal. Such samples are known as *anchors* and low and high quality anchors are then respectively known as *low-anchor* and *high-anchor*.

### 6.1.4 Some use cases

- *During research and development* of speech and audio processing methods, researchers have to evaluate the performance of their methods. Most typically such evaluations are quite informal; when you get output from a new algorithm, the first thing to do is to listen to the output - is it any good? In some stages of development, such evaluations are an ongoing process; tweak a parameter and listen how it affects the output. In early development, listening is in practice also a sanity check; programming errors often cause bad distortions on the output, which can be caught by listening.
- *When publishing results* and *at later stages of development*, it is usually necessary to evaluate quality in a more formal manner. The engineer developing an algorithm is not a good listener, because he has extremely detailed knowledge about the performance and could often spot his or her own method, from a set of sound samples. The developer is therefore *biased* and not a reliable listener.Therefore, when publishing results we need *reproducible* experiments in the sense that if another team would repeat the listening experiment, then they could draw the same conclusions. Listening tests therefore have to have a sufficient number of listeners (expert or naïve) such that the outcome is statistically significant (see *Analysis of evaluation results*).
- *When selecting a product among competing candidates* we would like to make a good evaluation. The demands are naturally very different depending on the scenario; 1) if you want to choose between Skype, Google and Signal for your personal VoIP calls during a visit abroad, you are probably content with informal listening. 2) If on the other hand, you are an engineer and assigned with the task of choosing a codec for all VoIP calls within a multi-national company, then you probably want to do a proper formal listening test.
- *Monitoring quality of in-production system*; The quality of a running system can abruptly or gradually change due to bugs and equipment-failures, including memory-errors. To detect such errors, we need to monitor quality. Often such monitoring is based on automated objective tests.

### 6.1.5 Frequently used standards and recommendations for quality evaluation

#### Expert listeners

- By far the most commonly used standard applied with expert listeners is known as MUSHRA, or MUltiple Stimuli with Hidden Reference and Anchor, defined by ITU-R recommendation BS.1534-3. It offers direct comparison of multiple target samples, with a reference signal, hidden reference and anchor. Users can switch between samples on the fly and many interfaces also allow looping short segments of the signal.Each sample is rated on an integer scale 1-100.MUSHRA is very useful for example when publishing results, because it is simple to implement and well-known. Open source implementations such as webMUSHRA are available.Practical experience have shown that MUSHRA is best applied for intermediate quality samples, where a comparison of 2-5 samples is desired. Moreover, a suitable length of sound samples ranges from 2 to approximately 10 seconds. Furthermore, if the overall length of a MUSHRA test is more than, say, 30 minutes, then the fatigue of listeners starts be a significant problem. With 10 good listeners it is usually possible to achieve statistically significant results, whereas 6 listeners can be sufficient for informal tests (e.g. during testing). These numbers should not be taken as absolute, but as practical guidance to give a rough idea of what makes a usable test.MUSHRA is however often misused by omitting the anchors; a valid argument for omitting anchors is that if the distortions in the target samples are of a very different type then the typical anchors, then anchors do not provide an added value. Still, such omissions are not allowed by the MUSHRA standard.
- For very small impairments in audio quality, Recommendation ITU-R BS.1116-3 (ABC/HR) is recommended instead of MUSHRA (see https://www.itu.int/rec/R-REC-BS.1116-3-201502-I/en).

For illustrations and examples of the MUSHRA test, see https://www.audiolabs-erlangen.de/resources/webMUSHRA

### Naïve listeners

**P.800** is the popular name of a set of listening tests defined in the standard ITU-TRecommendation P.800 "Methods for subjective determination of transmission quality". It is intended to be a test which gives as realistic results as possible, by assessing performance in setups which resemble real use-cases. The most significant consequences are that P.800 focuses on naïve listeners and, since telecommunication devices are typically hand-held over one ear, P.800 mandates tests with headphones which are held only on one ear. To make the test simpler for naïve listeners, P.800 most typically uses an integer scale 1-5 known as mean opinion score (MOS). Each grade is given a characterisation such as

| Rating | Label |
|---|---|
| 1 | Excellent |
| 2 | Good |
| 3 | Fair |
| 4 | Poor |
| 5 | Bad |

This makes the ratings more concrete and easier to understand for users. A downside of labelling the ratings is that such labels are specific to each language and the MOS scores given in different languages might thus not be directly comparabale. Who is to know whether *excellent*, *erinomainen*, *□□□□□*, and *маш сайн* mean exactly the same thing? (Those are english, finnish, arabic and mongolian, in case you were wondering.)

P.800 is further split into

- *Conversation opinion tests*, where participants grade the quality after *using* telecommunication system for a conversation. Typically the question posed to participants is "Opinion of the connection you have just been using: Excellent, God, Fair, Poor, Bad". An alternative is "Did you or your partner have any difficulty in talking or hearing over the connection? Yes/No".
- *Listening opinion tests*, where participants grade the quality after *listening* to the output of a telecommunication system.

The grading of listening opinion tests can, more specifically, be one of the following:

- *Absolute category rating (ACR)*, where the above MOS scale is used to answer questions like "How good is system X?"
- *Degradation category rating (DCR)*, where the objective is to evaluate the amount of degradation caused by some processing. Samples are preseted to listeners by pairs (A-B) or repeated pairs (A-B-A-B) where A is the quality reference and B the degraded sample. Rating labels cane be for example

| Rating | Label |
|---|---|
| 1 | Degradation is inaudible |
| 2 | Degradation is audible but not annoying |
| 3 | Degradation is slightly annoying |
| 4 | Degradation is annoying |
| 5 | Degradation is very annoying |

- *Comparison category rating (CCR)*, is similar to DCR, but such that the processed sample B can be also better than A. Rating labels can then be for example

| Rating | Label |
|---|---|
| 3 | Much better |
| 2 | Better |
| 1 | Slightly better |
| 0 | About the same |
| -1 | Slightly worse |
| -2 | Worse |
| -3 | Much worse |

**P.804** Subjective diagnostic test method for conversational speech quality analysis

**P.805** Subjective evaluation of conversational quality

**P.806** A subjective quality test methodology using multiple rating scales

**P.807** Subjective test methodology for assessing speech intelligibility

**P.808** Subjective evaluation of speech quality with a crowdsourcing approach

**P.835** Subjective test methodology for evaluating speech communication systems that include noise suppression algorithm

And many more, see https://www.itu.int/rec/T-REC-P/en

Holding a phone on one ear

Photo by Fezbot2000 on Unsplash

### 6.1.6 Intelligibility testing

When a speech signal has a been corrupted by a considerable level of noise and/or reverberation, it's intelligibility starts to deteriorate. We might miss-interpret or -understand words or entirely miss them. Observe that the level of distortion is quite a bit higher than what we usually consider in quality-tests.

Typically we have to choose between correctly interpreted words or phonemes/letters. For example, if the sentence is

> How to recognize speech?

and what we hear is

> How to wreck a nice beach?

then we can count correct words for example as

```
How to recognize    speech?
How to wreck a nice beach?
       S     I I    S
```

where we use the notation S - substitution, I - insertion, D - deletion. The word error rate would then be

$$WER = 100 \times \frac{S + D + I}{N}$$

where $N$ is the total number of words. In the above example we thus have WER = 100%.

If we would go letter by letter, than instead we would have

```
How to  rec..og.nize speech
How to wreck a  nice  beach
       I     SI   S  DS S
```

Here we would then define the letter error rate as

$$LER = 100 \times \frac{S + D + I}{N}$$

where $N$ is the total number of letters. The value in the above example would then be 33%.

It is clear that word error rate is thus much more strict than letter error rate. A single incorrect letter will ruin a word, while the letter error rate is affected much less. WER is much easier to compute than the LER and also leads to fewer ambiguous situations. It is however dependent on the application which measure is better suited.

Observe that both word and error rates are applicable as both objective measures, in speech recognition experiments, as well as a subjective measure, where human subjects evaluate the quality of sounds.

## 6.2 Objective quality evaluation

### 6.2.1 Objective estimators for perceptual quality

With "objective evaluation" we usually refer to *estimators of perceptual quality*, where the objective is to predict the mean output of a *subjective listening* test using an algorithm. That is, we want a computer to listen to a sound sample and try to "guess" what a human listener would say about its quality (on average).

It is then clear that subjective evaluation is always the "true" measure of performance and objective evaluation is an approximation thereof. In this sense, subjective evaluation is "better". However, there are many good reasons to use objective instead of subjective evaluation:

- *Subjective evaluation is expensive*; a test requires that a large number of persons listens to sound samples, which is both time-consuming and requires infrastructure. Objective evaluation is performed on a computer, such that you can generally test a large number of sound samples in a short time.
- *Subjective evaluation is noisy*; even with a large number of expert listeners it is generally difficult to get exactly the same result in two consecutive tests. Objective evaluation always gives the same rating for the same input, such that testing is consistent and reliable. This is especially important for scientific reproducibility; an independent laboratory can verify and confirm your results, the objective measure always gives the same output. With subjective evaluation, independent researchers can get different results, and you can never be 100% certain where the difference in results comes from. Did one of the researchers do an error or is it just that subjective listeners give always slightly different results?

Some of the most frequently used objective measures include:

- PESQ is probably the most frequently used objective evaluation method and it is defined in ITU-T Recommendation P.862: Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs (2001). It is thus an evaluation method designed explicitly for telecommunications applications. It estimates the mean score of an P.800 ACR test.PESQ accepts only narrow-band input and is *not directly applicable* on other bandwidths. The degradation types whose effect PESQ can reliably predict are
  - Speech input levels to a codec
  - Transmission channel errors
  - Packet loss and packet loss concealment with CELP codecs
  - Bit rates if a codec has more than one bit-rate mode
  - Transcodings
  - Environmental noise at the sending side
  - Effect of varying delay in listening only tests
  - Short-term time warping of audio signal
  - Long-term time warping of audio signal

  Observe that distortions other than those listed above can provide unreliable results. An important missing feature are distortions caused by spectral processing, such as musical noise. Specifically, for example, using PESQ to evaluate speech enhancement methods based on processing in the STFT domain, *can give unreliable results*.
- Perceptual Objective Listening Quality Assessment (POLQA) is the successor of PESQ and defined in ITU-T Recommendation P.863: Perceptual objective listening quality assessment. It is important to notice that for most practical purposes, POLQA is better than PESQ. It has a wider range of applications and acceptable degradation types and the output is more reliable. However, from a scientific perspective it is extremely regrettable that implementations of POLQA are commercial and *expensive* products, rendering application of POLQA infeasible in normal scientific work. Even if an individual team could afford purchasing a POLQA licence, verification of POLQA results by independent research labs is possible only if they also purchase a POLQA licence. Despite of its limitations, PESQ has therefore remained the scientific standard in objective evaluation of speech.
- Perceptual Evaluation of Audio Quality (PEAQ) evaluates, instead of only speech, also other types of audio samples. It is therefore less accurate with respect to distortions specific to speech signals, but it generalizes better to other audio such as music and background noises. The measure is defined in ITU-R Recommendation BS.1387: Method for objective measurements of perceived audio quality (PEAQ).
- The short-term objective intelligibility (STOI) measure focuses on how *intelligible* a speech sample is. It is thus clearly focused on lower-quality scenarios where speech is so badly corrupted that it is hard to understand what is said. Like all objective measures, it is not a completely reliable estimate of quality, but can be useful in combination with other measures. A good feature of STOI is that an implementation is available.

### 6.2.2 Other objective performance criteria

There are many cases where other performance criteria are well-warranted than merely prediction of subjective listening test results. Most typically these criteria are applied when there is no user involved, such as speech recognition, or, when we want to have more detailed characterization of performance than given by predictors of subjective listening test results.

Some examples of such performance criteria include:

- *Word error rate (WER)* is used in speech recognition to measure the proportion of words correctly recognized from a test signal.
- *Signal to noise ratio (SNR)* is used to measure the proportion of the desirable speech signal and undesirable noise components (which includes for example background noises, distortions caused by processing algorithms and transmission, as well as undesirable competing speakers).
- *Perceptual signal to noise ratio (pSNR)* measures SNR in a perceptually motivated domain. Essentially distortions are weighted such that they approximately correspond to human perception. This is similar to the above predictors of subjective listening tests, but works also on small segments of speech. It can be used to for detailed analysis of distortions to, for example, which parts of the signal contain undesirable distortions.
- *The speech distortion index (SDI)* measures the amount by which a desirable speech signal is distorted. In speech enhancement, it is often used in combination with the *noise attenuation factor* (NAF), which measures the amount by which undesirable noises are removed. It is clear that by doing nothing, we obtain a perfect SDI and by setting the output to zero, we obtain a perfect NAF. Neither outcome is usually satisfactory. It is therefore usually not clear what the right balance between the two measures are.
- Unweighted and weighted average recall (UAR, WAR) are often used to measure performance in speech classification tasks, such as classifying a speech segment into one of finite number of possible emotions. UAR is defined as the mean of class-specific recalls (the proportion of class samples recognized correctly) while WAR is the overall proportion of samples recognized correctly across all classes (sometimes also referred to as *accuracy*). UAR is often preferred over WAR in experiments where there is a notable class imbalance in the test data, and where it is important to have systems that are also sensitive to the less-frequent classes.
- Receiver operating characteristic (ROC) curves and its derivatives such as area under the curve (AUC) or equal error rate (EER) are often used to report performance of systems that have some type of detection threshold that can be varied, and when performance for each threshold value is measured in terms of precision and recall. For instance, performance of speaker verification systems is often evaluated using such metrics.

## 6.3 Other performance measures

### 6.3.1 Computational Complexity

On an application level, speech processing algorithms usually are used in low-resource devices like mobile phones. Mobile devices have limited computational capabilities and, in order to preserve their battery, it is important to design efficient algorithms for them. There are multiple ways of analyzing the computational complexity of an algorithm depending on the stage of the design process or the purpose of the final application:

### Big-O notation:

The complexity of an algorithm is usually understood as a measurement of the time that an algorithm would take to complete, given an input of size n. When the size of the input grows, the computing time should remain within a practical bound. For this reason, complexity is measured asymptotically as n approaches infinity. The most popular representation of algorithmic complexity is the Big-O notation. The Big-O notation gives an upper bound to the growth of the computing time of an algorithm. This proves especially useful, because this notation allows us to compare algorithms in worst case scenarios. Figure 1 shows the growth rate of different Big-O notations with respect to the input size.

For example, a complexity of O(n), read as "O n complexity', represents an algorithm whose computation time grows linearly with the input size. Some examples of every type of complexity are:

- O(1) - The computation time does not grow with the size of the input:
  - Accessing an array index (a = array[4]).
- O(log n) - The computation time grows with the logarithm of the input size:
  - Binary Search algorithms.
- O(n) - The computation time grows linearly with the input size:
  - Traversing an array.
  - Comparing two strings.
- O(n2) - The computation time grows with the square of the input size:
  - Matrix operations like traversing a 2D array or multiplying a 1D array with a 2D matrix.
  - Conventional Discrete Fourier Transform (Matrix multiplication).
- O(n log n) - The "log n" term is added when O(n2) algorithms are performed with Divide and Conquer techniques to increase their efficiency.
  - Fast Fourier Transform.
- O(2n) - The computation time doubles with each addition to the input data, therefore it grows exponentially:
  - Recursive algorithms → To solve a problem of size N, it is necessary to solve two problems of size N - 1.
- O(n!) - Factorial growth represents algorithms that grow even faster than exponential examples:
  - Finding all the possible permutations of a list.

### Weighted Million Operations Per Second (WMOPS), ITU-T. Software tool library: User's manual, 2009

The Big-O notations gives us an intuitive idea of the complexity of specific algorithms. This allows us to compare which algorithm to use and choose the most efficient option. However, in applications like speech coding, it is important to know the exact number of operations that the system needs to perform in order to process each frame of audio.

The ITU-T provides guidelines to measure the number of operations in a program. This measurement takes into account that not all the operations have the same computational load and scales their values accordingly. For example, a logarithm is a much heavier operation than an addition. The final result is represented as Weighed Million Operations Per Second (WMOPS). Table 1 shows the weights used for each specific operation carried out.

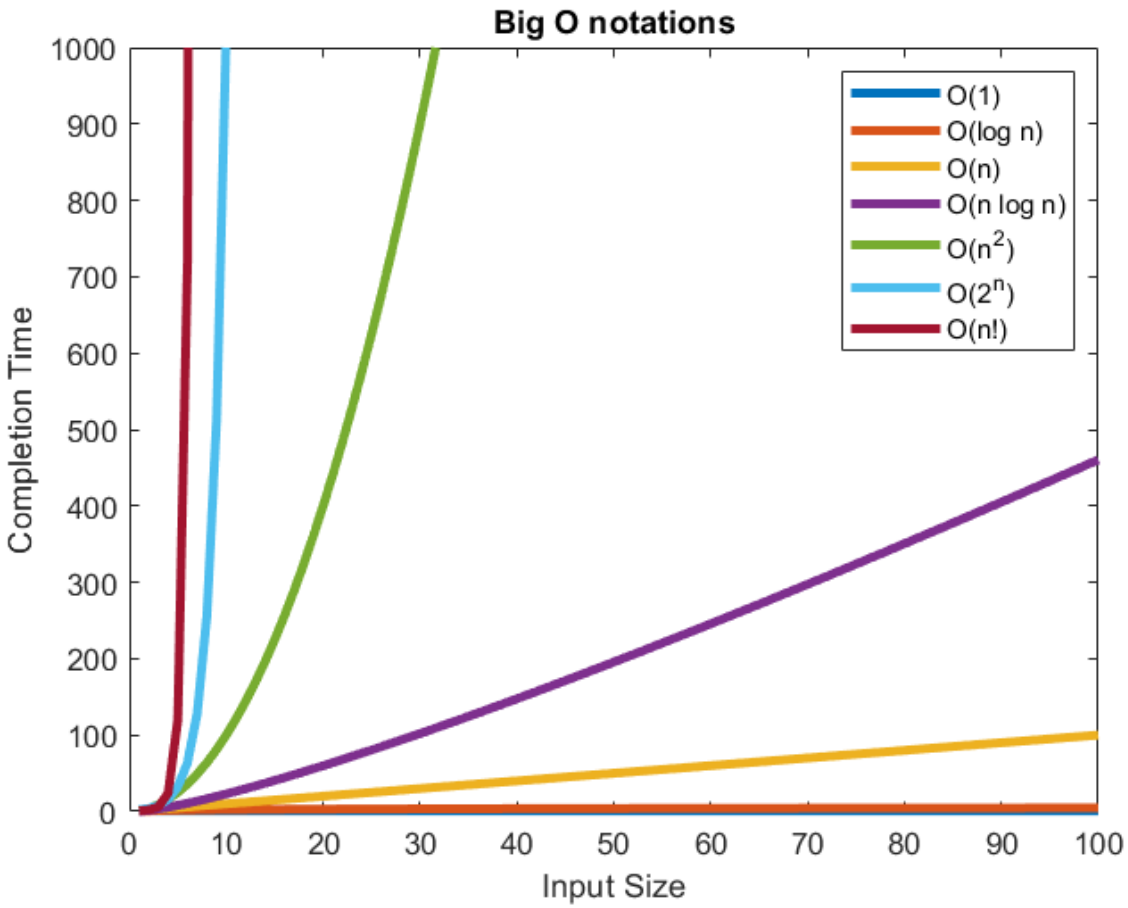



**Figure 1:** Evolution of computation time for multiple Big-O notations dependint on the input size.

| | | |
|---|---|---|
| Operation | Example | Weight |
| Addition | a = b + c | 1 |
| Multiplication | a = b ∗ c | 1 |
| Multiplication + addition | a+ = b ∗ c | 1 |
| Move | a = b | 1 |
| Store in array | a[i] = b[i] + c[i] | 1 |
| Logical | AND, OR, etc. | 1 |
| Shift | a = b >> c | 1 |
| Branch | if, if…else | 4 |
| Division | a = b/c | 18 |
| Square-root | a = sqrt(b) | 10 |
| Transcendental | sine, arctan, etc. | 25 |
| Function call | a = func(b, c, d) | 2 + number of arguments passed and returned |
| Loop initialization | for(i=0;i | 3 |
| Indirect addressing | a = b.c | 2 |
| Pointer initialization | a[i] | 1 |
| Exponential | pow, en | 25 |
| Logarithm | log | 25 |
| Conditional test | used in conjunction with BRANCH | 2 |

**Table 1:** Operations accounted by the WMOPS tool and their relative weight.

## 6.4 Analysis of evaluation results

To determine the quality and performance of speech processing methods, we often use *subjective* and *objective evaluation* methods. These methods however give an array of results, one result for each sound sample for objective methods, and in the case of subjective listening tests, one result for each sounds sample per listener. Invariably, the results are noisy in the sense that each sound sample and each listener will give a different result. How do we then determine whether one method is better than the other? In short, we have to perform statistical analysis of results. The types of question we can answer with statistical analysis include: "Based on measurement results, which result is most probable, that A is better than B, that B is better than A, or is it impossible to determine?"

### 6.4.1 Informal analysis

In practical laboratory work, after an experiment, you would always like to get a first impression of the results as quickly as possible. With informal analysis, we here refer to quick-and-dirty evaluation of data to determine whether there it is worth investing time in a more formal analysis. That is, if we already in the informal analysis determine that our new method is not really better than past methods, then it more productive to improve the method rather than perform a detailed analysis of results. On the other hand, a detailed analysis can sometimes reveal effects which are not visible in informal analysis and which could be used to improve the method in question. In any case, when reporting results (in a publication or even just to your superior), informal analysis should never replace properly applied statistical tests. Informal tests just given an indication of how much work will be required for proper tests.

#### Example 1

Suppose we want to compare methods A and B, and we have already applied PESQ on the outputs of both methods for 100 speech samples. How do we know if A is better than B?

**First idea:** Calculate the mean and standard deviation of both measurements. Which method has a higher (better) mean? Is the standard deviation clearly smaller than the difference in means?

- Suppose the mean of PESQ scores for A and B are 4.5 and 3.1, and their corresponding standard deviations are 0.2 and 0.15. Clearly A has a higher (better) PESQ score than B. The standard deviations are small which indicates that the measurement rarely diverge far from the mean. *This is the best case scenario* but unfortunately it happens rarely in practice.
- Suppose the mean of PESQ scores for A and B are 3.5 and 3.1, and their corresponding standard deviations are 0.2 and 0.15. Clearly *the mean* PESQ score is higher (better) for A than B. In this case it is however not immediately clear whether we can draw any definite conclusions. Is the difference significant? The standard deviations are not much smaller than the difference between the means, indicating that sometimes A could be worse than B! *Further analysis is required.*

Note that here we have chosen to discuss the *standard deviations*, although we could equivalently present the *variances* of measurements. The benefit of the standard deviation is that it is expressed in the same units as the mean, such that we can directly get an intuitive impression of the magnitude of the differences in means in comparison to the standard deviations.

### Informative illustrations 1

### Histograms

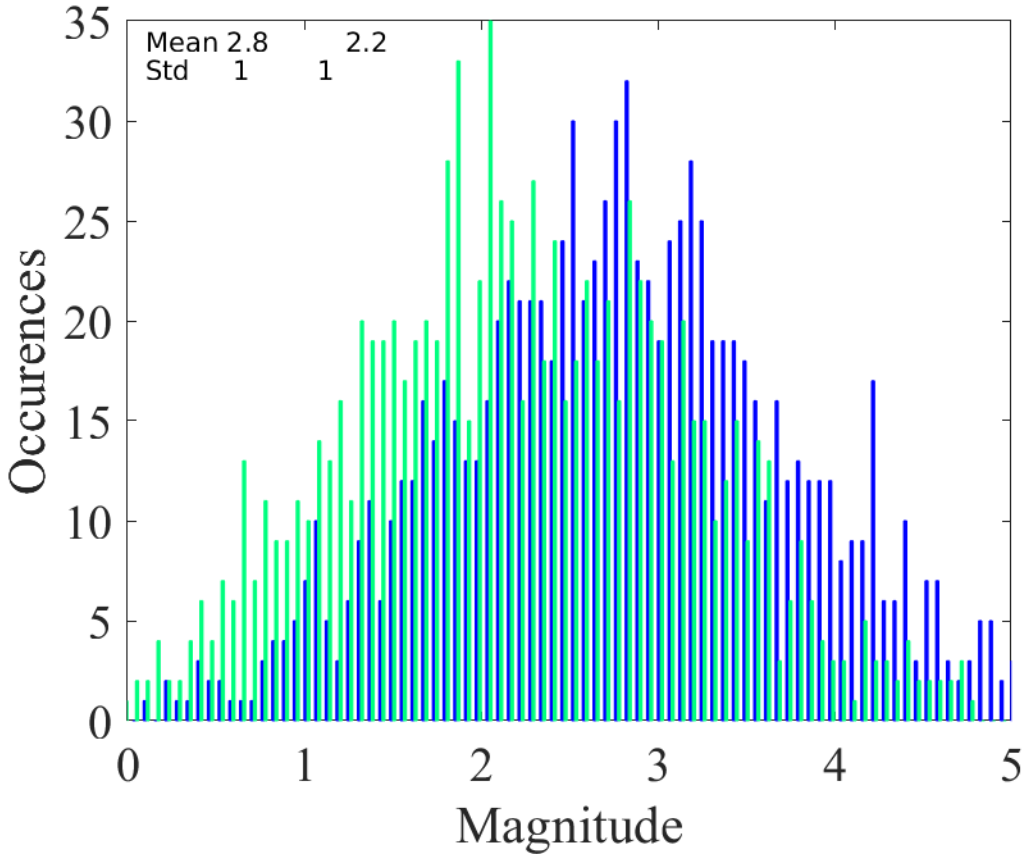
By plotting the histograms of two measurements, we can study the distribution of measurements. For example in Histogram 1 on the right, we can see two distributions (blue and green), whose means are 2.8 and 2.2. We can clearly see that the means of the sample are different, but it is not immediately clear whether that is just a random coincidence or if this is statistically significant. In Histogram 2 on the right, we see a much clearer difference. Though the means are the same, 2.8 and 2.2, the standard deviations are now smaller 0.25 instead of 1.0, such that we can be fairly confident that the difference in means is significant.In Histogram 3, there is not doubt left, the distributions are not overlapping and a statistical test would for sure show a significant difference.

If we then compare histograms 2 and 4, we see that they have the same means and standard deviations. However, while from histogram 2 we might not be entirely certain that the difference is significant, in histogram 4 the amount of overlap is much reduced because the distributions are skewed. Therefore already in informal analysis, we can be rather confident that there is a significant difference in distributions. This example thus demonstrates that the histograms can reveal properties of the measurements which cannot be captured by the mean and standard deviation.

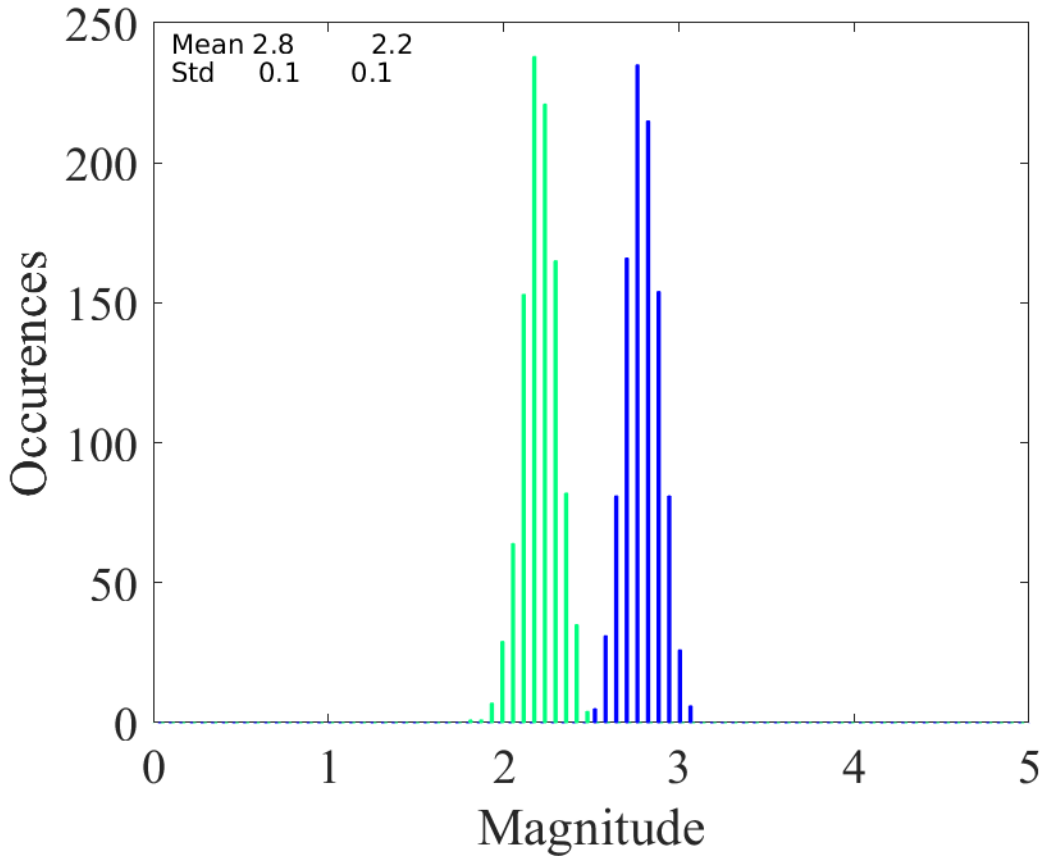
A further example which demonstrates how some differences are not captured by the mean and standard deviation is illustrated in Histogram 5. Here both measurements have the same mean, which could give the impression that both measurements are "the same". However, by looking at the histogram, we find that the blue curve actually has two peaks, one clearly lower and another higher than the green curve. Further analysis is thus needed to determine the cause for the strange distribution. Typically, for example, some methods behave differently for different inputs. A speech enhancement method could be for example be effective for voiced sounds, but fail for unvoiced sounds, such that the output SNR has two peaks corresponding to respective classes of sounds.

These two last examples, Histograms 4 and 5, illustrate why it is in general important to test whether evaluation results follow a Gaussian (normal) distribution. Analysis of approximately Gaussian results is less complicated then the analysis of skewed and multi-modal distributions. However, applying methods which assume Gaussian distributions on data which does not follow Gaussian distributions will *often lead to incorrect conclusions*.

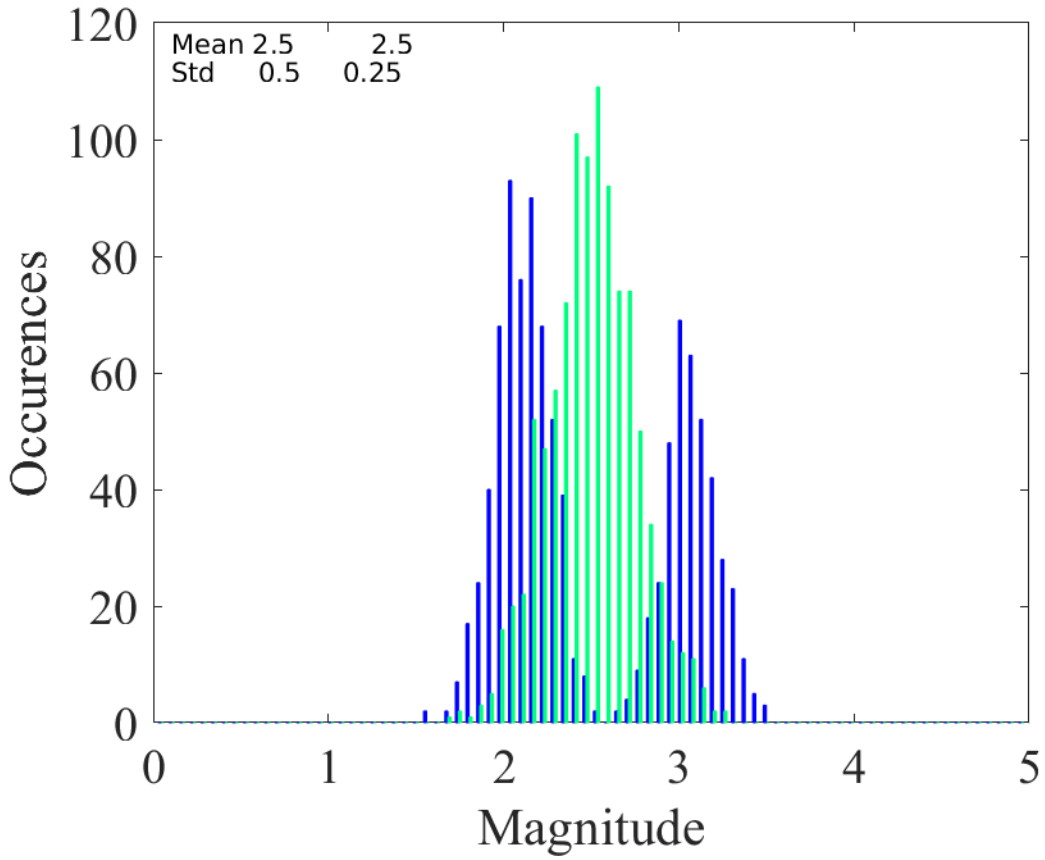
Histogram 1: Large overlap between distributions A (blue) and B (green)



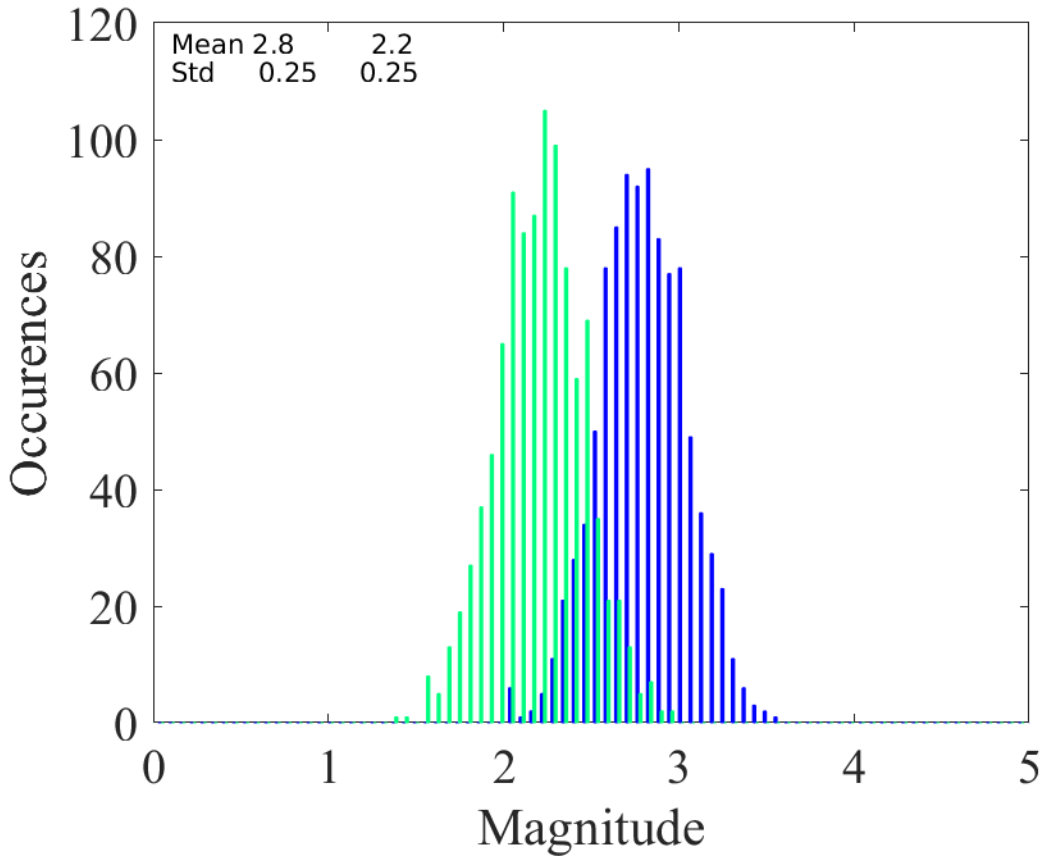

Histogram 3: Very small overlap between distributions



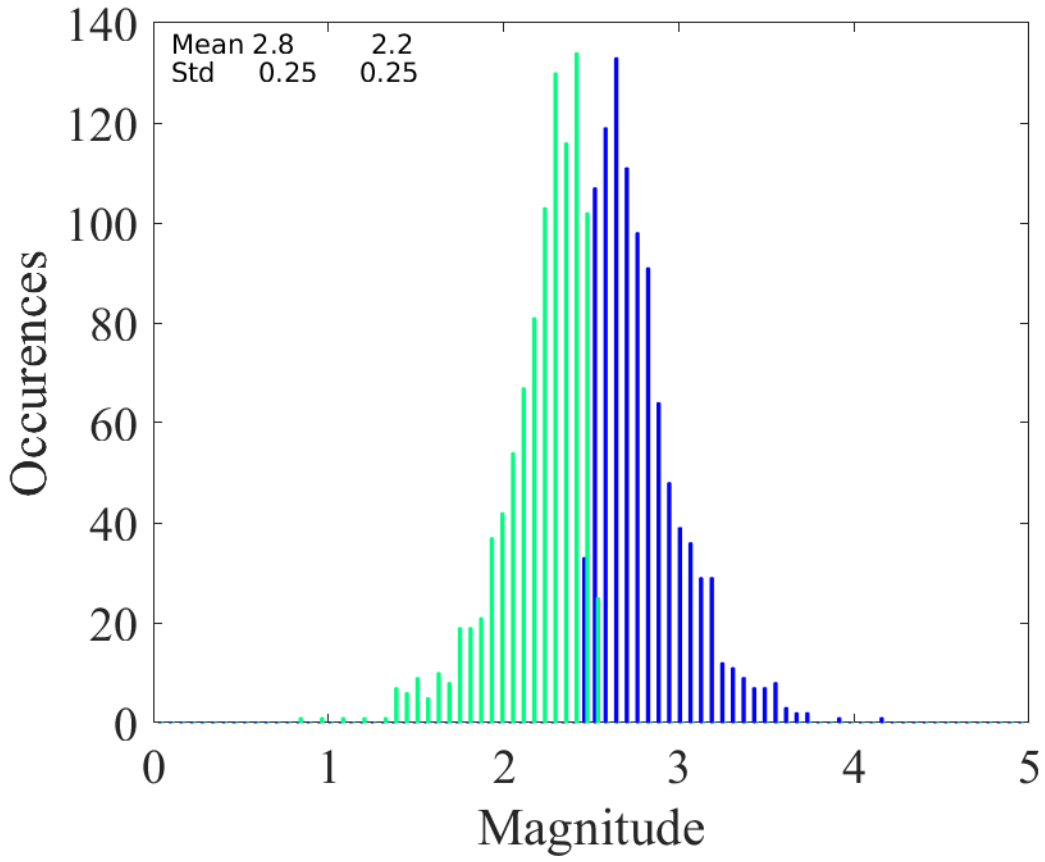

Histogram 5: Bimodal distribution for A, that is, the blue curve has two peaks.




Histogram 2: Smaller overlap between distributions




Histogram 4: Skewed distributions such that overlap is smaller than standard deviation would indicate

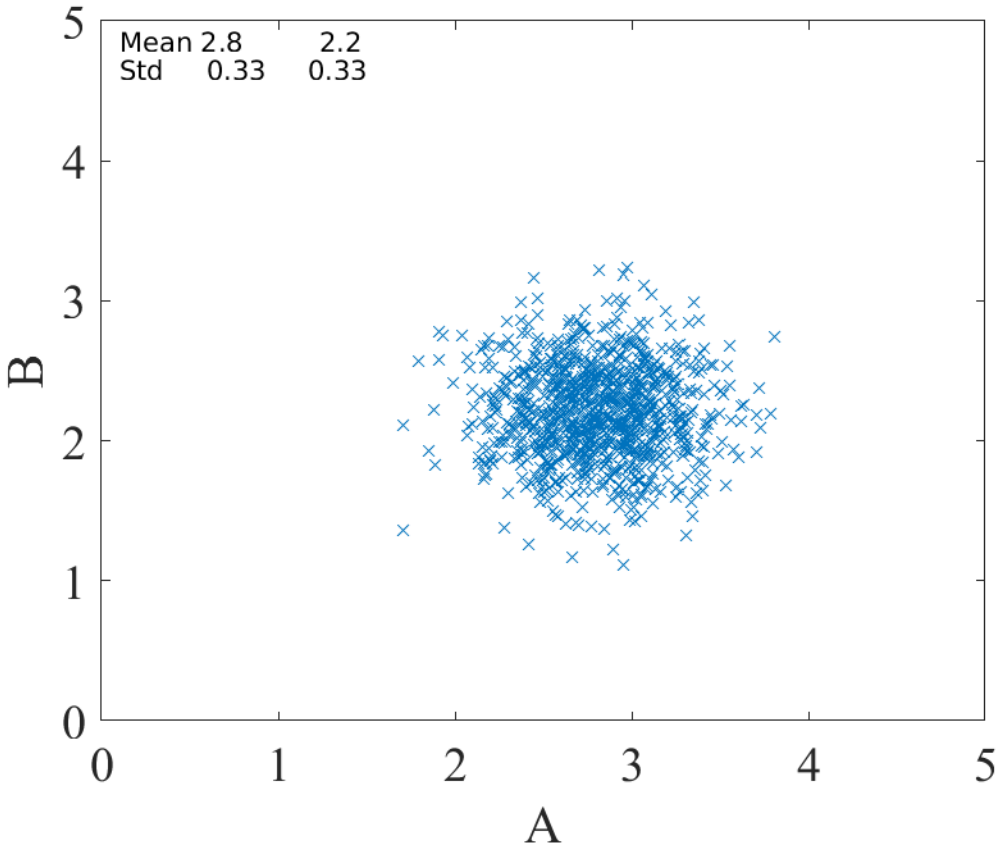



### Scatter plots

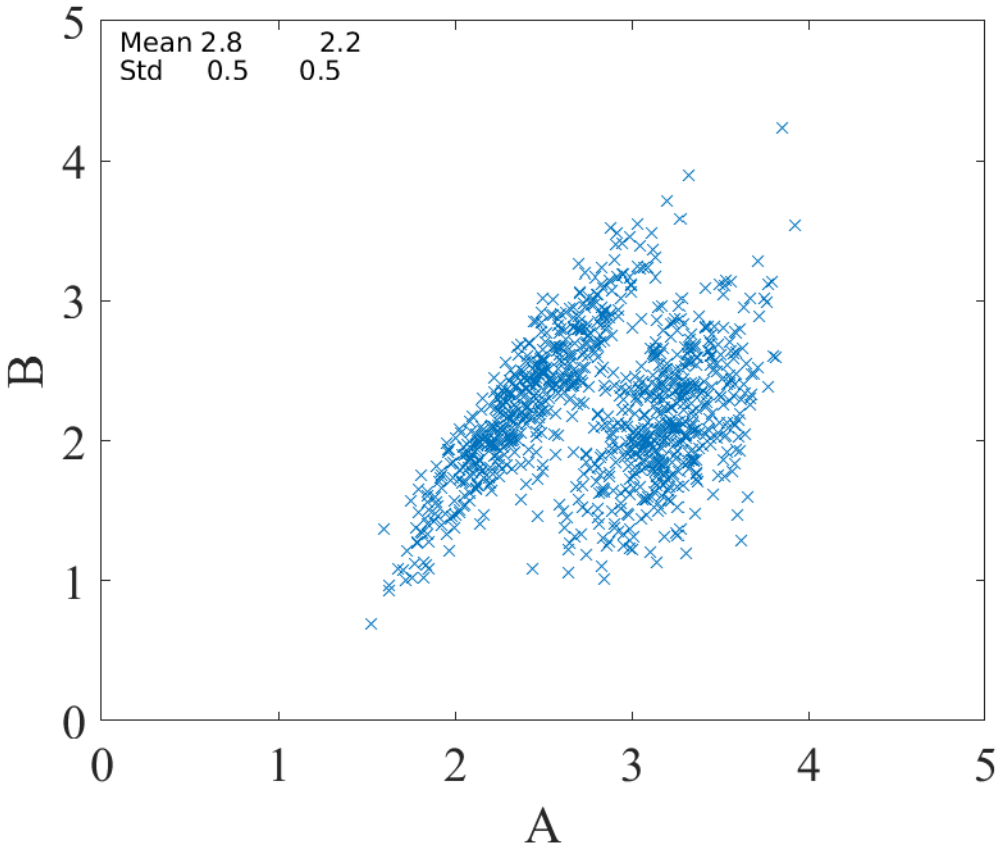
A useful tool for determining whether there are in hidden structures between the results of two methods is to plot a scatter plot of the results, where we plot dots (or other symbols) in a graph whose x- and y- axis correspond to the evaluation results of two methods. Scatter plots 1-3 on the right illustrate different typical scenarios. Plot 1 shows measurements in a circular area, indicating that the measurements are uncorrelated. In speech and audio experiments this is highly unusual. Scatter plot 2 is much more common, where measurements form a tight band. This means that measurements are correlated. For example, in a speech coding scenario, it could be that some speech samples are easy to encode with high quality, such that methods A and B both give high scores. Our objective is however to determine whether A or B is better, and therefore we want to cancel out the effect of these correlations. We can then take the difference between A and B to get the relative quality of the two methods (see histogram on the right).

Another typical scenario is illustrated in Scatter plot 3, where the scatter plot seems to form two distinctive groups. This would indicate that there are some underlying categories in the data (such as male/female, voiced/unvoiced, speech/silence) where performance is different. Typically we would then like to find out what the categories to determine if we can improve performance based on that information. For example, if performance is bad for unvoiced sounds, we could have a detector for unvoiced sounds, and perform different processing for such sounds. See also section “Parallel plots” below.

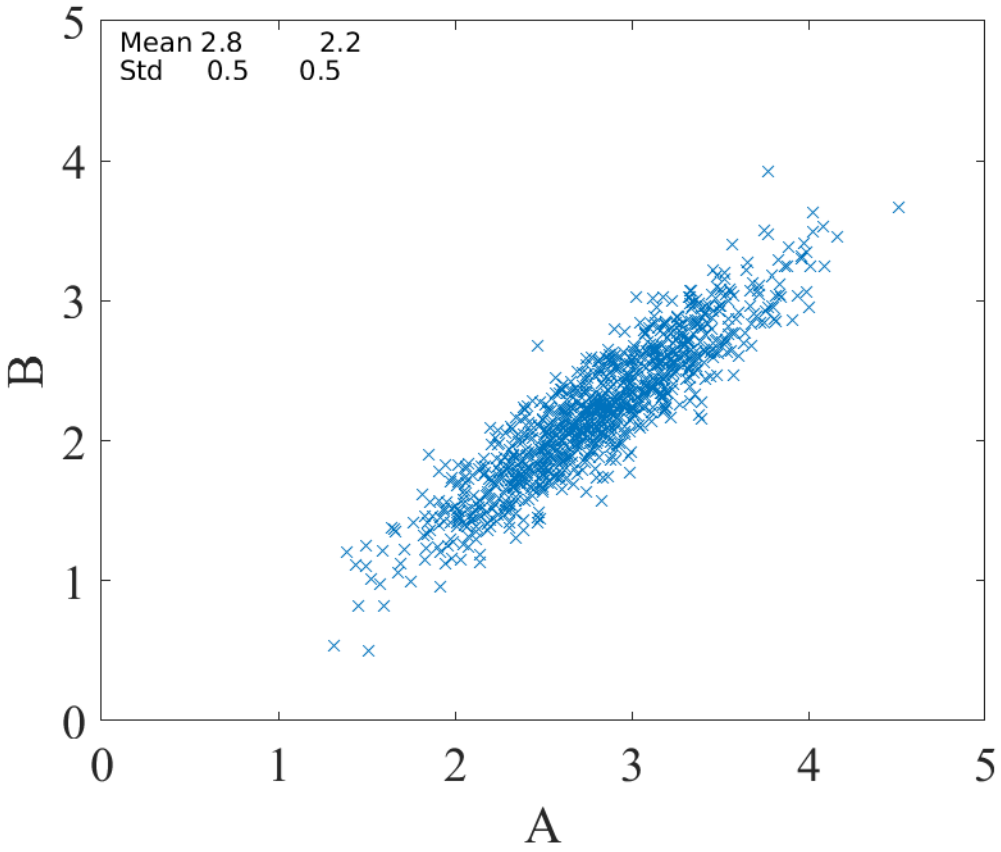
Scatter plot 1: Two uncorrelated measurements A and B.

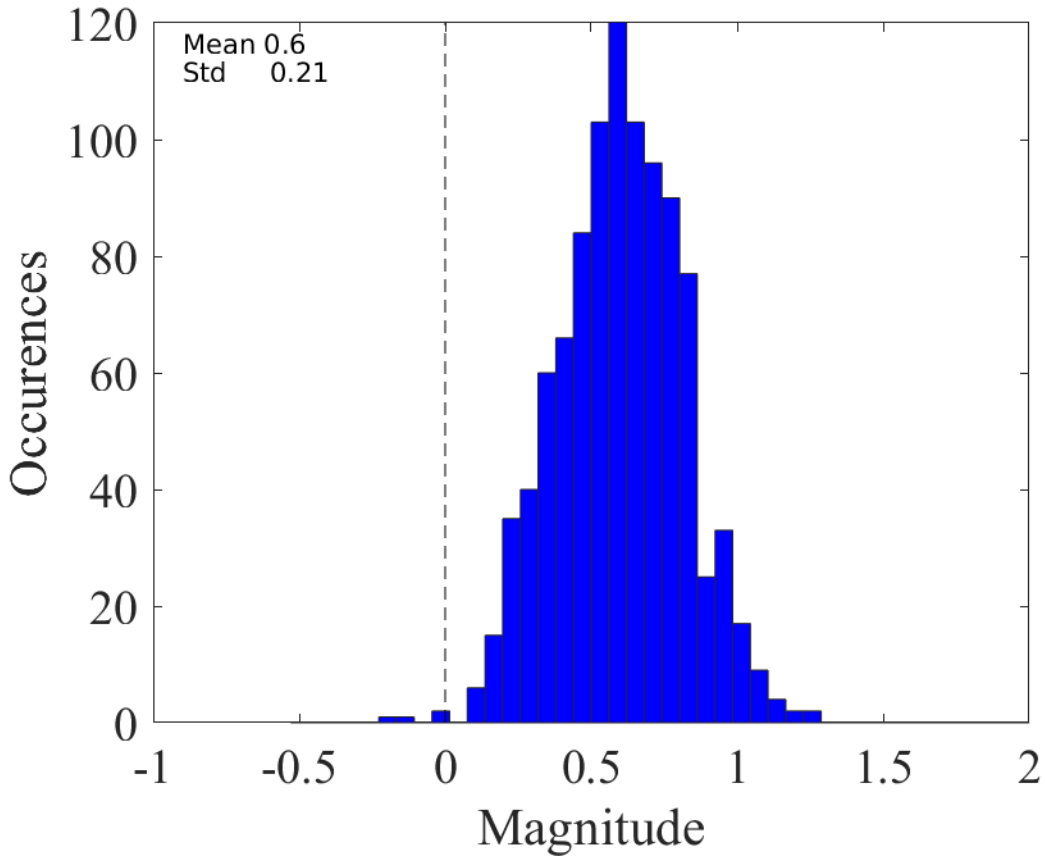



Scatter plot 3: Two somewhat separate groups with different means and correlations.




Scatter plot 2 illustrating two measurements with slightly differing means but high correlation (measurements form a tight group). In such a case, the difference between the two measurements can often be very informative, illustrated in the histogram below. We see that the mean difference is 0.6 with a standard deviation of 0.21 and that the distribution is clearly separated from 0 (=clearly separated from the null-hypothesis "no difference between A and B").


Mean 2.8 2.2
Std 0.5 0.5
B
A
0
1
2
3
4
5




### Box plots

For comparison of distributions of several different measurements, we often use box-plots (see Figure on the right). Here we have the sound samples on x-axis (here called "Items") and the improvement in POLQA scores on the y-axis (for more about delta scores, see section "Delta" below). The red line in the midle represent the median of all scores, the blue box contains 50% of all measurements and the black horizontal lines indicate the largest and lowest scores. In other words, each quartile which contains 25% of measurements, is indicated by an interval. An exception are *outliers*, that is, those measurements which are deemed *exceptional* in some sense, are marked with red plus-signs. Note that the choice of outliers is an sensitive issue; you cannot just choose which values are exceptional, but have to apply formal methods (it is an advanced topic which is not discussed here).

Note that box-plots are visualization methods and cannot be used alone to make conclusions about statistical significance. Instead, they valuable for descriptive analysis of results. For example in the figure on the right, female and male speakers are the three items on the left and right, respectively. Clearly we get an impression that the delta scores of the female speakers are higher than those of male speakers. However, the distributions are overlapping to such an extent that we would need proper statistical analysis to determine how likely it is that this is not just a coincidence.

In any case, box plots are useful in illustration of for example MUSHRA, POLQA and PESQ scores as well as their delta scores. Box plots contain less information than the corresponding histograms, but that omission reduces clutter; We can easily display 20 items side by side with a box-plot, whereas histograms of 20 items would be a mess.

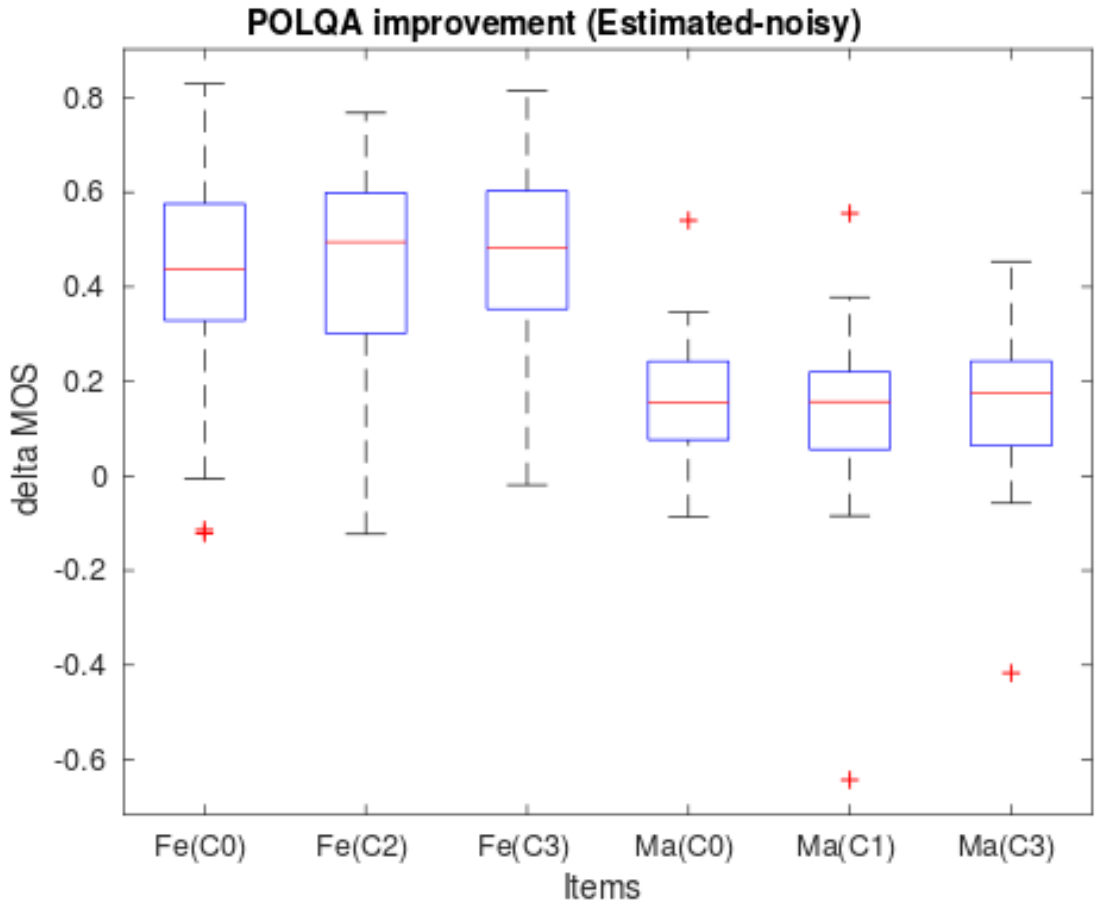



Image by Sneha Das (with permission)

### Confusion matrix

A confusion matrix is the equivalent of a scatter plot for categorical data, often either expressed in a table format or visualized as the as a heat-map. For example, suppose you have a classifier whose objective is to classify between speech, music and background noise. For each testing sample, you then have the target output and the actual output of your classifier, which you can write as a matrix such as:

| | Actual music | Actual speech | Actual noise |
|---|---|---|---|
| *Predicted music* | 78 | 2 | 13 |
| *Predicted speech | 3 | 77 | 5 |
| *Predicted noise* | 22 | 18 | 83 |

Often the largest value in each column is identified by bold-face. In this toy example, all classes are usually predicted correctly, which means that for each actual label, the most often predicted label matches the actual label. That is, music is most often classified as music (78 times), but also quite often as noise (22 times). Similarly, speech is classified most often as speech (77 times) but also often as noise (18 times). Noise, on the other hand, is rarely classified as speech (5 times) and almost always correctly as noise (83 times).

If the number of labels is large, then it is often useful to plot the matrix as a heat-map.

### Example 2

Suppose we want to compare methods A and B, and we have already calculated the frame-wise SNR on the outputs of both methods for a range of speech samples. We have also already calculated means and standard deviations and plotted histograms and scatter plots. The only we have found that there seems to be distinct categories of results indicated by a bimodal or multi-modal (multipeak) distribution of SNRs.

How do we know if A is better than B? How does behaviour of A differ from B? How do we characterize the differences between A and B?

**First idea:** In the previous example we already demonstrated the usefulness of illustrations. One of the all-time favourite illustrations is to plot the original sound signal (or its spectrogram) side-by-side with the frame-by-frame measurement results.

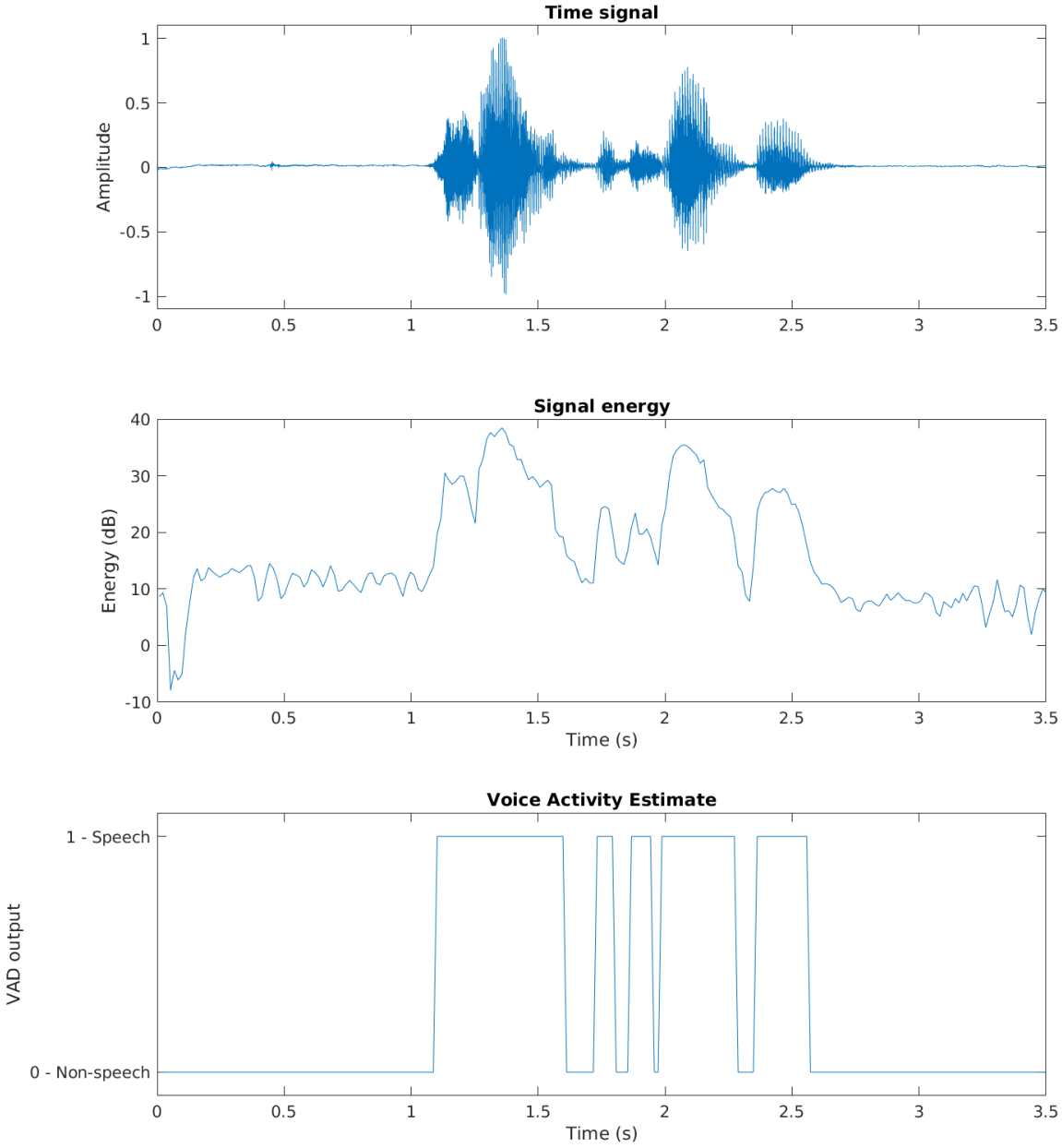
### Informative illustrations 2

### Parallel plots

When doing frame-by-frame measurements such as SNR, we can plot results over time to see whether there are any particular structures in the results. Typically, for example, it could be that the SNR is high for some continuous sections and poor somewhere else. Such cases could be for example voiced and unvoiced sounds, where the speech processing method behaves differently. However, to determine which regions correspond to which types of speech, we can plot the waveform (or spectrogram) in parallel with the SNR.

As an example, consider the output of a trivial voice activity detector (VAD) illustrated on the right. The top-pane contains the original waveform, the second pane signal energy and the third pane the thresholded energy as VAD output. We can immediately see that the VAD output is correct when signal energy is high. However, in low-energy parts of the speech signal, VAD output oscillates between speech and non-speech, even if it would make sense that the whole segment (between 1.1 and 2.7s) would be labelled speech. In the fourth pane, we see a comparison of the raw/original, desired and post-processed output methods (the results are shifted vertically for better clarity).

In this way, parallel plots are very good for illustrating and characterizing performance of a system. We can easily relate properties of the speech signal with the outputs, such that we can describe which types of inputs give which types of outputs.


Time signal
Amplitude
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5
Signal energy
Energy (dB)
40
30
20
10
0
-10
Time (s)
Voice Activity Estimate
VAD output
1 - Speech
0 - Non-speech
Time (s)

#### Extracting hidden structures

#### Deltas

People are often unable to accurately grade absolute quality, even if they can grade relative quality very accurately. That is, for example, in a MUSHRA test, it is perfectly normal that three subjects would grade two methods A and B such that:

| | | |
|---|---|---|
| Subject | Grade for A | Grade for B |
| 1 | 35 | 37 |
| 2 | 44 | 48 |
| 3 | 51 | 57 |
| Mean | 43.3 | 47.3 |
| Std | 8.0 | 10.0 |

In other words, if we look at the absolute mean value and its standard deviation, we find that B has a higher mean, but by only a tiny amount in comparison to the standard deviation. However, closer inspection reveals that the score of B is *always* higher than that of A. Every subject thought that B is better than A!

Thus if we calculate the difference in scores between A and B, perhaps that would give us a more conclusive answer.

| Subject | Grade for A | Grade for B | A-B |
|---|---|---|---|
| 1 | 35 | 37 | 2 |
| 2 | 44 | 48 | 4 |
| 3 | 51 | 57 | 6 |
| Mean | 43.3 | 47.3 | 4 |
| Std | 8.0 | 10.0 | 2 |

Clearly we now see that the difference A-B has a mean value of 4 and the standard deviation is only 2. By collecting more listeners, then it could be possible that this is significant difference between A and B.

The same approach can be applied also to other tests. For example, PESQ scores for two methods A and B can vary a lot between speech samples, but often the difference PESQ(A)-PESQ(B) can be much more consistent.

A formal way to check whether it is reasonable to use delta-values would be a correlation test (see below).

### 6.4.2 Statistical tests

**DISCLAIMER:** If you have not already, *you should* take a course in statistical analysis. There are plenty of good courses out there. In this short chapter we can only skim through some basic principles, which can not replace a thorough understanding of methods. Applying statistical tests without understanding them properly will *regularly lead to incorrect conclusions*.

### Student's t-test

When we want to determine if two normally distributed sets of values have the same mean, we use Student's t-test. To be applicable, it is critical that the sets really do follow the Gaussian distribution (see "Normality tests" below). The t-test is intended for small data-sets, usually with less than 30 data-points and it is not meaningful for larger sets. Typical applications in speech and audio are:

- Suppose we have made a MUSHRA-test, where we want to compare weather a new method B is better than the old method A. The mean of B is better (higher) than that of A, but how do we know if the result is statistically significant? First of all, we need to check that measurements follow the Gaussian distribution (see "Normality tests" below). If distributions are indeed Gaussian, we can apply the t-test. Given a threshold for significance (say 5%), the t-test gives an answer; either H=0, the difference between A and B is not significant or, H=1, that B is better than A.
- Similarly, we can calculate the difference between A and B, and if the distribution is Gaussian, we can determine whether the difference to zero is statistically significant.

If the data sets are not normally distributed, it is often possible to use the Wilcoxon rank-sum test or the Wilcoxon signed-rank test instead.

### Normality tests

Many statistical tests are only applicable when the input signal follows a Gaussian distribution. To test whether a distribution indeed is Gaussian, we use a normality test. The Shapiro-Wilk test is one particular test of normality, which is particularly reliable. The output of normality tests is that, given a particular level of confidence (such as 5%), the given data set is Gaussian.

### ANOVA

When comparing multiple data sets at the same time, analysis of variance (ANOVA) can be interpreted as a generalization of the t-test. It tries to explain phenomenons in the data by assigning them to different sources of variation. Say if we observe differences between methods A and B, but know that some listeners have taken the test in the morning and others in the evening, we can analyse whether the difference is due to the the time of day or due to inherent differences in A and B.

Again it is important to observe the assumptions of this test; namely, the classical version of ANOVA assumes that prediction errors are normal, that measurements are independent and the assigned groups of data have uniform statistics.

### Correlation tests

Often two data sets are correlated. For example, if we analyse the fundamental frequency and intensity of a speech signal, we probably find that shouted speech has a high intensity and fundamental frequency, whereas silent speech has low intensity and a low fundamental. To check whether two data sets indeed have a meaningful correlation, we use correlation tests. The two most common correlation tests are

- Pearson correlation coefficient (PCC), which essentially fits a linear regression line through the data, such that we can then use the t-test to determine whether the thus-explained data is significantly different from the original. In other words, the Pearson correlation coefficient assumes a linear correlation between variables.
- Spearman's rank correlation coefficient analyses whether data can be explained by a monotonic function.

CHAPTER

SEVEN

# SPEECH ANALYSIS

- *Fundamental frequency estimation*
- Formant estimation and tracking
- *Measurements for medical applications*
- *Inverse filtering for glottal activity estimation*

## 7.1 Fundamental frequency estimation

The fundamental frequency (F0) is central in describing speech signals whereby we need methods for estimating the $F_0$ from speech signals. In speech analysis applications, it can be informative to study the absolute value of the fundamental frequency as such, but more commonly, extraction of the $F_0$ is usually a pre-processing step. For example, in recognition tasks, $F_0$ is often used as a feature for machine learning methods. A voice activity detector could, for instance, set a lower and higher threshold on the $F_0$, such that sounds with an $F_0$ outside the valid range would be classified as non-speech.

The fundamental frequency is visible in multiple different domains:

- In an acoustic time-signal, the $F_0$ is visible as a repetition after every $T$ samples.
- In the autocovariance or -correlation, the $F_0$ is visible as a peak at lag $T$ as well as its integer multiples $kT$.
- In the magnitude, power or log-magnitude spectrum, the $F_0$ is visible as a peak at the frequency $F_0 = F_s/T$, as well as its integer multiples, where $F_s$ is the sampling frequency.
- In the cepstrum, the $F_0$ is visible as a peak at quefrency $T$ as well as its integer multiples $kT$.

Consequently, we can use any of these domains to estimate the fundamental frequency $F_0$. A typical approach applicable in all domains except the time-domain, is peak-picking. The fundamental frequency corresponds to a peak in each domain, such that we can determine the $F_0$ by finding the highest peak. For better robustness to spurious peaks and for computational efficiency, we naturally limit our search to the range of valid $F_0$'s, such as $80 \leq F_0 \leq 450$.

The harmonic structure however poses a problem for peak-picking. Peaks appear at integer multiples of either $F_0$ or lag T, such that sometimes, by coincidence or due to estimation errors, the harmonic peaks can be higher than the primary peak. Such estimation errors are known as *octave errors*, because the error in $F_0$ corresponds to the musical interval of an octave. A typical post-processing step is therefore to check for octave jumps. We can check whether $F_0/2$ or $F_0/3$ would correspond to a sensible $F_0$. Alternatively, we can check whether the previous analysis frame had an $F_0$ which an octave or two octaves off. Depending on application, we can then fix errors or label problematic zones for later use.

Another problem with peak-picking is that peak locations might not align with the samples. For example in the autocorrelation domain, the true length of the period could be 100.3 samples. However, the peak in the autocorrelation would then appear at lag 100. One approach would then be to use quadratic interpolation between samples in the vicinity of a peak and use the location of the maximum of the interpolated peak as an estimate of the peak location. Interpolation makes the estimate less sensitive to noise. For example, background noise could happen to have a peak at lag 102, such

that desired maximum of 100 is lower than the peak at 102. By using data from more samples, as in the interpolation approach, we can therefore reduce the likelihood that a single corrupted data point would cause an error.

An alternative approach to peak-picking is to use all distinctive peaks to jointly estimate the $F_0$. That is, if you find $N$ peaks at frequencies $p_k$, which approximately correspond to harmonic peaks $kF0$, then you can approximate $F_0 \approx \frac{1}{N}\sum_{k=1}^{N} p_k/k$. Another alternative is to calculate the distance between consecutive peaks an estimate $F_0 \approx \frac{1}{N-1}\sum_{k=1}^{N-1}(p_{k+1} - p_k)$. We can also combine these methods as we like.

In many other applications, we use discrete Fourier transform (DFT) or cosine transforms (DCT) to resolve frequency components. It would therefore be tempting to apply the same approach also here. In such a domain we would also already have a joint estimate which does not rely on single data points. However, note that the spectrum is already the DFT of the time signal and the cepstrum is the DCT or DFT of the log-spectrum. Additional transforms therefore usually do not resolve the ambiguity between harmonics.

## 7.2 Measurements for medical applications

1. Electroglottography (Wikipedia)
2. Stroboscopy and videokymography (Wikipedia)
3. Highspeed camera
4. MRI
5. Rothenberg mask
6. *Inverse filtering for glottal activity estimation*

### 7.2.1 Inverse filtering for glottal activity estimation

#### Motivation

The main source of speech sounds, where the acoustic signal is first created, are the vocal folds, which oscillate in the airflow generated by the lungs (for details, see Speech production and acoustic properties). The opening between the vocal folds is known as the *glottis* and the movements of vocal folds as well and the corresponding airflow are jointly known as glottal activity.

Since glottal activity is thus central to speech, it is also important to understand how it works. What makes a "good" voice? If there is a disruption to the vocal folds, how does that affect the voice? These are often medical questions, but they have a large social and societal impact. If you loose your voice, you can easily become isolated since you loose an important mode of communication. If you are in a voice profession such as teaching, sales, singing or acting, also your ability to work relies on your voice such that any disturbance in the voice impedes your ability to work. Studying the glottis is thus paramount.

The vocal folds are located in the neck, covered and surrounded by a cartilage. Accessing them is therefore difficult. Putting a camera in the throat is uncomfortable to say the least and even when it is possible, it impedes normal speech, giving measurements a bias of unknown size. Moreover, since the vocal folds oscillate with a fundamental frequency that can go up to 400 Hz, we need a camera whose frame rate is at least 4000 Hz to get 10 frames per period. In other words, we would need a high-speed camera. While such cameras are today readily available, they need a lot of light, which generates a lot of heat, which is not compatible within the sensitive tissues inside the throat. Imaging with other methods, X-rays or magnetic resonance imaging, generally have a slower frame rate and some imaging like X-rays also generate harmful radiation (esp. at high frame rates).

The cartilage surrounding the vocal folds also prevents ultrasound measurements. The only usable direct measurement is electroglottography (EGG), which measures the impedance through the neck using electrodes. It measures conductivity, which is highly dependent on the contact area of the vocal folds, thus giving information about the position of the vocal

folds. However, this information is usually one-dimensional which limits the usability of such measurements. It also sensitive to and requires careful placement of electrodes.

What remains is the acoustic signal. With a microphone, we can record the sound emitted from the mouth, and try to deduce the movements of the vocal folds from the sound. It is minimally invasive, because we do not need to insert any sensors inside or onto the the body. Airflow through the glottis is closely related to the movements of the vocal folds; when the vocal folds are open air can flow and when they are closed, airflow is stopped. Airflow in turn is related to the acoustic signal; sound is a variation in air pressure such that the gradient of airflow approximately translates to the corresponding sound.

When sound is generated in the vocal folds, it is however acoustically shaped by the vocal tract (for details, again, see Speech production and acoustic properties); some frequencies are emphasised and others attenuated. To analyse glottal activity, we therefore need to cancel the acoustic effect of the vocal tract. Recall that the effect of the vocal tract can be efficiently modelled by a linear predictive filter. We can thus estimate a filter corresponding to the effect of the vocal tract, and then invert the effect of that filter. This process is known as *inverse filtering*.

### Signal Model

In glottal inverse filtering, we follow the source-filter paradigm for speech source modelling, where the acoustic speech signal is modelled as an excitation signal filtered by the vocal tract. The assumption is that these two components are independent. By assuming that we know the effect of the vocal tract, we can therefore remove its effect by inverting its effect. If we further model the vocal tract as a tube-model, its effect corresponds to IIR filtering, such that the inverse process is FIR filtering.

The main difficulty in of glottal inverse filtering is estimation of the filter corresponding to the effect of the vocal tract. The task resembles classical linear predictive modelling, where the parameters of an IIR filter are uniquely estimated from the autocorrelation of the signal. Covertly, this however assumes that the excitation is uncorrelated white noise. However, in voiced speech, the excitation is the glottal excitation, which resembles a half-wave rectified sinusoid. That is, it is a fairly smooth curve with a characteristic comb-structure in the spectrum, as well as a distinct tilt with more energy at low frequencies. Though we know a lot about the glottal excitation, it is hard to model. If we would have the glottal excitation, then the vocal tract would be easy to estimate and vice versa - a typical chicken-and-egg problem.

One of the most popular methods for glottal inverse filtering is iterative adaptive inverse filtering (IAIF), where both the glottal excitation and the vocal tract are modelled with linear predictive filtering, and both filters are estimated in an alternating iteration. Many improvements based on more advanced signal models as well as machine learning have been proposed, but the question is far from solved. A central problem in evaluation such methods is the ground truth. We can estimate curves which look like glottal excitations, but we would need to know the actual movements of the vocal folds to verify the accuracy of the obtained curves. Since we do not have a satisfactory direct method for observing glottal activity, which would not bias results, we cannot verify our models.

## 7.3 Forensic analysis

CHAPTER

EIGHT

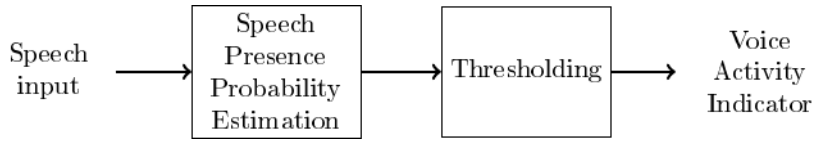
# RECOGNITION TASKS IN SPEECH PROCESSING

Typical tasks in speech processing, where machine learning is often applied include:

- *Speech recognition*, which refers to converting an acoustic waveform of spoken speech to the corresponding text (speech-to-text).
- *Speaker recognition* and *speaker verification,* which refer to, respectively, identifying the speaker (who is speaking?) and verifying whether the speaker is who he claims to be (is it really you?).
- *Speech synthesis*, which entails the creation of a natural sounding speech signal from text input (text-to-speech).
- Speech enhancement, refers to improving a recorded speech signal, for example with the objective of removing background noise (noise attenuation) or the effect of room acoustics.
- Wake-word *and* keyword detection, refers to the task where the purpose is to find single characterizing words from continuous speech. The idea is that by using a light-weight algorithm, we can extract useful information without a computationally complex speech recognizer. Specifically, wake-word detection refers to the waiting for the activation command, that is, the device sleeps until the wake-word is heard. Keyword detection can refer to similar task, or for example, the task of recognizing the topic of a conversation.
- Voice activity detection *(VAD)*, refers to the task of determining whether a signal contains speech or not (is someone speaking?). Many of the above tasks are resource-intensive operations, such that we would like to, for example, use speech recognition only when speech is present. We can therefore first use a simple VAD to determine whether a signal is speech or not, and only start the speech recognizer when speech is present.
- *Speech diarisation* is the process of segmenting a multi-speaker conversation into continuous single-speaker segments.
- *Paralinguistic analysis tasks*, refers generally to the extraction of non-linguistic and non-speaker identity related information from speech signals, such as speaker emotions, health, attitude, sleepiness etc.

## 8.1 Voice Activity Detection (VAD)

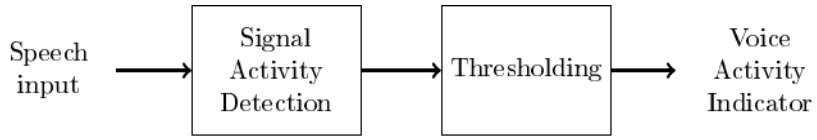
### 8.1.1 Introduction



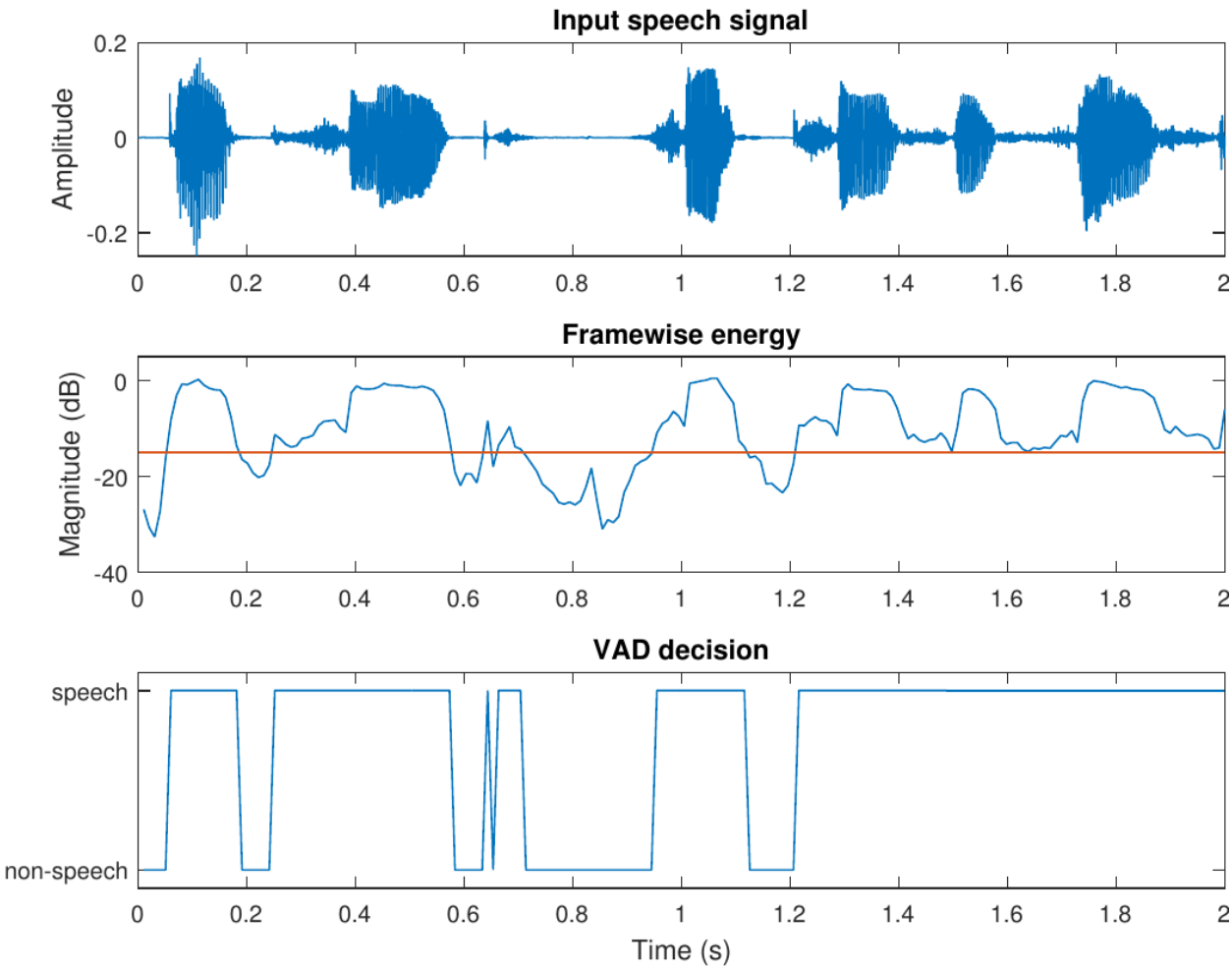

- *Voice activity detection* (VAD) (or speech activity detection, or speech detection) refers to a class of methods which detect whether a sound signal contains speech or not.
- A closely related and partly overlapping task is *speech presence probability* (SPP) estimation.

  - Instead of a binary present/not-present decision, SPP gives a probability level that the signal contains speech.
  - A VAD can be derived from SPP by setting a threshold probability above which the signal is considered to contain speech.
  - In most cases, SPP is thus the more fundamental problem.
- Voice activity detection is used as a pre-processing algorithm for almost all other speech processing methods.
  - In *speech coding*, it is used to to determine when speech transmission can be switched off to reduce the amount of transmitted data.
  - In *speech recognition*, it is used to find out what parts of the signal should be fed to the recognition engine. Since recognition is a computationally complex operation, ignoring non-speech parts saves CPU power.
- VAD or SPP is thus used mostly as a resource-saving operation.
- In *speech enhancement*, where we want to reduce or remove noise in a speech signal, we can estimate noise characteristics from non-speech parts (learn/adapt) and remove noise from the speech parts (apply).
- A closely related method in *audio* applications is *noise gateing*, where typically a microphone signal is muted whenever there is no signal present.
  - For example, when a singer is not singing in the microphone, then the microphone is off. When the singer is not singing, microphone signal is only noise and therefore the noise gate removes (gates) noise.
- VADs can thus also be used in improving signal quality.

### 8.1.2 Low-noise VAD = Trivial case




- To introduce basic vocabulary and methodology, let us consider a case where a speaker is speaking in an (otherwise) silent environment.
  - When there is no speech, there is silence.
  - (Any) Signal activity indicates voice activity.
- Signal activity can be measured by, for example, estimating signal energy per frame $\Rightarrow$ the *energy thresholding algorithm*.




- Clearly energy thresholding works for silent speech signals.
  - Low-energy frames are correctly labeled as non-speech and speech parts are likewise correctly labeled.
- It is however not trivial to choose an appropriate threshold-level.
  - A low threshold level would make sure that all speech-frames are correctly labeled. However, we might then also label frames with other sounds, like breathing sounds or other background noises, as speech frames.
  - A high threshold would make sure that all detected speech-frames actually are truly speech frames. But then we could miss offsets (sounds which are trailing off), since they often have a low energy.
- What strategy should we use to choose a threshold?
  - What is the correct label for something like breathing-noises?
  - How do we actually measure performance of a VAD?

### 8.1.3 VAD objective and performance measurement

- The objective of a VAD implementation depends heavily on the application.
  - In speech coding, our actual objective is to reduce bitrate without decreasing quality. ⇒ We want to make sure that no speech frames are classified as background noise, because that would reduce quality.

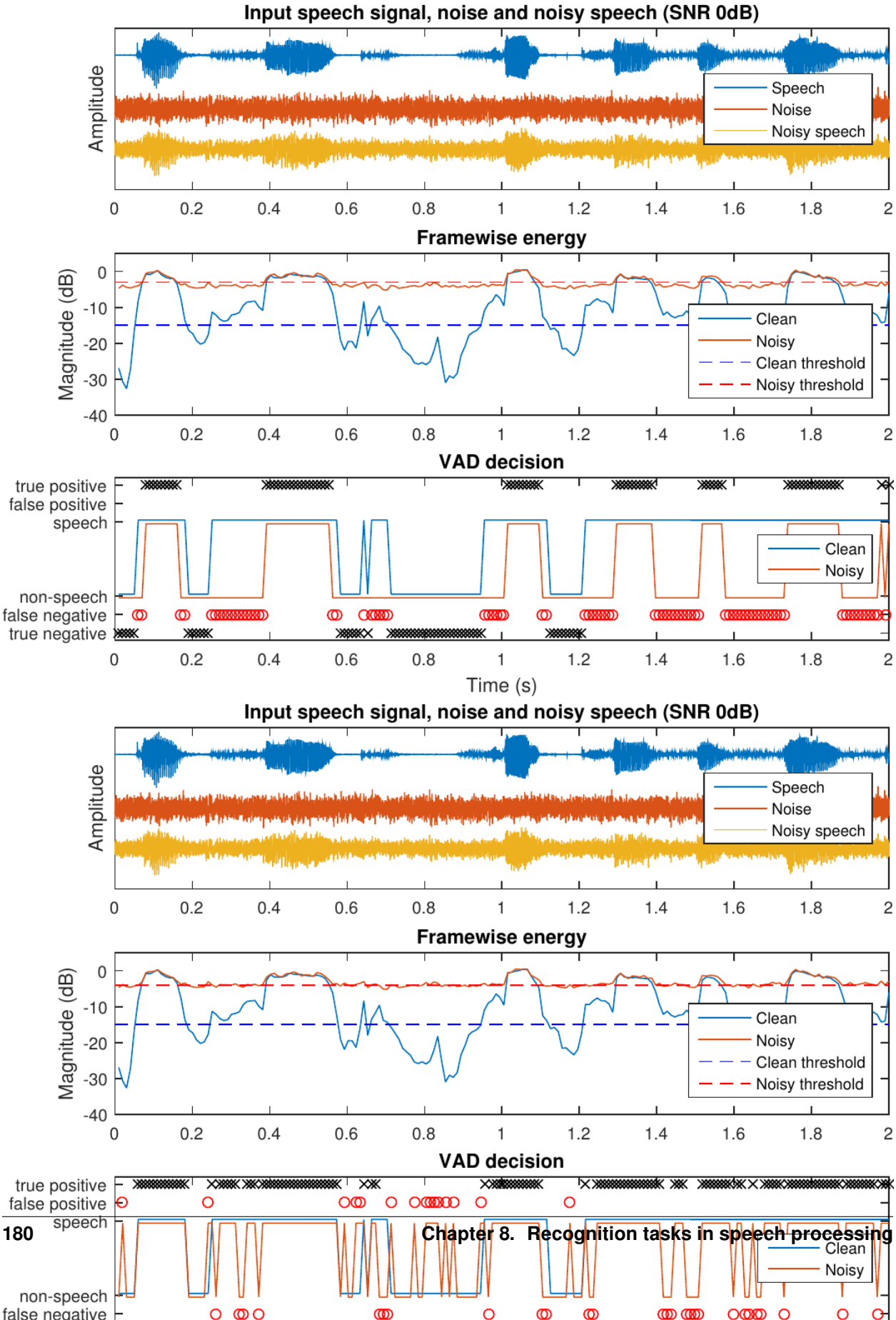
    ⇒ We make a conservative estimate.
  - In keyword spotting (think "Siri" or "OK Google"), we want to detect the start of a particular combination of words. The VADs task is to avoid running a computationally expensive keyword spotting algorithm all the time. Missing one keyword is not so bad (the user would then just try again), but if it is too sensitive then the application would drain the battery.

    ⇒ We want to be sure that only keywords are spotted.
- The objective of a VAD implementation depends heavily on the application.
  - In speech enhancement, we want to find non-speech areas such that we can there estimate noise characteristics, such that we can remove anything which looks like noise. We want to be sure that there is no speech in the noise estimate, otherwise we would end up removing some speech and not only noise.
  - In speech recognition, VAD is used purely for resource saving. We do not want to reduce accuracy of the recognition, but want to minimize CPU usage.
- We need a set of performance measures which reflect these different objectives.
- The performance is then often described by looking at how often are frames which do contain speech labeled as speech/non-speech, and how often is non-speech labeled as speech/non-speech?

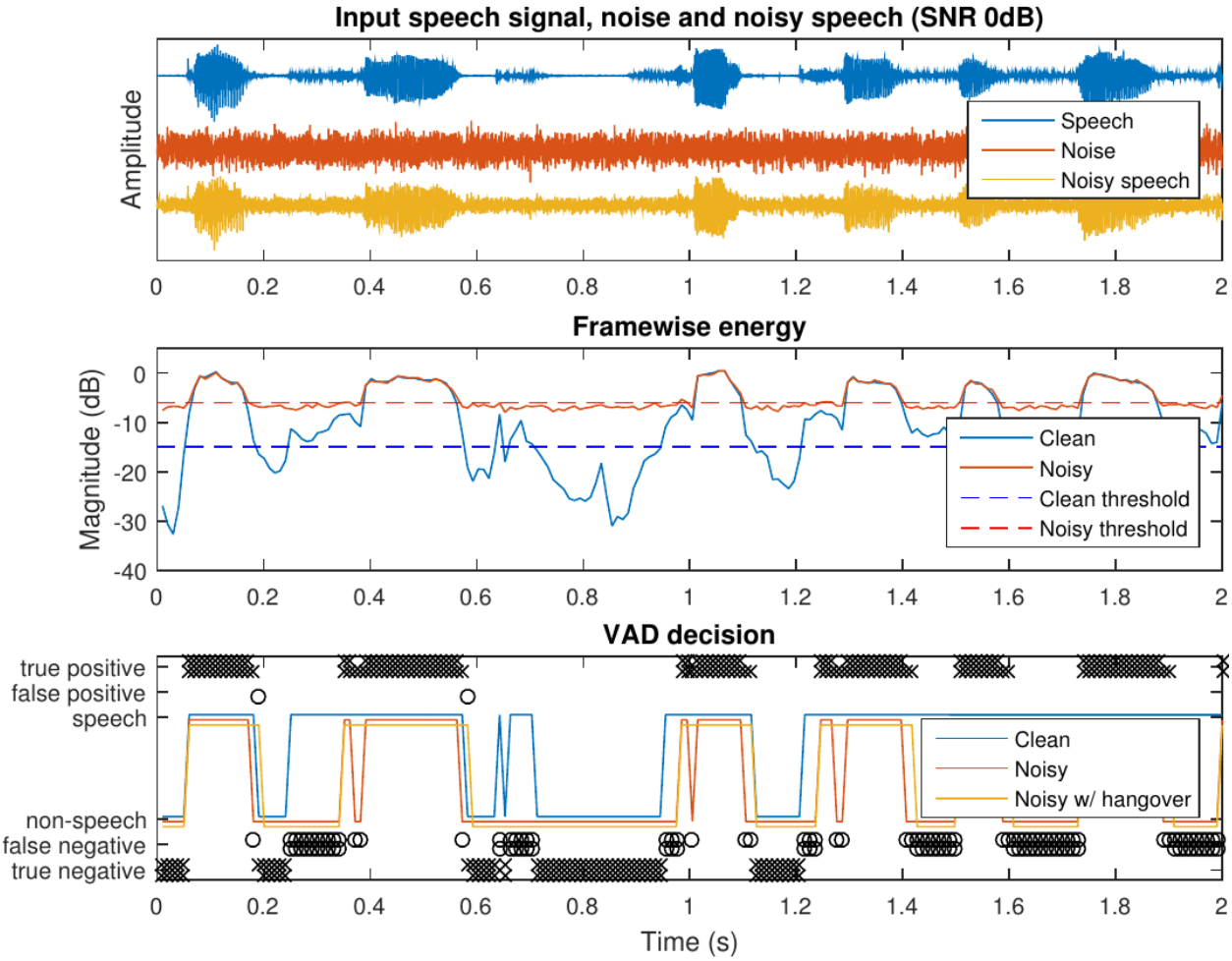
| Input | Speech | Non-speech |
|---|---|---|
| Speech | True positive | False negative |
| Non-speech | False positive | True negative |

- For speech coding, we want to keep the number of false negatives low, and false positives are of only secondary importance.
- For keyword spotting, we want to keep the number of false positives low, and false negatives are secondary importance.

### Performance in noise – -3dB / -4dB threshold



### 8.1.4 Post-processing

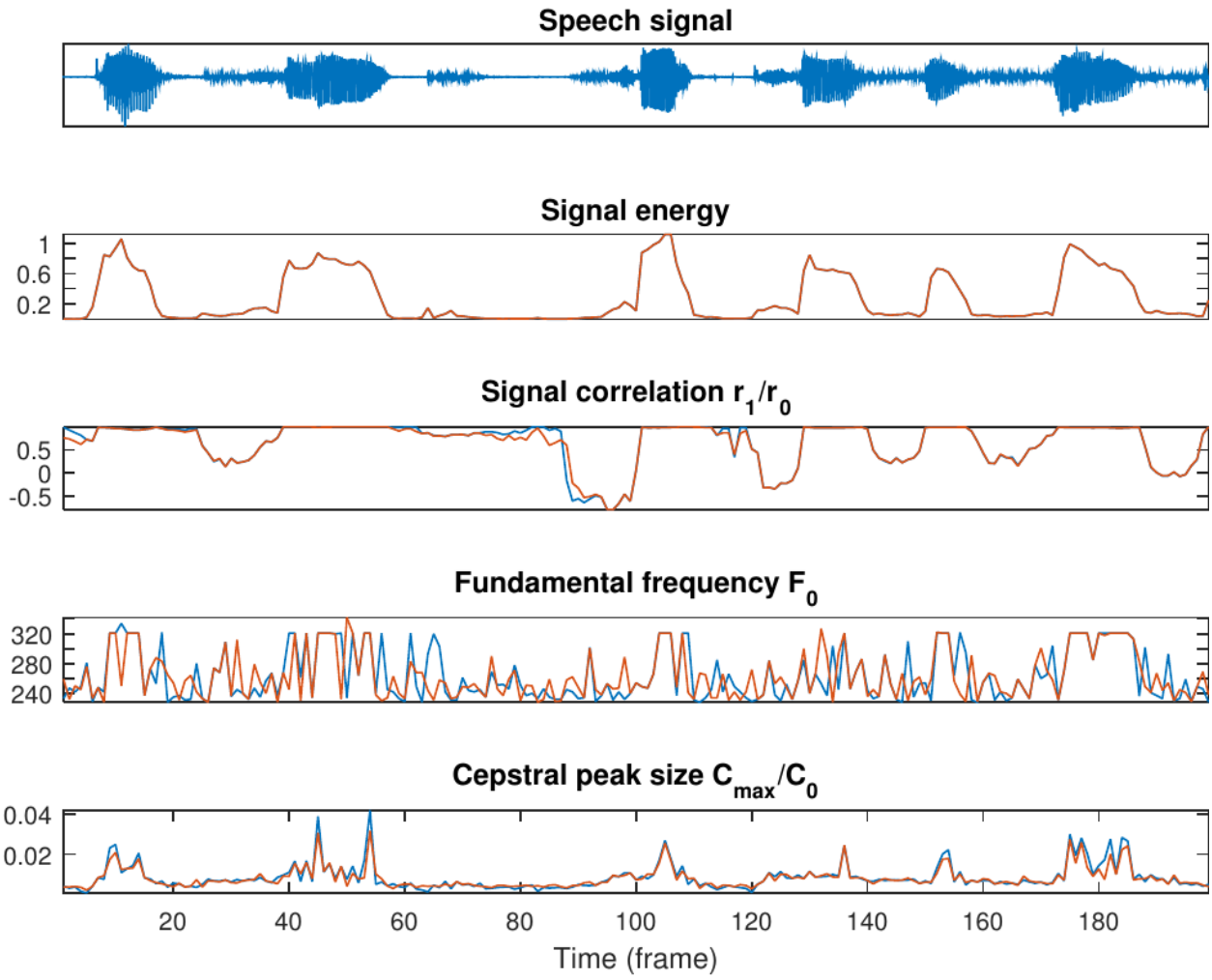



- We already saw that speech coding wants to avoid false negatives (=speech frames labeled as non-speech).
- Can we identify typical situations where false negatives occur?
  - Offsets (where a phonation ends) often have low energy⇒ Easily misclassified as non-speech.
  - Stops have a silence in the middle of an utterance.⇒ Easily misclassified as non-speech.
- We should be careful at the end of phonations.
  - We can use a *hangover* time, such that after a speech segment we keep the label as speech for a while until we are sure that speech has ended.
  - For onsets (starts of phonemes) we usually want to be very sensitive.
- We obtain a hysteresis rule;
  - If any of the last $K$ frames was identified as speech, then the current frame is labelled as speech. Otherwise non-speech.

### 8.1.5 VAD for noisy speech

- Clean speech (absolutely no background noise) is very rare if not impossible to achieve.
- Real-life speech recordings practically always have varying amounts of background noise.
  - Performance of energy thresholding decreases rapidly when the SNR drops.
  - For example, weak offsets easily disappear in noise.
- We need more advanced VAD methods for noisy speech.
  - We need to identify characteristics which differentiate between speech and noise.
  - Measures for such characteristics are known as *features*.

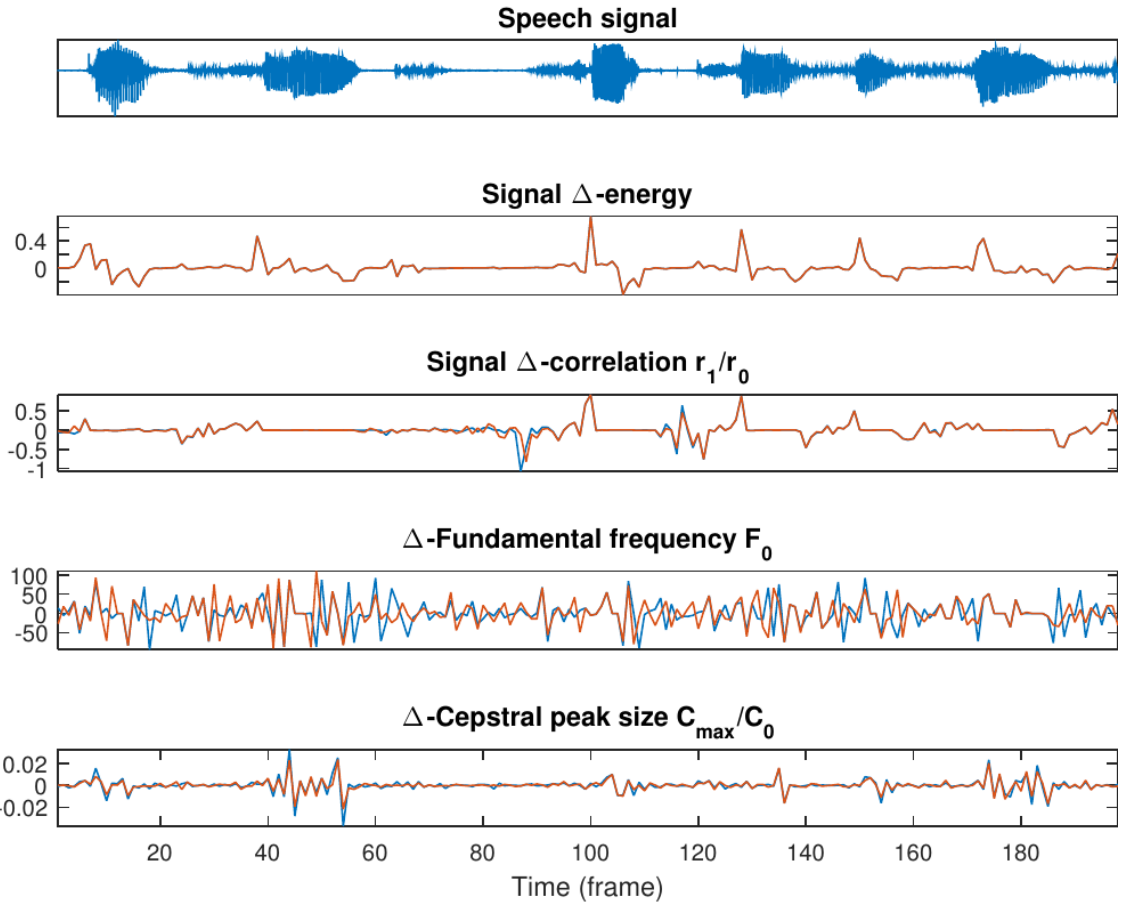
#### Features




- In VAD, with features we try to measure some property of the signal which would give an indication to whether the signal is speech or non-speech.
  - Signal energy is naturally a useful feature, since the energy of speech varies a lot.
  - Voiced sounds generally have energy mainly at the low frequencies, whereby estimators for spectral tilt are often useful. For example,

    * Zero-crossings (per time unit) is high for high-frequency signals (noise) and low for low-frequency signals (voiced speech), whereby it can be used as a feature.
    * The lag-1 autocorrelation is high (close to one) for low-frequency signals and low (close to -1) for high-frequency signals.

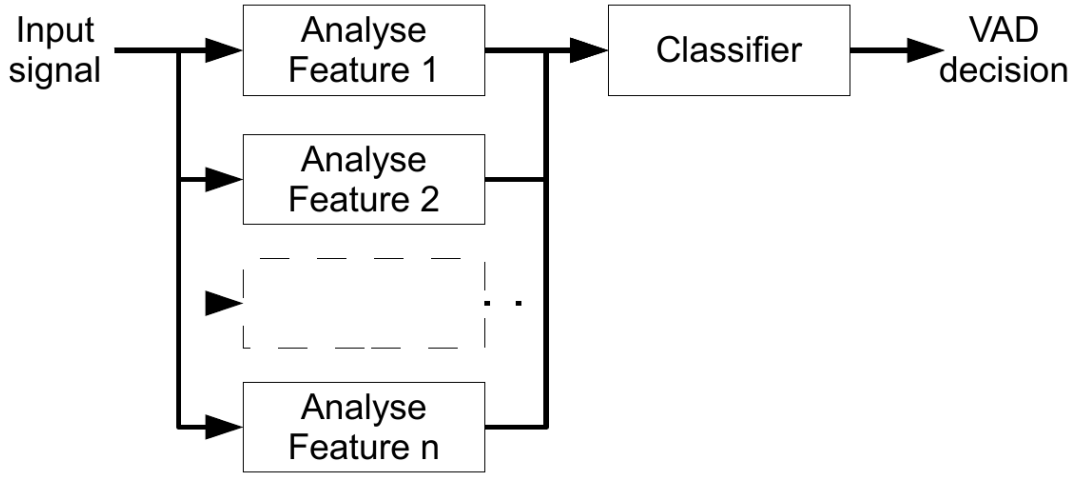
- Speech sounds can be efficiently modelled by linear prediction.
  - If the prediction error is small, then it is likely that the signal is speech.
  - If the prediction error is large, then it is probably non-speech.
- Voiced speech has by definition a prominent pitch.
  - If we can identify a prominent pitch in the range then it is likely voiced speech.
- Speech information is described effectively by their spectral envelope.
  - MFCC can be used as a description of envelope information and it is thus a useful set of features.
  - Linear prediction parameters (esp. prediction residual) also describe envelope information and can thus also be used as a feature-set.




- Speech features vary rapidly and frequently.
  - By looking at the rate of change $\Delta_k = f_{k+1} - f_k$ in other features $f_k$, we obtain information about the rate of change of the signal. (Estimate of derivative)
  - Likewise, we can look at the second difference $\Delta\Delta_k = \Delta_{k+1} - \Delta_k$. (Estimate of second derivative)
  - These first and second order differences can be used as features and they are known as $\Delta$- and $\Delta\Delta$-features.

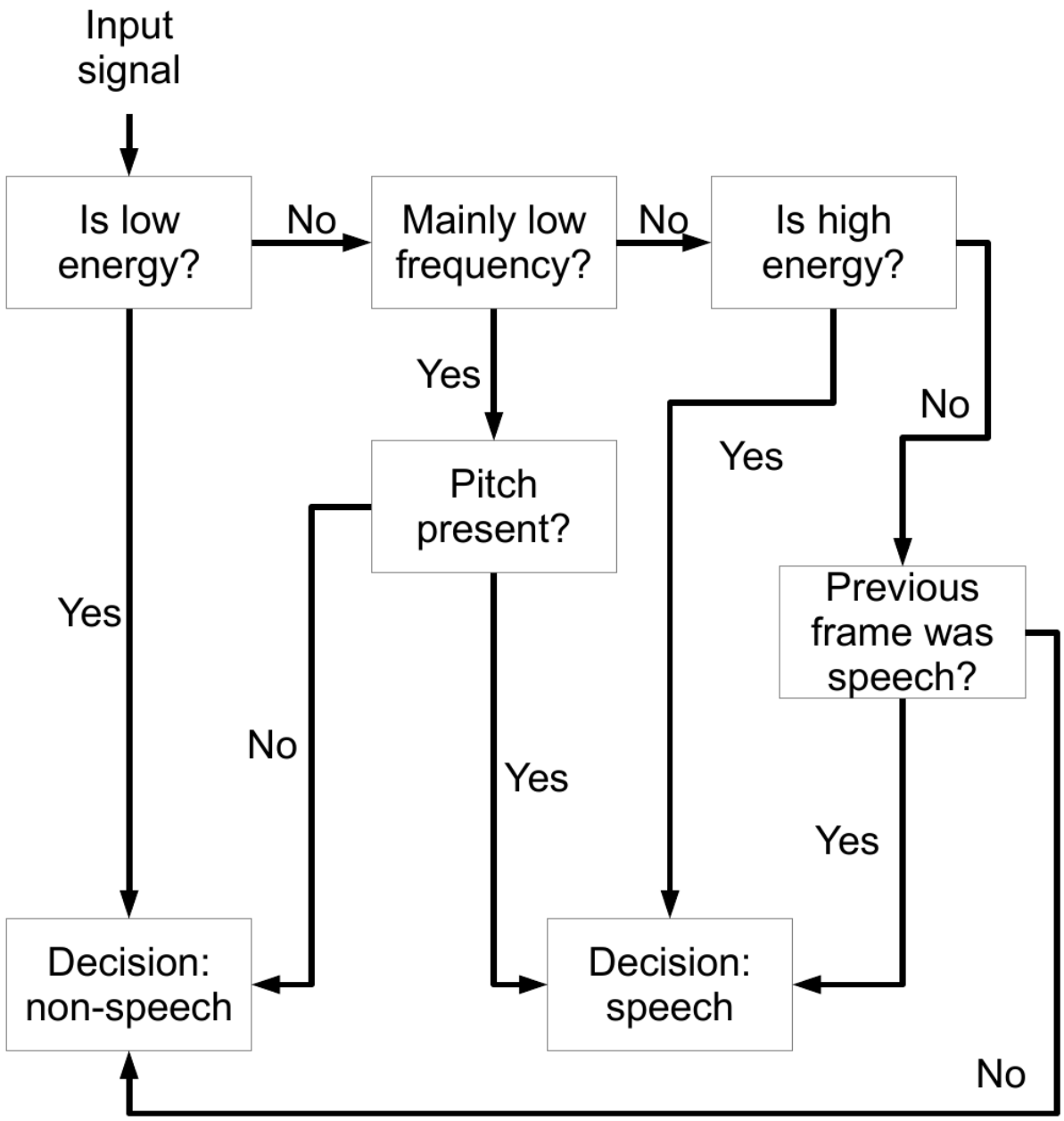
### 8.1.6 Classifier

- We have collected a set of indicators for speech, the features, whereby the next step is to merge the information from these features to make a decision between speech and non-speech.
- Classification is generic problem, with plenty of solutions such as
  - decision trees (low-complexity, requires manual tuning)
  - linear classifier (relatively low-complexity, training from data)
  - advanced methods such as neural networks, Gaussian mixture models etc. (high-complexity, high-accuracy, training from data)




#### Decision trees (historical)

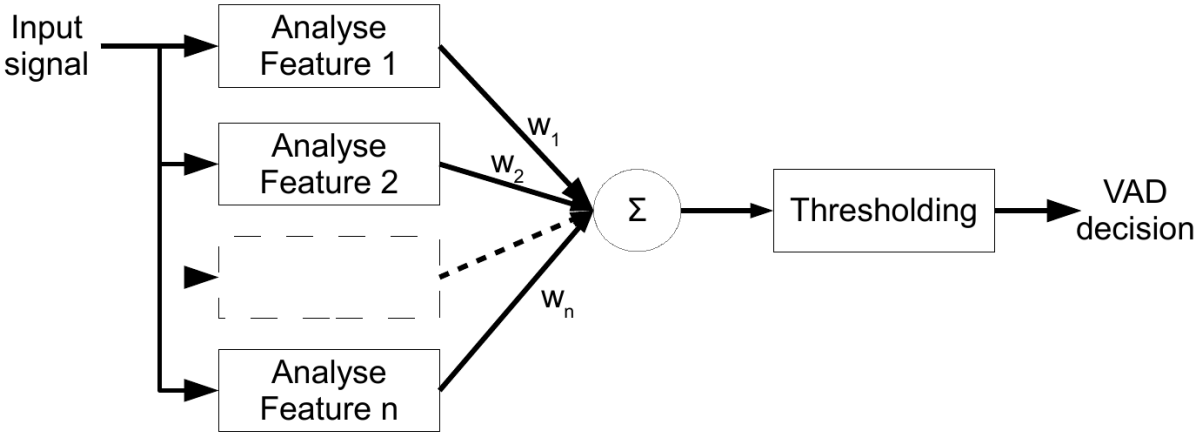
- Make a sequence of binary decisions (for example, is low or high energy?) to decide whether signal is speech or non-speech.
- Decision trees are very simple to implement.
- Hard-coded ⇒ not very flexible.
- Noise in one feature can cause us to follow wrong path.
  - One noisy feature can break whole decision tree.
- Requires that each decision is manually tuned⇒ Lots of work, especially when tree is large
- Structure and development becomes very complex if the number of features increase.
- Suitable for low-complexity systems and low-noise scenarios where accuracy requirements are not so high.
- I did not prepare an illustration/figure of result.




### Linear classifier

- Instead of manually-tuned, binary decisions, can we use observed data to make a statistical estimate?
  - Using training data would automate the tuning of the model.Accuracy can be improved by adding more data.
  - By replacing binary decisions, we can let tendencies in several features improve accuracy.
- Linear classifiers attempt to achieve a decision as a weighted sum of the features.
  - Let $\xi_k$ be the features.

  - The decision is then obtained by $\eta = \sum_k \omega_k \xi_k$, where $\omega_k$ are scalar weights.
  - The objective is to find weights $\omega_k$ such that $\eta = \begin{cases} -1 & \text{non-speech} \\ +1 & \text{speech.} \end{cases}$
- We then need to develop a method for choosing optimal weights $\omega_k$.
- The first step is to define an *objective function*, which we can minimize.
  - A good starting point is the classification error.
  - If $\eta$ is the desired class for a frame and our classifier gives $\hat{\eta}$, then the classification error is $\nu^2 = (\eta - \hat{\eta})^2$.
  - By minimizing the classification error, we can determine optimal parameters $\omega_k$.
- Let $x_k$ be the vector of all features for a frame $k$ and $X = [x_0, x_1 \ldots]$ a matrix with all features for all frames.
  - The classification of a single frame is then $\eta_k = x_k^T w$.
  - The classification of all frames is then a vector $y = X^T w$, where $w$ is the vector of weights $\omega_k$.
  - The sum of classification errors is the norm $\|y - \hat{y}\|^2$.

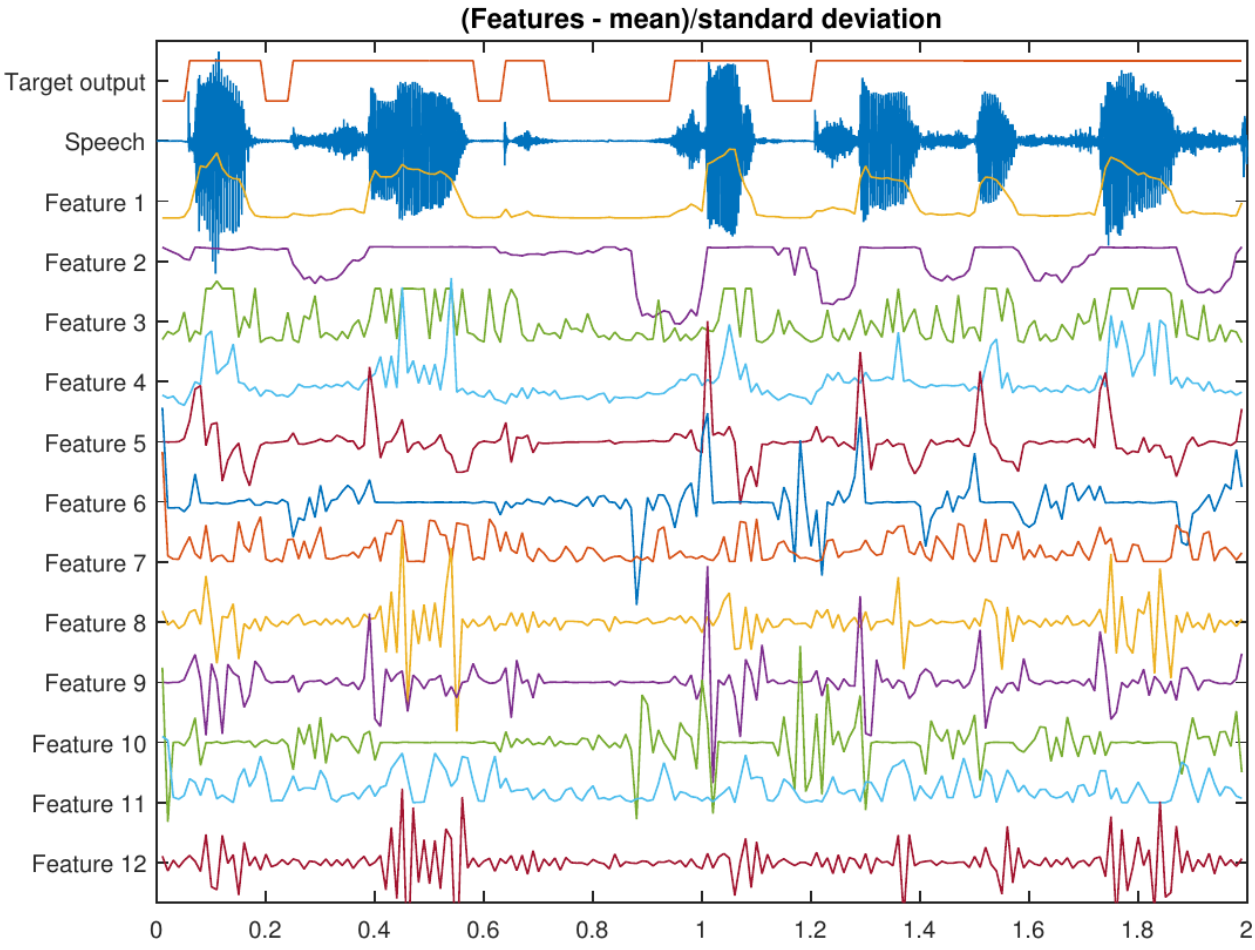


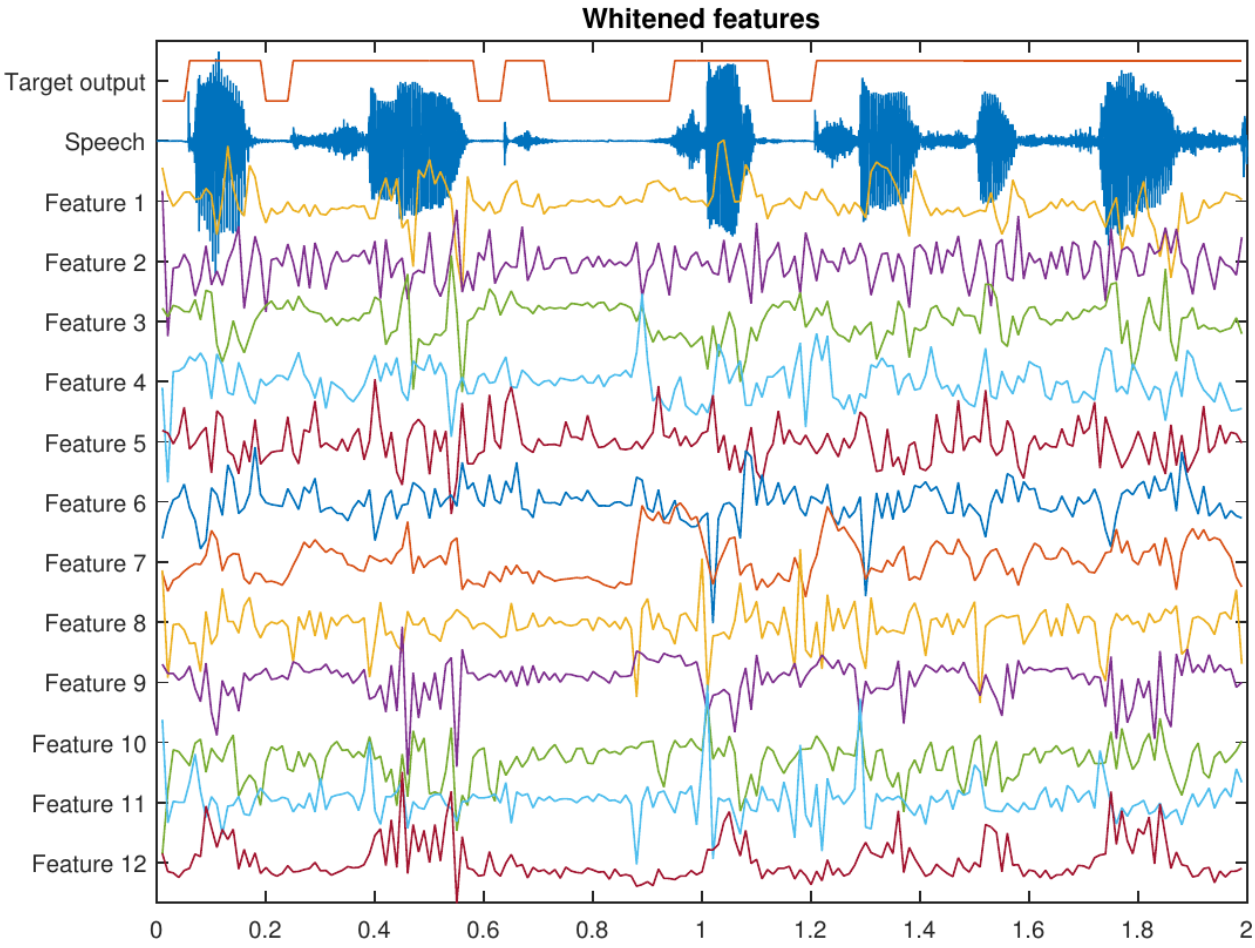

### A bit of math

- The minimum of the classification error $\|y - \hat{y}\|^2$ can be found by setting the partial derivative to zero. $$
- The solution is the Moore-Penrose pseudo-inverse $w = (XX^T)^{-1} X y^T := X^\dagger y.$
- *Note:* This is a very common mathematical approach for solving problems in speech processing, so it is much more important and broadly applicable than "only" VAD.

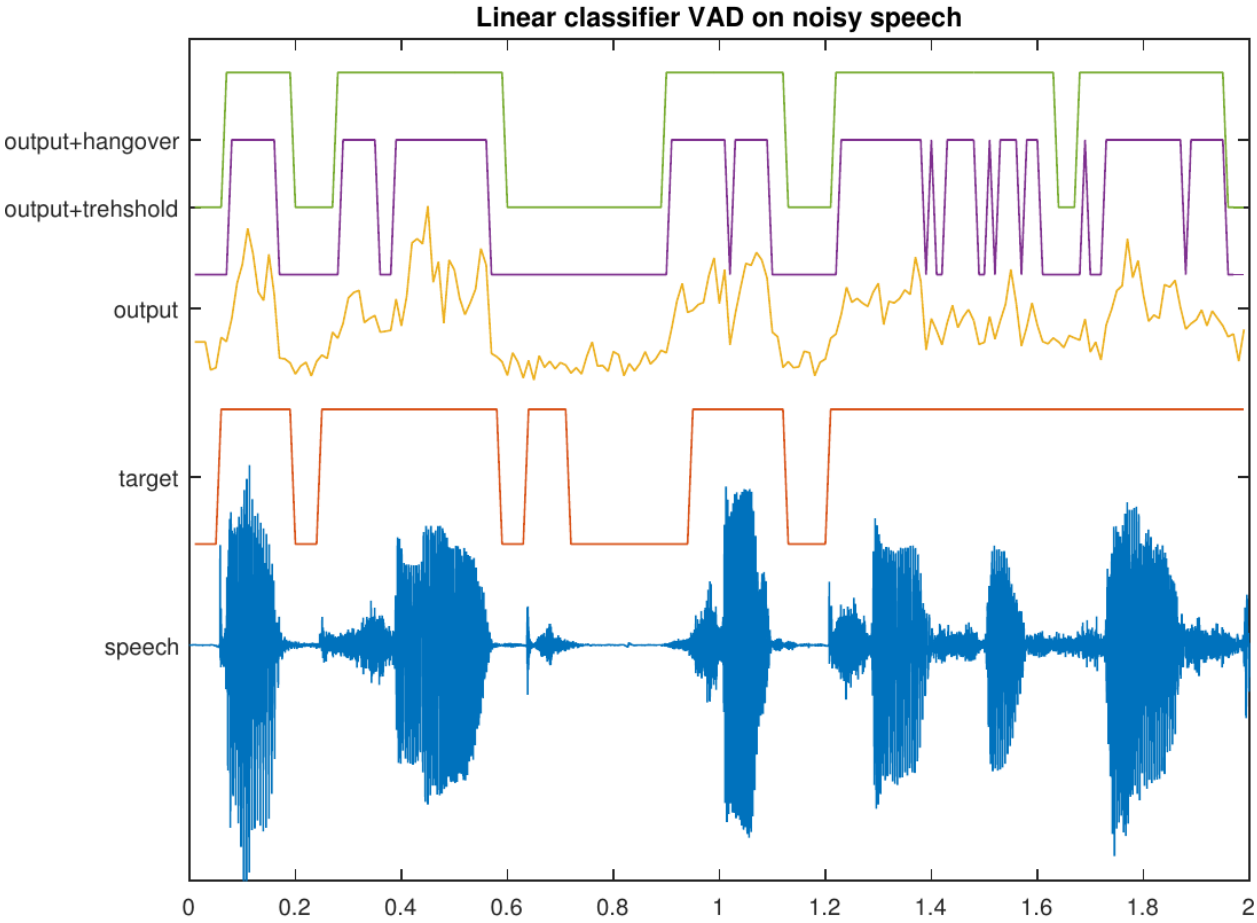
### Pre-whitening (advanced topic)

- If the range of values from features are very different, we end up with problems.
  - A very loud voice will overrun weaker ones, even if the loud one is full of crap.
  - The range (mean and variance) of features need to be normalized.
  - Correlations between features are also undesirable.
- The first step is removal of the mean, $x' = x - E[x]$, where $E[x] \approx \frac{1}{N}\sum_k x_k$ and $N$ is the number of frames.
- The covariance of the features is then $C = E[x'(x')^T] \approx \frac{1}{N}X^T X$, where $X$ now contains the zero-mean features.
- The eigenvalue decomposition of $C$ is $C = V^T DV$, whereby we can define the pre-whitening transform $A = D^{1/2}V$ and $x'' = Ax'$.
  - The covariance of the modified vector is $E[x''(x'')^T] = AE[x'(x')^T]A^T = ACA^T = D^{1/2}VV^T DVV^T D^{1/2} = I$.
  - That is, $x''$ has uncorrelated samples with equal variance and zero-mean.
- What you need to know is that pre-whitening is a pre-processing step, applied *before* training $w$.
- We thus train the classifier on the modified vectors $x'' = A(x - E[x])$, to obtain the weights $w$.
- The classifier with pre-whitening is \$\nu = w^T x'' = w^T A(x - E[x]) = \hat{w}^T(x - E[x])where\hat w=Aw\$.
- In other words, the pre-whitening can be included in the weights, so no additional complexity is introduced other than removal of the mean (which is trivial).

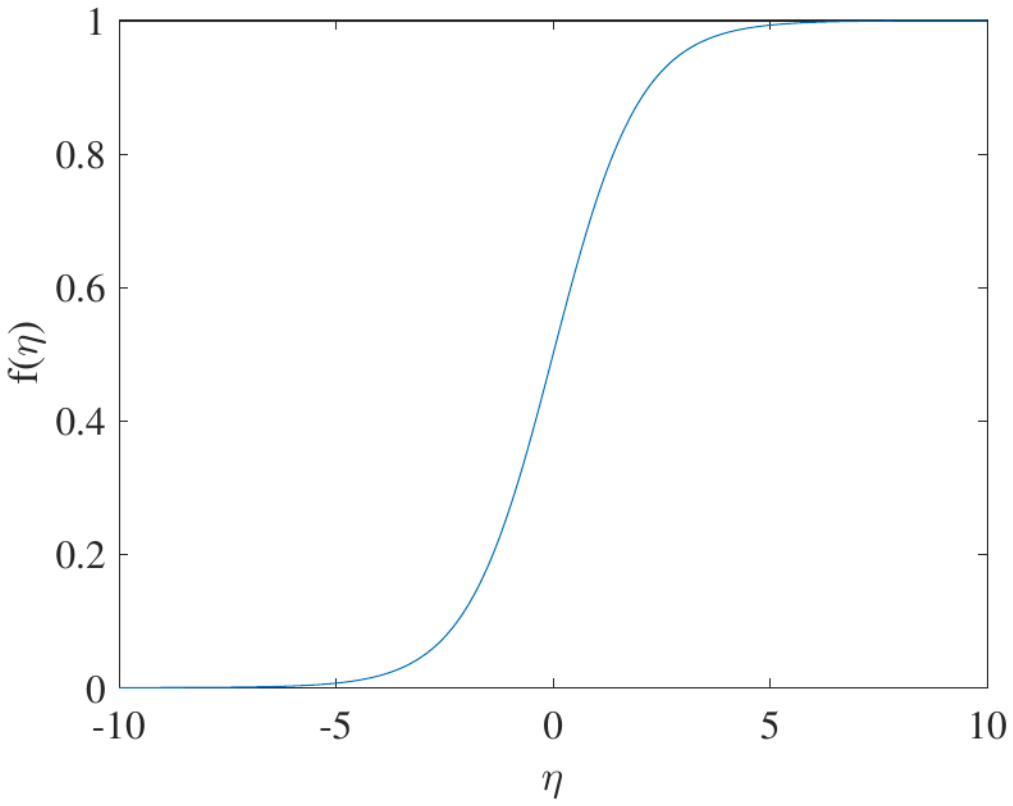

(Features - mean)/standard deviation
Target output
Speech
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Whitened features
Target output
Speech
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

### Post-processing




- Above, we trained the classifier to get values $\pm 1$, but when using the classifier, we choose classes based on a threshold.
- In practice, it does not matter if the output value is $0.1$, $1$ or $1000$, because if it is above threshold, then it belongs to the same class.
- Trying to hit $\pm 1$ is therefore an unnecessarily difficult task!
  - We should be just trying to obtain a multidimensional threshold which separates classes!
- We can truncate the output such that big values are reduced to $\pm 1$.
- The sigmoid function is a map $\mathbb{R} \to [0, 1]$ defined as $\$f(\nu) = \frac{1}{1+\exp(-\nu)}.\$$
- If we apply the sigmoid function on the output of the linear classifier, then overshooting is not a problem anymore: $\$\hat{\nu} = f(w^T x) \qquad \in [0, 1].\$$



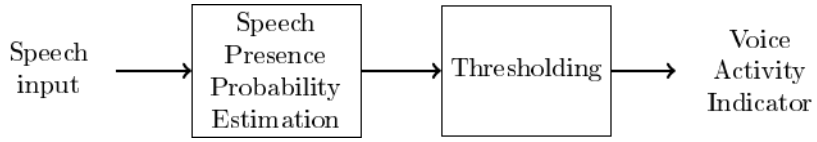

- Then there is no analytic solution, but we must use iterative methods.
- This function is known as a *perceptron*. Combining perceptrons into a interconnected network gives a *neural network*.⇒ A topic for other courses.
- Linear classifiers are only slightly more complex than decisions trees, but much more accurate.
  - Main complexity of VAD lies in feature-extraction anyway, so the differences in complexity of decision trees and linear classifiers is negligible.
- The main advantages of linear classifiers in comparison to decision trees are that
  - (unbiased) we can use real data to train the model, whereby we can be certain that it corresponds to reality (no bias due to manual tuning),
  - (robust) whereas noise in one feature can break a decision tree, linear classifiers merge information from all features, thus reducing effect of noise.

### More advanced classifiers

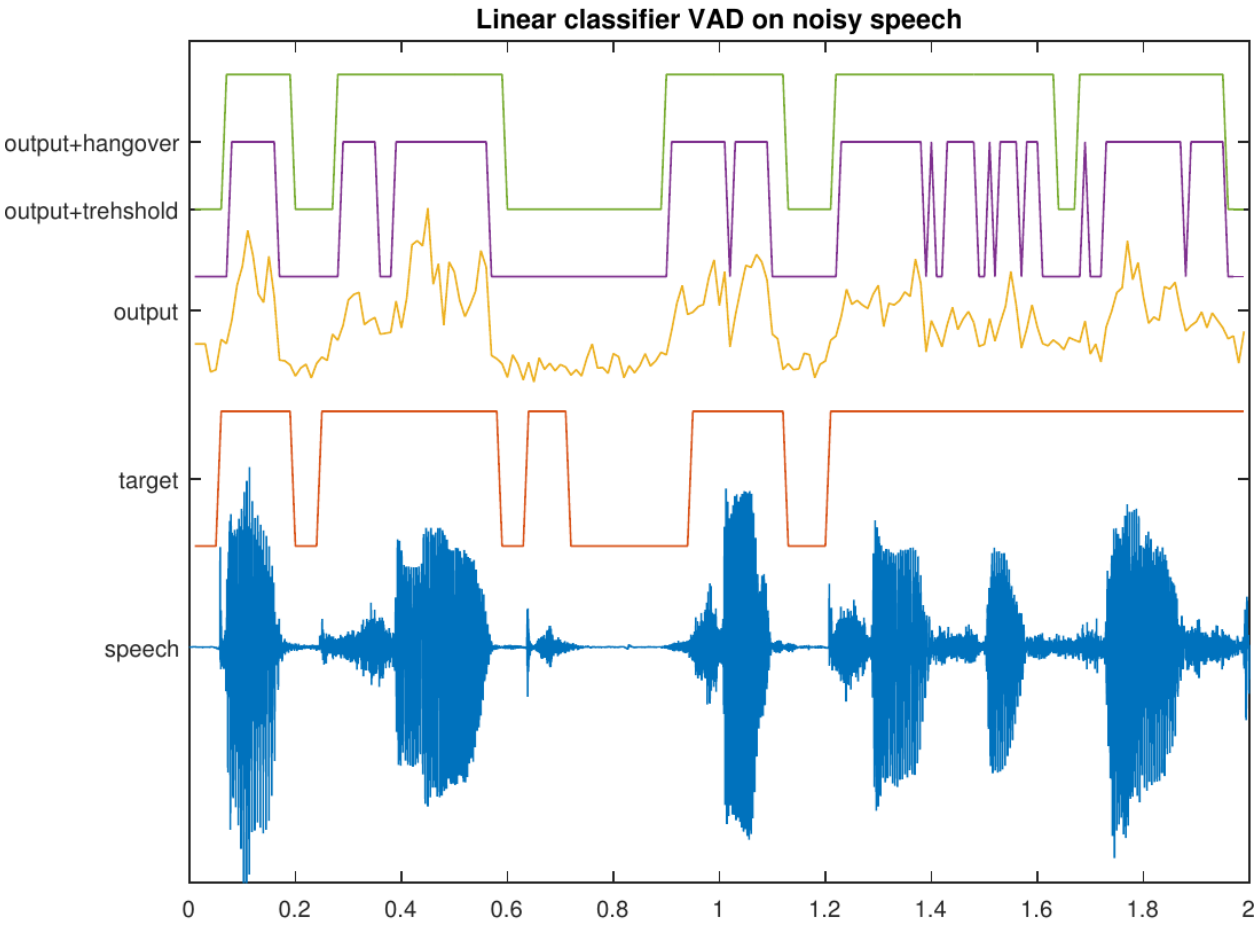
- There exists a large range of better and more complex classifiers in the general field of machine learning.
  - Linear discriminant analysis (LDA) – splits the feature space using hyper-planes.
  - Gaussian mixture models (GMM) – the feature space is modelled by a sum of Gaussians.
  - Deep neural networks (DNN) – similar to linear classifiers but adds non-linear mappings and several layers of sums.
  - K-nearest neighbors (kNN), support vector machine (SVM), random forest classifiers, etc.
- These methods are in general more effective, but training and application is more complex.
  - Advice: Try a simple approach first and see if its good enough.

### 8.1.7 Speech Presence Probability

- The output of the classifier is a continuous number, but it is thresholded to obtain a decision.
- The continuous output contains a lot of information about the signal which is lost when applying thresholding.
  - With a high value we are really certain that the signal is speech, while a value near the threshold is relatively uncertain.
- We can use the classifier output as an estimate of the probability that the signal is speech ⇒ It is an estimator for *speech presence probability*.
- Subsequent applications can use this information as input to improve performance.




#### Output before thresholding



### 8.1.8 Noise types

- As noted before, VAD is trivial in noise-free scenarios.
- In practice, typical background noise types are for example, office noise, car noise, cafeteria (babble) noise, street noise, factory noise, …
- Clearly the problem is easier if the noise has a very different character than the speech signal.
  - Speech is quickly varying ⇒ stationary noises are easy.
  - Speech is dominated by low frequencies ⇒ high frequency noises are easy.
- The classic worst case is a competing (undesired) speaker, when someone else is speaking in the background (babble noise).
  - However, that would be difficult also for a human listener, whereby it actually is a very difficult problem.

### 8.1.9 Conclusions

- Voice activity detection is a type of methods which attempt to determine if a signal is speech or non-speech.
  - In a noise-free scenario the task is trivial, but it is also not a realistic scenario.
- The basic idea of algorithms is:
  1. Calculate a set of features from the signal which are designed to analyze properties which differentiate speech and non-speech.
  2. Merge the information from the features in a classifier, which returns the likelihood that the signal is speech.
  3. Threshold the classifier output to determine whether the signal is speech or not.
- VADs are used as a low-complexity pre-processing method, to save resources (e.g. complexity or bitrate) in the main task.

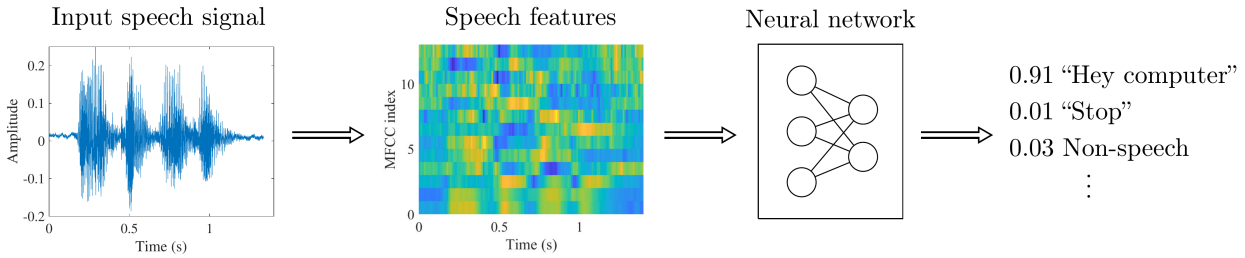
## 8.2 Wake-word and keyword spotting

Full large-vocabulary speech recognition is unnecessary and far too expensive for many practical applications. For a table lamp, it would quite sufficient if it understands “Light off” and “Light on.” Besides, most of the time, speech devices are just sitting there waiting for instructions when nobody is speaking. Speech recognition algorithms moreover use a lot of computational resources such that it would be wasteful to have them analyse sounds when there is no speech present. Voice activity detection (VAD) takes care of the first part, to detect whether speech is present or not, such that all activities are in a sleep mode when speech is not present.

Wake-word and keyword spotting refer to small-vocabulary speech recognition tasks. They are used either in very simple applications where proper speech recognition is unnecessarily complex (keyword spotting), and in pre-processing tasks, where we want to save resources by waiting for a “Hey computer!”. In the latter task, wake-word spotting is thus a trigger for more complex speech processing tasks. Though the two tasks have rather different objectives, the underlying technology is very similar and they will here be discussed jointly.

Most typically wake-word and keyword spotting algorithms run on devices with limited resources. They can be limited in memory footprint and in computation resources (CPU power) or often both. Increasing amount of memory or using a larger CPU would both increase cost of device (investment cost), but would also require more power (maintenance cost). In small devices such marginal costs are a very significant part of the overall cost of the device.

The overall structure of of keyword-spotting algorithms is illustrated below. The input speech signal is first converted to a feature representation, such as MFCCs, which are fed to a neural network, and the output is the likelihood of each keyword.




Adapted from [Zhang *et al.*, 2017]

In other words, keyword and wake-word spotters have a small set of accepted keywords, which are hard-coded into the software. If we have *N* possible keywords, then the neural network has *N* outputs corresponding to the probability that the input is each of those keywords. The output is then thresholded such that that keyword is chosen which has the largest probability.

The choice of structure for the neural network is highly dependent on the device constraints as well as the objective, operating environment and context of the application. Usually the structure is a deep neural network featuring some recurrent, convolutional and long-short term memory (LSTM) components.

A central challenge in training keyword spotting algorithms is finding and choosing training data. To get good quality, you would typically need several tens of thousands of utterances of the keywords, spoken by a large range of different speakers and in different environments. For example the "Speech Commands Dataset" by Google has 65.000 utterances of 30 short words. However, the choice of keywords is naturally dependent on those functions that the keyword spotter should activate, or the desired wake-word. For real-world applications we therefore often cannot use pre-collected datasets, but have to collect our own. You can just imagine the workload required to collect 65.000 utterances from over a thousand speakers!

### 8.2.1 References

## 8.3 Speech Recognition

### 8.3.1 Introduction to ASR

An ASR system produces the most likely word sequence given an incoming speech signal. The statistical approach for speech recognition has dominated Automatic Speech Recognition (ASR) research over the last few decades leading to a number of successes. The problem of speech recognition is defined as the conversion of spoken utterances into textual sentences by a machine. In the statistical framework, the Bayesian decision rule is employed to find the most probable word sequence, $\hat{H}$ , given the observation sequence $O = (o_1, ..., o_T)$ :

$$\hat{H} = \underset{H}{\text{argmax}} \; P(H|O)$$

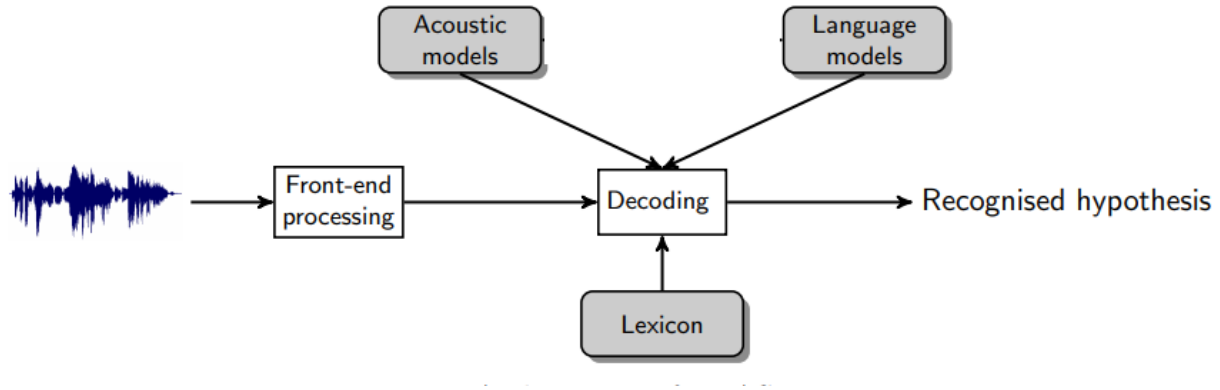
Following Bayes' rule, the posterior probability in the above equation can be expressed as a conditional probability of the word sequence given the acoustic observations, $P(O|H)$ , multiplied by a prior probability of the word sequence, $P(H)$ , and normalized by the marginal likelihood of observation sequences, $P(O)$ :

$$\hat{H} = \underset{H}{\text{argmax}} \; \frac{P(O|H) \; P(H)}{P(O)} \hat{H} = \text{argmax}_H \; P(O|H) \; P(H)$$

The marginal probability, $P(O)$ , is discarded in the second equation since it is constant with respect to the ranking of hypotheses, and hence does not alter the search for the best hypothesis. $P(O|H)$ is calculated by the acoustic model and $P(H)$ is modeled by the language model.

### 8.3.2 Component of ASR

- Feature Extraction: It converts the speech signal into a sequence of acoustic feature vectors. These observations should be compact and carry sufficient information for recognition in the later stage.
- Acoustic Model: It Contains a statistical representation of the distinct sounds that make up each word in the Language Model or Grammar. Each distinct sound corresponds to a phoneme.
- Language Model: It contain a very large list of words and their probability of occurrence in a given sequence.
- Decoder: It is a software program that takes the sounds spoken by a user and searches the acoustic Model for the equivalent sounds. When a match is made, the decoder determines the phoneme corresponding to the sound. It keeps track of the matching phonemes until it reaches a pause in the users speech. It then searches the language model for the equivalent series of phonemes. If a match is made, it returns the text of the corresponding word or phrase to the calling program.




Architecture of an ASR system

### 8.3.3 Types of ASR

Speech recognition systems can be classified on the basis of the constraints under which they are developed and which they consequently impose on their users. These constraints include: speaker dependence, type of utterance, size of the vocabulary, linguistic constraints, type of speech and environment of use. We will describe each constraint as follows:

**Speaker Dependence**: Speaker dependent speech recognition system requires the user to be involved in its development whereas speaker independent systems do not. Speaker independent systems can be used by anybody. Speaker dependent systems usually perform much better than speaker independent systems. This is due to the fact that the acoustic variations among different speakers are very difficult to describe and model. There are approaches to make a system speaker independent. The first one is the use of multiple representations for each reference to capture the speaker variation and the second one is the speaker adaptation approach.

**Type of Utterance**: A speech recognizer may recognize every word independently. It may require its user to speak each word in a sentence separating them by artificial pause or it may allow the user to speak in a natural way. The first type of system is categorized as an isolated word recognition system. It is the simplest form of a recognition strategy. It can be developed using word-based acoustic models without any language model. If, however, the vocabulary increases sentences composed of isolated words to be recognized, the use of sub-word acoustic models and language models become important. The second one is the continuous speech recognition systems. It allows the users to utter the message in a relatively or completely unconstrained manner. Such recognizers must be capable of performing well in the presence of all the co-articulatory effects. Developing continuous speech recognition systems is, therefore, the most difficult task. This is due to the following properties of continuous speech: word boundaries are unclear in continuous speech; and co-articulatory effects are much stronger in continuous speech

**Vocabulary Size**: The number of words in the vocabulary is a constraint that makes a speech recognition system small, medium or large. As a rule of thumb, small vocabulary systems are those which have a vocabulary size in the range of 1-99 words; medium, 100-999 words; and large, 1000 words or more. Large vocabulary speech recognition systems perform much worse compared to small vocabulary systems due to different factors such as word confusion that increases with the number of words in the vocabulary. For small vocabulary recognizer, each word can be modeled. However, it is not possible to train acoustic models for thousands of words separately because we cannot have enough training speech and storage for parameters of the speech that is needed. The development of large vocabulary recognizer, therefore, requires the use of sub-word units. On the other hand, the use of sub-word units results in performance degradation since they cannot capture co-articulatory effects as words do. The search process in large vocabulary recognizer also uses pruning instead of performing a complete search.

**Type of Speech:** A speech recognizer can be developed to recognize only read speech or to allow the user speak spontaneously. The latter is more difficult to build than the former due to the fact that spontaneous speech is characterized by false starts, incomplete sentences, unlimited vocabulary and reduced pronunciation quality. The primary difference in recognition error rates between read and spontaneous speech are due to disfluencies in spontaneous speech. Disfluencies in spontaneous speech can be characterized by long pauses and mispronunciations. Spontaneous is, therefore, both acoustically and grammatically difficult to recognize.

**Environment**: Speech recognizer may require the speech to be clean from environmental noises, acoustic distortions, microphones and transmission channel distortions or they may ideally handle any of these problems. While current speech recognizer give acceptable performance in carefully controlled environments, their performance degrades rapidly when they are applied in noisy environments. This noise can take the form of speech from other speakers; equipment sounds, air conditioners or others. The noise might also be created by the speaker himself in a form of lip smacks, coughs or sneezes.

### 8.3.4 Models for Large Vocabulary Speech Recognition (LVCSR)

LVCSR can be divided into two categories: HMM-based model and the end-to-end model.

#### HMM-Based Model

The HMM-based model has been the main LVCSR model for many years with the best recognition accuracy. An HMM-based model is divided into three parts:acoustic, pronunciation and language model. In HMM based model, each model is independent of each other and plays a different role. While the acoustic model models the mapping between speech input and feature sequence, the pronunciation model maps between phonemes (or sub-phonemes) to graphemes, and the language model maps the character sequence to fluent final transcription.

**Acoustic Model:** In the acoustic model, the observation probability is generally represented by GMM. The posterior probability distribution of hidden state can be calculated by DNN method. These two different calculations result into two different models, namely HMM-GMM and HMM-DNN. HMM-GMM model was a general structure for many speech recognition systems. However, with the development of deep learning technology, DNN is introduced into speech recognition for acoustic modeling. DNN has been used to calculate the posterior probability of the HMM state replacing the conventional GMM observation probability. Thus, HMM-GMM model is replaced by HMM-DNN since HMM-GMM provides better results compared to HMM-GMM and becomes state-of-the-art ASR model. In the HMM-based model, different modules use different technologies and have different roles. While the HMM is mainly used to do dynamic time warping at the frame level, GMM and DNN are used to calculate emission probability of HMM hidden states.

**Pronunciation Model:** Its main objective is achieve the connection between acoustic sequence and language sequence. The dictionary includes various levels of mapping, such as pronunciation to phone, phone to trip-hone. The dictionary is used to achieve structural mapping and map the probability calculation relationship.

**Language Model:** It contains rudimentary syntactic information. Its aim is to predict the likelihood of specific words occurring one after another in a given language. Typical recognizers use n-gram language models. An n-gram contains the prior probability of the occurrence of a word (unigram), or of a sequence of words (bigram, trigram etc.):

unigram probability $P(w_i)$

bigram probability $P(w_i|w_{i-1})$

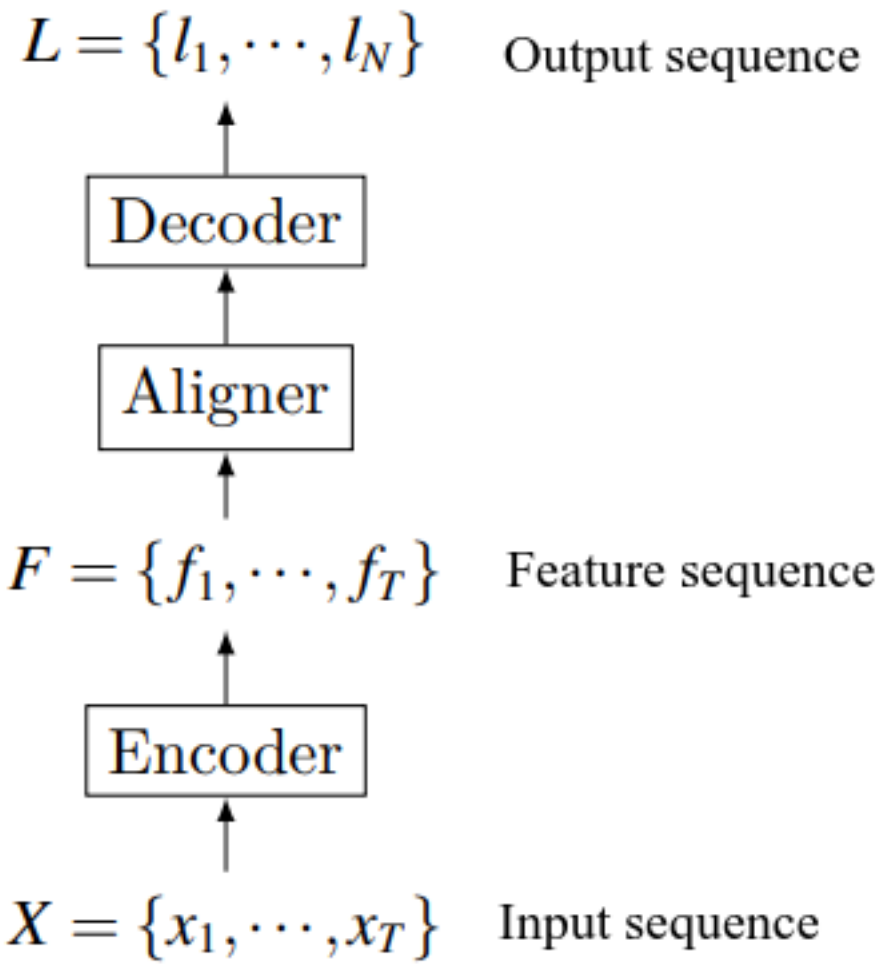
ngram probability $P(w_n|w_{n-1}, w_{n-2}, ..., w_1)$

**Limitations of HMM-models**

- The training process is complex and difficult to be globally optimized. HMM-based model often uses different training methods and data sets to train different modules. Each module is independently optimized with their own optimization objective functions which are generally different from the true LVCSR performance evaluation criteria. So the optimality of each module does not necessarily bring global optimality.
- Conditional independence assumptions. To simplify the model's construction and training, the HMM-based model uses conditional independence assumptions within HMM and between different modules. This does not match the actual situation of LVCSR.

### End-to-End Model

Because of the above-mentioned shortcomings of the HMM-based model and coupled with the promotion of deep learning technology, more and more works began to study end-to-end LVCSR. The end-to-end model is a system that directly maps input audio sequence to sequence of words or other graphemes.




Function

structure of end-to-end model

Most end-to-end speech recognition models include the following parts: the encoder maps speech input sequence to feature sequence; the aligner realizes the alignment between feature sequence and language; the decoder decodes the final identification result. Note that this division does not always exist since end-to-end itself is a complete structure. Contrary to the HMM-based model that consists of multiple modules, the end-to-end model replaces multiple modules with a deep network, realizing the direct mapping of acoustic signals into label sequences without carefully-designed intermediate states. In addition to this, there is no need to perform posterior processing on the output.

Compared to HMM-based model, the main characteristics of end-to-end LVCSR are:

- Multiple modules are merged into one network for joint training. The benefit of merging multiple modules is there is no need to design many modules to realize the mapping between various intermediate states. Joint training enables the end-to-end model to use a function that is highly relevant to the final evaluation criteria as a global optimization goal, thereby seeking globally optimal results.
- It directly maps input acoustic signature sequence to the text result sequence, and does not require further processing to achieve the true transcription or to improve recognition performance . But, in the HMM-based models, there is usually an internal representation for pronunciation of a character chain.

These features of of end-to-end LVCSR model enables to greatly simplify the construction and training of speech recognition models.

The end-to-end model are mainly divided into three different categories depending on their implementations of soft alignment:

- CTC-based: It first enumerates all possible hard alignments. Then, it achieves soft alignment by aggregating these hard alignments. CTC assumes that output labels are independent of each other when enumerating hard alignments.
- RNN-transducer: It also enumerates all possible hard alignments and then aggregates them for soft alignment. But unlike CTC, RNN-transducer does not make independent assumptions about labels when enumerating hard alignments. Thus, it is different from CTC in terms of path definition and probability calculation.
- Attention-based: This method no longer enumerates all possible hard alignments, but uses attention mechanism to directly calculate the soft alignment information between input data and output label.

CTC-Based End-to-End Model

Although HMM-DNN provides still state-of-the-art results, the role played by DNN is limited. It is mainly used to model the posterior state probability of HMM's hidden state. The time-domain feature is still modeled by HMM. When attempting to model time-domain features using RNN or CNN instead of HMM, it faces a data alignment problem: both RNN and CNN's loss functions are defined at each point in the sequence, so in order to be able to perform training, it is necessary to know the alignment relation between RNN output sequence and target sequence.

CTC makes it possible to make fuller use of DNN in speech recognition and build end-to-end models, which is a breakthrough in the development of end-to-end method. Essentially, CTC is a loss function, but it solves hard alignment problem while calculating the loss. CTC mainly overcomes the following two difficulties for end-to-end LVCSR models:

- Data alignment problem. CTC no longer needs to segment and align training data. This solves the alignment problem so that DNN can be used to model time-domain features, which greatly enhances DNN's role in LVCSR tasks.
- Directly output the target transcriptions. Traditional models often output phonemes or other small units, and further processing is required to obtain the final transcriptions. CTC eliminates the need for small units and direct output in final target form, greatly simplifying the construction and training of end-to-end model.

**RNN-Transducer End-to-End Model**

CTC has two main deficiencies in CTC which hinder its effectiveness:

- CTC cannot model interdependencies within the output sequence because it assumes that output elements are independent of each other. Therefore, CTC cannot learn the language model. The speech recognition network trained by CTC should be treated as only an acoustic model.
- CTC can only map input sequences to output sequences that are shorter than it. Thus, it is powerless for scenarios where output sequence is longer.

For speech recognition, the first point has huge impact. RNN-transducer was proposed to solve the above-mentioned shortcomings of CTC. Theoretically, it can map an input to any finite, discrete output sequence. Interdependencies between input and output and within output elements are also jointly modeled.

The RNN-transducer has many similarities with CT: their main goals is to solve the forced segmentation alignment problem in speech recognition; they both introduce a "blank" label; they both calculate the probability of all possible paths and aggregate them to get the label sequence. However, their path generation processes and the path probability calculation methods are completely different. This gives rise to the advantages of RNN-transducer over CTC.

### 8.3.5 Types of errors made by speech recognizers

Though ASR research has come a long way, today's systems are far from being perfect. Speech recognizer are brittle and make errors due to various causes. Most errors made by ASR systems fall into one of the following categories:

- **Out-of-vocabulary (OOV) errors**: Current state of the art speech recognizers have closed vocabularies. This means that they are incapable of recognizing words outside their training vocabulary. Besides misrecognition, the presence of an out-of-vocabulary word in input utterance causes the system to err to a similar word in its vocabulary. Special techniques for handling OOV words have been developed for HMM-GMM and neural ASR systems (see, e.g., [Zhang, 2019]).
- **Homophone substitution**: These errors can occur if more than one lexical entry has the same pronunciation (phone sequence), i.e., they are homophones. While decoding, homophones may be confused with one another causing errors. In general, a well-functioning language model should disambiguate homophones based on the context.
- **Language model bias**: Because of an undue bias towards the language model (effected by a high relative weight on the language model), the decoder may be forced to reject the true hypothesis in favor of a spurious candidate with high language model probability. These errors may occur along with analogous acoustic model bias.
- **Multiple acoustic problems**: This is a broad category of errors comprising those due to bad pronunciation entries; disfluency, mispronunciation by the speaker himself/herself, or errors made by acoustic models (possibly due to acoustic noise, data mismatch between training and usage etc.).

### 8.3.6 Challenges of ASR

Recent advances in ASR has brought automatic speech recognition accuracy close to human performance in many practical tasks. However, there are still challenges:

- Out-of-vocabulary words are difficult to recognize correctly
- Varying environmental noises impair recognition accuracy.
- Overlapping speech is problematic for ASR system.
- Recognizing children's speech and the speech of people with speech production disabilities is suboptimal with regular training data.
- DNN-based models usually require a lot of data for training, in the order of thousands of hours. End-to-end models may need up to 100,000h of speech to reach high performance.
- Uncertainty self-awareness is limited: typical ASR systems always output the most likely word sequence instead of reporting if some part of the input was incomprehensible or highly uncertain.

### 8.3.7 Evaluation

The performance of an ASR system is measured by comparing the hypothesized transcriptions and reference transcriptions. Word error rate (WER) is the most widely used metric. The two word sequences are first aligned using a dynamic programming-based string alignment algorithm. After the alignment, the number of deletions (D), substitutions (S), and insertions (I) are determined. The deletions, substitutions and insertions are all considered as errors, and the WER is calculated by the rate of the number of errors to the number of words (N) in the reference.

$$WER = \frac{I + D + S}{N} * 100\%$$

Sentence Error Rate (SER) is also sometime used to evaluate the performance of ASR systems. SER computes the percentage of sentences with at least one error.

#### References

## 8.4 Speaker Recognition and Verification

### 8.4.1 1. Introduction to Speaker Recognition

Speaker recognition is the task of identifying a speaker using their voice. Speaker recognition is classified into two parts: speaker identification and speaker verification. While speaker identification is the process of determining which voice in a group of known voices best matches the speaker', speaker verification is the task of accepting or rejecting the identity claim of a speaker by analyzing their acoustic samples. Speaker verification systems are computationally less complex than speaker identification systems since they require a comparison between only one or two models, whereas speaker identification requires comparison of one model to N speaker models.

Speaker verification methods are divided into text-dependent and text-independent methods. In text-dependent methods, the speaker verification system has prior knowledge about the text to be spoken and the user is expected to speak this text. However, in a text-independent system, the system has no prior knowledge about the text to be spoken and the user is not expected to be cooperative. Text-dependent systems achieve high speaker verification performance from relatively short utterances, while text-independent systems require long utterances to train reliable models and achieve good performance.

Block diagram of a basic speaker verification system

As it is shown in the above block diagram of a basic speaker verification system, a speaker verification system involves two main phases: the training phase in which the target speakers are enrolled and the testing phase in which a decision about the identity of the speaker is taken. From a training point of view, speaker models can be classified into generative and discriminative. Generative models such as Gaussian Mixture Model (GMM) estimate the feature distribution within each speaker. Discriminative models such as Support Vector Machine and Deep Neural Network (DNN), in contrast, model the boundary between speakers.

The performance of speaker verification systems is degraded by the variability in channels and sessions between enrolment and verification speech signals. Factors which affect channel/session variability include:

1. Channel mismatch between enrolment and verification speech signals such as using different microphones in enrolment and verification speech signals.
2. Environmental noise and reverberation conditions.
3. The differences in speaker voice such as ageing, health, speaking style and emotional state.
4. Transmission channel such as landline, mobile phone, microphone and voice over Internet protocol (VoIP).

### 8.4.2 2. Front-end Processing

Many front-end processing are often used to process the speech signals and to extract the features which are used in the speaker verification system. The front-end processing consists of mainly voice activity detection (VAD), feature extraction and channel compensation techniques;

1. *Voice activity detection (VAD)*; The main goal of voice activity detection is to determine which segments of a signal are speech and non-speech. A robust VAD algorithm can improve the performance of a speaker verification system by making sure that speaker identity is calculated only from speech regions. Therefore, it is necessary to review the VAD algorithm to overcome the problems in designing a robust speaker verification system. The three widely used techniques for VAD are the following: energy based, model based and hybrid approaches.
2. Feature extraction techniques are used to transform the speech signals into acoustic feature vectors. Thus, the extracted acoustic features should carry the essential characteristics of the speech signal which recognizes the identity of the speaker by their voice. The aim of feature extraction is to reduce the dimension of acoustic feature vectors by removing unwanted information and emphasizing the speaker-specific information. The MFCCs are commonly used as the feature extraction technique for the modern speaker verification.
3. Channel compensation techniques are used to reduce the effect of channel mismatch and environmental noise. Channel compensation can be used in different stages of speaker verification such as feature and model domains. Various channel compensation techniques such as cepstral mean subtraction (CMS) , feature warping, cepstral mean variance normalization (CMVN) and relative spectral (RASTA) processing have been used to reduce the effect of channel mismatch during the feature extraction phase. In the model domain, Joint Factor Analysis (JFA) and i-vectors are used to combat enrolment and verification mismatch.

### 8.4.3 3. Speaker Modeling Techniques

One of the crucial issues in speaker diarization is the techniques employed for speaker modeling. Several modeling techniques have been used in speaker recognition and speaker diarization tasks. The state-of-the-art speaker modeling techniques in speaker diarization are the following:

### 8.4.4 3.1. Gaussian Mixture Modeling (GMM) - Universal Background Model (UBM) Approach

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs have been successfully used to model the speech features in different speech processing applications. A Gaussian mixture model is a weighted sum of M component Gaussian densities. Each of the components is a multi-variant Gaussian function. A GMM is represented by mean vectors, covariance matrices and mixture weights.

\[ \lambda = \{w_i, \mu_i, \Sigma_i\}, \quad i= 1,……..,\textit{C} \]

The covariance matrices of a GMM, \( \Sigma_i \) , can be full rank or constrained to be diagonal. The parameters of a GMM can also be shared, or tied, among the Gaussian components. The number of GMM components and type of covariance matrices are often determined based on the amount of data available for estimating GMM parameters.

In speaker recognition, a speaker can be modeled by a GMM from training data or using Maximum A Posteriori (MAP) adaptation. While the speaker model is built using the training utterances of a specific speaker in the GMM training, the model is also usually adapted from a large number of speakers called Universal Background Model in MAP adaptation.

Given a set of training vectors and a GMM configuration, there are several techniques available for estimating the parameters of a GMM . The most popular and used method is the maximum likelihood (ML) estimation.

The ML estimation finds the model parameters that maximize the likelihood of the GMM given a set of data. Assuming an independence between the training vectors \( X = \{x_i,\dots,x_N\} \) , the GMM likelihood is typically described as:

\[ p(X|\lambda) = \prod_{t=1}^{N} p(x_t|\lambda) (1) \]

Since direct maximization is not possible on equation on equation 1, the ML parameters are obtained iteratively using expectation-maximization (EM) algorithm. The EM iteratively estimate new model parameters \( \bar{\lambda} \) based on a given model
$lambda$ such that \( p(X|\bar{\lambda}) \ge p(X|\lambda) \) .

Example of speaker model adaptation

The parameters of a GMM can also be estimated using Maximum A Posteriori (MAP) estimation, in addition to the EM algorithm. The MAP estimation technique derives a speaker model by adapting from a universal background model (UBM). The "Expectation" step of EM and MAP are the same. MAP adapts the new sufficient statistics by combining them with old statistics from the prior mixture parameters.

Given a prior model and training vectors from the desired class, \( X = {x_1 … , x_T } \) , we first determine the probabilistic alignment of the training vectors into the prior mixture components. For mixture \( i \) in the prior model \( Pr(i|x_t,\lambda_{UBM}) \) is computed as the percentage of the mixture component \( i \) to the total likelihood,

\[ Pr(i|x_t,\lambda_{UBM})= \frac {w_i\,g(x_t|\mu_i,\Sigma_i)} {\sum_{i=1}^{M} w_i\,g(x_t|\mu_i,\Sigma_i)} \]

Then, the sufficient statistics for the weight, mean and variance parameters is computed as follows:

\[ n_i=\sum_{t=1}^{T}Pr(i|x_t,\lambda_{prior}) \, weight \] \[ E_i(x)=\frac{1}{n_i}\sum_{t=1}^{T}Pr(i|x_t,\lambda_{prior})x_t \;\; mean \] \[ E_i(x^2)=\frac{1}{n_i}\sum_{t=1}^{T}Pr(i|x_t,\lambda_{prior})x_t^2 \;\; variance \]

Finally, the new sufficient statistics from the training data are used to update the prior sufficient statistics for mixture \( i \) to create the adapted mixture weight, mean and variance for mixture \textit{i} as follows:

\[ w_i=[\alpha^w_in_i/T + (1-\alpha^w_i)w_i]\gamma \] \[ \mu_i=\alpha^m_iE_i(x) + (1-\alpha^m_i)\mu_i \] \[ \mu^2_i=\alpha^v_iE_i(x^2) + (1-\alpha^v_i)(\sigma^2_i+\mu^2_i)-\mu^2_i \]

The adaptation coefficients controlling the balance between old and new estimates are \( \{\alpha^w_i, \alpha^m_i, \alpha^v_i\} \) for the weights, means and variances, respectively. The scale factor, \( \gamma \) , is computed over all adapted mixture weights to ensure they sum to unity.

### 8.4.5 3.2. i-Vectors

Different approaches have been developed recently to improve the performance of speaker recognition systems. The most popular ones were based on GMM-UBM. The Joint Factor Analysis (JFA) is then built on the success of the GMM-UBM approach. JFA modeling defines two distinct spaces: the speaker space defined by the eigenvoice matrix and the channel space represented by the eigen-channel matrix. The channel factors estimated using JFA, which are supposed to model only channel effects, also contain information about speakers. A new speaker verification system has been proposed using factor analysis as a feature extractor that defines only a single space, instead of two separate spaces. In this new space, a given speech recording is represented by a new vector, called total factors as it contains the speaker and channel variabilities simultaneously. Speaker recognition based on the i-vector framework is currently the state-of-the-art in the field.

Given an utterance, the speaker and channel dependent GMM supervector is defined as follows:

\[ M=m+Tw \]

where \( m \) is a speaker and channel independent supervector, \( T \) is a rectangular matrix of low rank and \( w \) is a random vector having a standard normal distribution \( N(0,1) \) . The components of the vector \( w \) are the total factors. These new vectors are called i-vectors. \( M \) is assumed to be normally distributed with mean vector and covariance matrix \( TT^t \) .

The total factor is a hidden variable, which can be defined by its posterior distribution conditioned to the Baum–Welch statistics for a given utterance. This posterior distribution is a Gaussian distribution and the mean of this distribution corresponds exactly to i-vector. The Baum–Welch statistics are extracted using the UBM.

Given a sequence of L frames \( \{y_1,y_2,……,y_n\} \) and a UBM \( \Omega \) composed of \( C \) mixture components defined in some feature space of dimension \( F \) , the Baum–Welch statistics needed to estimate the i-vector for a given speech utterance \( u \) is given by :

\[ N_c=\sum_{t=1}^{L}P(cly_t,\Omega) \] \[ F_c=\sum_{t=1}^{L}P(cly_t,\Omega)y_t \]

where \( m_c \) is the mean of UBM mixture component \( c \) . The i-vector for a given utterance can be obtained using the following equation:

\[ w=(I + T^t \Sigma^{-1} N(u)T)^{-1}. \,T^t\Sigma^{-1}\hat{F}(u) \]

where \( N_u \) is a diagonal matrix of dimension \( CF \times CF \) whose diagonal blocks are \( N_cI(c=1,……, C) \) . The supervector obtained by concatenating all first-order Baum–Welch statistics \( F_c \) for a given utterance \( u \) is represented by \( \hat{F}(u) \) which has \( CF \times 1 \) dimension. The diagonal covariance matrix, \( \Sigma \) , with dimension \( CF \times CF \) estimated during factor analysis training models the residual variability not captured by the total variability matrix \( T \) .

Process of i-Vector extraction

One of the most widely used feature normalization techniques of i-vectors is length normalization. Length normalization ensures that the distribution of i-vectors matches the Gaussian normal distribution and makes the distributions of i-vector more similar. Performing whitening before length normalization improves the performance of speaker verification systems. i-vector normalization improves the gaussianity of the i-vectors and reduces the gap between the underlying assumptions of the data and real distributions. It also reduces the dataset shift between development and test i-vectors.

\[ w\leftarrow \frac{\Sigma^{-\frac{1}{2}}(w-\mu)}{||\Sigma^{-\frac{1}{2}}(w-\mu)||} \]

where \( \mu \) and \( \Sigma \) are the mean and the covariance matrix of a training corpus, respectively. The data is standardized according to covariance matrix \( \Sigma \) and length-normalized (i.e., the i-vectors are confined to the hypersphere of unit radius.

The two most widely and common intersession compensation techniques of i-vectors are Within-Class Covariance Normalization (WCCN) and Linear Discriminant Analysis (LDA). WCCN uses the within-class covariance matrix to normalize the cosine kernel functions in order to compensate for intersession variability. LDA attempts to define a reduced special axes that minimize the within-speaker variability caused by channel effects, and maximize between-speaker variability.

**Cosine Distance**

Once the i-vectors are extracted from the outputs of speech clusters, cosine distance scoring tests the hypothesis if two i-vectors belong to the same speaker or different speakers. Given two i-vectors, the cosine distance among them is calculated as follows:

\[ cos(w_i,w_j)=\frac{w_i.w_j}{||w_i||.||w_j||}\gtreqless \theta \]

where \( \theta \) is the threshold value, and \( cos(w_i,w_j) \) is the cosine distance score between clusters \( i \) and \( j \) . The corresponding i-vectors extracted for clusters \( i \) and \( j \) are represented by \( w_i \) and \( w_j \) , respectively.

The cosine distance scoring considers only the angle between two i-vectors, not their magnitude. Since the non-speaker information such as session and channel variabilities affect the i-vector magnitude, removing the magnitudes can increase the robustness of i-vector systems.

**Probabilistic Linear Discriminant Analysis**

The i-vector representation followed by probabilistic linear discriminant analysis (PLDA) modeling technique is the state-of-the-art in speaker verification systems. PLDA has been successfully applied in speaker recognition experiments. It is also applied to handle speaker and session variability in speaker verification task. It has also been successfully applied in speaker clustering since it can separate speaker and noise specific parts of an audio signal which is essential for speaker diarization.

Example of PLDA model

In PLDA, assuming that the training data consists of \textit{J} i-vectors where each of these i-vectors belong to speaker \textit{I}, the \textit{j'th} i-vector of the \textit{I'th} speaker is denoted by:

\[ w_{ij}=\mu + Fh_i + Gy_{ij} + \Sigma_{ij} \]

where \( \mu \) is the overall speaker and segment independent mean of the i-vectors in the training dataset, columns of the matrix F define the between-speaker variability and columns of the matrix G define the basis for the within-speaker variability subspace. \( \Sigma_{ij} \) represents any unexplained data variation. The components of the vector \( h_i \) are the eigenvoice factor loadings and components of the vector \( y_{ij} \) are the eigen-channel factor loadings. The term \( Fh_i \) depends only on the identity of the speaker, not on the particular segment.

Although the PLDA model assumes Gaussian behavior, there is empirical evidence that channel and speaker effects result in i-vectors that are non-Gaussian. It is reported in that the use of Student's t-distribution, on the assumed Gaussian PLDA model, improves the performance. Since this normalization technique is complicated, a non-linear transformation of i-vectors called radial Gaussianization has been proposed. It whitens the i-vectors and performs length normalization. This restores the Gaussian assumptions of the PLDA model.

A variant of PLDA model called Gaussian PLDA (GPLDA) is shown to provide better results. Because of its low computational requirements, and its performance, it is the most widely used PLDA modeling. In GPLDA model, the within-speaker variability is modeled by a full covariance residual term which allows us to omit the channel subspace. The generative PLDA model for the i-vector is represented by

\[ w_{ij}=\mu + Fh_i + \Sigma_{ij} \]

The residual term \( \Sigma \) representing the within-speaker variability is assumed to have a normal distribution with full covariance matrix \( \Sigma \) .

Given two i-vectors \( w_1 \) and \( w_1 \) , the PLDA computes the likelihood ratio of the two i-vectors as follows:

\[ Score(w_1,w_2)= \frac{p(w_1,w_2|H_1)}{p(w_1|H_2) p(w_2|H_2)} \]

where the hypothesis \( H_1 \) indicates that both i-vectors belong to the same speaker and \( H_0 \) indicates they belong to two different speakers.

### 8.4.6 3.3. Deep Learning (DL)

The recent advances in computing hardware, new DL architectures and training methods, and access to large amount of training data has inspired the research community to make use of DL technology again as in speaker recognition systems. DL techniques can be used in the frontend or/and backend of a speaker recognition system. The whole end-to-end recognition process can even be performed by a DL architecture.

**Deep Learning Frontends: **The traditional i-vector approach consists of mainly three stages: Baum-Welch statistics collection, i-vector extraction, and PLDA backend. Recently, it is shown that if the Baum-Welch statistics are computed with respect to a DNN rather than a GMM or if bottleneck features are used in addition to conventional spectral features, a substantial improvement can be achieved. Another possible use of DL in the frontend is to represent the speaker characteristics of a speech signal with a single low dimensional vector using a DL architecture, rather than the traditional i-vector algorithm. These vectors are often referred to as speaker embeddings. Typically, the inputs of the neural network are a sequence of feature vectors and the outputs are speaker classes.

**Deep Learning Backends: **One of the most effective backend techniques for i-vectors is PLDA which performs the scoring along with the session variability compensation. Usually, a large number of different speakers with several speech samples each are necessary for PLDA to work efficiently. Access to the speaker labeled data is costly and in some cases almost impossible. Moreover, the amount of the performance gain, in terms of accuracy, for short utterances is not as much as that for long utterances. These facts motivated the research community to look for DL based alternative backends. Several techniques have been proposed. Most of these approaches use the speaker labels of the background data for training, as in PLDA, and mostly with no significant gain compared to PLDA.

**Deep Learning End-to-Ends: **It is also interesting to train an end-to-end recognition system capable of doing multiple stages of signal processing with a unified DL architecture. The neural network will be responsible for the whole process from the feature extraction to the final similarity scores. However, working directly on the audio signals in the time domain is still computationally too expensive and, therefore, the current end-to-end DL systems take mainly the handcrafted feature vectors, e.g., MFCCs, as inputs. Recently, there have been several attempts to build an end-to-end speaker recognition system using DL though most of them focus on text-dependent speaker recognition.

### 8.4.7 4. Applications of Speaker Recognition

- Transaction authentication – Toll fraud prevention, telephone credit card purchases, telephone brokerage (e.g., stock trading)
- Access control – Physical facilities, computers and data networks
- Monitoring – Remote time and attendance logging, home parole verification, prison telephone usage
- Information retrieval – Customer information for call centers, audio indexing (speech skimming device), speaker diarization
- Forensics – Voice sample matching

### 8.4.8 5. Performance Evaluations

The performance of the speaker verification is measured in terms of errors. The types of error and evaluation metrics commonly used in speaker verification systems are the following.

### 8.4.9 5.1 Types of errors

False acceptance: A false acceptance occurs when the speech segments from an imposter speaker are falsely accepted as a target speaker by the system.

\[ False\; Acceptance = \frac{Total \; number \;of\; false\; acceptance\; errors} {Total\; number\; of\; imposter\; speaker\; attempts} \]

****

**False rejection:** A false rejection occurs when the target speaker is rejected by the verification systems.

\[ False\;rejection = \frac{Total \;number\; of\; false\; rejection\; errors} {Total\; number\; of\; enrolled\; speaker\; attempts} \]

### 8.4.10 5.2 Performance metrics

The performance metrics of speaker verification systems can be measured using the equal error rate (EER) and minimum decision cost function (mDCF). These measures represent different performance characteristics of system though the accuracy of the measurements is based on the number of trials evaluated in order to robustly compute the relevant statistics. Speaker verification performance can also be represented graphically by using the detection error trade-off (DET) plot. The EER is obtained when the false acceptance rate and false rejection rate have the same value. The performance of the system improves if the value of ERR is lower because the sum of total error of the false acceptance and false rejection at the point of ERR decreases. The decision cost function (DCF) is defined by assigning a cost of each error and taking into account the prior probability of target and impostor trails. The decision cost function is defined as:

\[ DCF = C_{miss}P_{miss}P_{target} + C_{fa}P_{fa}P_{impostor} \]

where \( C_{miss} \) and \( C_{fa} \) are the cost functions of a missed detection and false alarm, respectively. The prior probabilities of target and impostor trails are given by \( P_{target} \) and \( P_{impostor} \) , respectively. The percentages of the missed target and falsely accepted impostors' trails are represented by \( P_{miss} \) and \( P_{fa} \) , respectively. The mDCF is used to evaluate speaker verification by selecting the minimum value of DCF estimated by changing the threshold value. The mDCF can be used to evaluate speaker verification by selecting the minimum value of DCF estimated by changing the threshold value.

\[ mDCF = min[C_{miss}P_{miss}P_{target} + C_{fa}P_{fa}P_{impostor}] \]

where \( P_{miss} \) and \( P_{fa} \) are the miss and false alarm rates recorded from the trials, and the other parameters are adjusted to suit the evaluation of application-specific requirements.

### 8.4.11 Attachments:

## 8.5 Speaker Diarization

### 8.5.1 1. Introduction to Speaker Diarization

Speaker diarization is the process of segmenting and clustering a speech recording into homogeneous regions and answers the question "who spoke when" without any prior knowledge about the speakers. A typical diarization system performs three basic tasks. Firstly, it discriminates speech segments from the non-speech ones. Secondly, it detects speaker change points to segment the audio data. Finally, it groups these segmented regions into speaker homogeneous clusters.

An overview of a speaker diarization system.

Although there are many different approaches to perform speaker diarization, most of them follow the following scheme:

Feature extraction: It extracts specific information from the audio signal and allows subsequent speaker modeling and classification. The extracted features should ideally maximize inter-speaker variability and minimize intra-speaker variability, and represent the relevant information.

Speaker segmentation: It partitions the audio data into acoustically homogeneous segments according to speaker identities. It detects all boundary locations within each speech region that corresponds to speaker change points which are subsequently used for speaker clustering.

Speaker clustering: Speaker clustering groups speech segments that belong to a particular speaker. It has two major categories based on its processing requirements. Its two main categories are online and offline speaker clustering. In the former, speech segments are merged or split in consecutive iterations until the optimum number of speakers is acquired. Since the entire speech file is available before decision making in the later, it provides better results more than the online speaker clustering. The most widely used and popular technique for speaker clustering is Agglomerative Hierarchical Clustering (AHC). AHC builds a hierarchy of clusters, that shows the relations between speech segments, and merges speech segments based on similarity. AHC approaches can be classified into bottom-up and top-down clustering.

Two items need to be defined in both bottom-up and top-down clustering:

1. A distance between speech segments to determine acoustic similarity. The distance metric is used to decide whether or not two clusters must be merged (bottom-up clustering) or split (top-down clustering).

2. A stopping criterion to determine when the optimal number of clusters (speakers) is reached.

Bottom-up (Agglomerative): It starts from a large number of speech segments and merges the closest speech segments iteratively until a stopping criterion is met. This technique is the most widely used in speaker diarization since it is directly applied on the output of speech segments from speaker segmentation. A matrix of distances between every possible pair of clusters is computed and the pair with highest BIC value is merged. Then, the merged clusters are removed from the distance matrix. Finally, the distance matrix table is updated using the distances between the new merged cluster and all remaining clusters. This process is done iteratively until the stopping criterion is met or all pairs have a BIC value less than zero .

Top-down: Top-down Hierarchical Clustering methods start from a small number of clusters, usually a single cluster that contains several speech segments, and the initial clusters are split iteratively until a stopping criterion is met. It is not as widely used as the bottom-up clustering.

Bottom-Up and Top-down approaches to clustering

### 8.5.2 2. Approaches to Speaker Diarization

This section describes some of the state-of-the-art speaker diarization systems.

HMM/GMM based speaker diarization system: Each speaker is represented by a state of an HMM and the state emission probabilities are modeled using GMMs. The initial clustering is performed initially by partitioning the audio signal equally which generates a set of segments { \( s_i \) }. Let \( c_i \) represent \( i^{th} \) speaker cluster, \( b_i \) represent the emission probability of cluster \( c_i \) and \( f_t \) denote a given feature vector at time \( t \) . Then, the log-likelihood \( logb_i(s_t) \) of the feature ft for cluster \( c_i \) is calculated as follows:

\[ logb_i(s_t)=log \sum_{(r)} {w}^r_i N (f_i,{\mu}^r_i,\sum_{(i)}^r) \]

where \( N() \) is a Gaussian pdf and \( {w}^r_i, {\mu}^r_i,\sum_{(i)}^r) \) are the weights, means and covariance matrices of the \( r^{th} \) Gaussian mixture component of cluster \( c_i \) , respectively.

The agglomerative hierarchical clustering starts by overestimating the number of clusters. At each iteration, the clusters that are most similar are merged based on the BIC distance. The distance measure is based on modified delta Bayesian information criterion [Ajmera and Wooters, 2003]. The modified BIC distance does not take into account the penalty term that corresponds to the number of free parameters of a multivariate Gaussian distribution and is expressed as:

\[ \Delta BIC (c_i,c_j)= log \sum_{f_t \in ( {ci \; \cup \; c_j})} log b_{ij}(f_t) - log \sum_{f_t \in ci} log b_{i}(f_t) - log \sum_{f_t \in cj} log b_{j}(f_t) \]

where \( b_{ij} \) is the probability distribution of the combined clusters \( c_i \) and \( c_j. \)

The clusters that produce the highest BIC score are merged at each iteration. A minimum duration of speech segments is normally constrained for each class to prevent decoding short segments. The number of clusters is reduced at each iteration. When the maximum \( \Delta BIC \)

distance among these clusters is less than threshold value 0, the speaker diarization system stops and outputs the hypothesis.

Factor analysis techniques: Factor analysis techniques which are the state of the art in speaker recognition have recently been successfully used in speaker diarization. The speech clusters are first represented by i-vectors and the successive clustering stages are performed based on i-vector modeling. The use of factor analysis technique to model speech segments reduces the dimension of the feature vector by retaining most of the relevant information. Once the speech clusters are represented by i-vectors, cosine-distance and PLDA scoring techniques can be applied to decide if two clusters belong to the same or different speaker(s).

Deep learning approaches: Speaker diarization is crucial for many speech technologies in the presence of multiple speakers, but most of the current methods that employ i-vector clustering for short segments of speech are potentially too cumbersome and costly for the front-end role. Thus, it has been proposed by Daniel Povey an alternative approach for learning representations via deep neural networks to remove the i-vector extraction process from the pipeline entirely. The proposed architecture simultaneously learns a fixed-dimensional embedding for acoustic segments of variable length and a scoring function for measuring the likelihood that the segments originated from the same or different speakers. The proposed neural based system matches or exceeds the performance of state-of-the-art baselines.

### 8.5.3 3. Evaluation Metrics

Diarization Error Rate (DER) is the metric used to measure the performance of speaker diarization systems. It is measured as the fraction of time that is not attributed correctly to a speaker or non-speech.

The DER is composed of the following three errors:

Speaker Error: It is the percentage of scored time that a speaker ID is assigned to the wrong speaker. Speaker error is mainly a diarization system error (i.e., it is not related to speech/non-speech detection.) It also does not take into account the overlap speeches not detected.

False Alarm: It is the percentage of scored time that a hypothesized speaker is labelled as a non-speech in the reference. The false alarm error occurs mainly due to the the speech/non-speech detection error (i.e., the speech/non-speech detection considers a non-speech segment as a speech segment). Hence, false alarm error is not related to segmentation and clustering errors.

Missed Speech: It is the percentage of scored time that a hypothesized non-speech segment corresponds to a reference speaker segment. The missed speech occurs mainly due to the the speech/non-speech detection error (i.e., the speech segment is considered as a non-speech segment). Hence, missed speech is not related to segmentation and clustering errors.

\[ DER = Speaker \; Error + False \; Alarm + Miss \; Speech \]

## 8.6 Paralinguistic speech processing

Paralinguistic speech processing (PSP) refers to analysis of speech signals with the aim of extracting information beyond the linguistic content of speech (hence paralinguistic = alongside linguistic content; see also Schuller and Batliner [2013]). In other words, PSP does not focus on what is the literal transmitted message but on what additional information is conveyed by the signal. Speaker *diarization*, *recognition*, and *verification*, even though focusing on non-linguistic aspects, are also traditionally considered as separate problems that do not fall within the scope of PSP. A classical example of PSP is speech emotion recognition, where the aim is to infer the emotional state of a speaker based on a sample of his or her speech. In a similar manner, information related to the health or age of a speaker could be inferred from the speech signal.

### 8.6.1 Coupling between speaker states and the speech signal

The basic starting point for PSP systems is that the speech signal also reflects the underlying cognitive and neurophysiological state of a speaker. This is since speaking involves highly complicated cognitive processing in terms of real-time communicative, linguistic, and articulatory planning. In addition, execution of these plans requires highly-precise motor control of articulators paired with real-time monitoring of the resulting acoustic signal, and both of these tasks take place in parallel with further speech planning. The mental state of the speaker may also affect the manner that the speaker wishes to express himself or herself. Finally, the overall physiological characteristics of the speech production apparatus also shape the resulting signal, and details of these characteristics may also change due to illnesses or habits. This means that many temporary or permanent perturbations in the cognitive and physical machinery of a talker may show up in the resulting speech.

To give some examples, substantial cognitive load (e.g., a concurrent attention-requiring task) or neurodegenerative diseases affecting memory (e.g., Alzheimer's disease) may impact speech planning due to compromised cognitive resources, resulting in speech output that differs from the typical speech from the same person in non-stressful or healthy conditons. In the same way, neurodegenerative diseases affecting the brain's motor system (e.g., Parkinson's disease) may impact fluidity and clarity of speech production, and the symptoms will become more pronounced as the disease progresses. As for articulatory changes, stress and emotional distress can cause increased tension in the muscles of the larynx, which can result in tightening of the vocal folds and therefore also causes increases in the fundamental frequency of speech. Changes in the physical characteristics of the vocal tract may result from, e.g., having a cold. In this case, mucus on tract surfaces

may affect resonance and damping characteristics of the vocal tract. In addition, the mucus may prevent full closing of the velum, causing nasalized speech often associated with a severe cold. The speaker may also speak differently due to cognitive fatigue and throat soreness due to the cold. Aging will also change the characteristics of the speech production apparatus, not only in childhood but also in later years of life. These changes are driven by physiological changes in the glottis and in the vocal tract, where growth of the vocal tract length in early childhood has an especially pronounced effect. The voice change in puberty is also an example of quick growth of the larynx and vocal folds, but small changes in the vocal folds and their control may also take with later aging.

In addition to information that is not directly related to intended communicative goals, speech also contains paralinguistic characteristics related to communication. This is because speech has co-evolved with the development of other social skills in humans over thousands of years. Speech (and gestures) can thefore play different types of social coordinative roles beyond the literal linguistic message transmitted. For instance, prosody, and speaking style in general, can reflect different social roles such as submissiveness, arrogance, or authority in different interactions. Attitudes and emotions showing up in speech can also be considered as communicative signals facilitating social interaction and cohesion, not just being speaker-internal states that inadvertently "leak out" for others to perceive. Demonstration of anger or happiness through voice can transmit important information regarding social dynamics even when visual contact between the interlocutors is not possible. As a concrete example, consider having a telephone conversation with someone close to you without access to anything else than the literal message (e.g., substituting the original speech with monotone but perfectly intelligible speech synthesis) while trying to communicate highly sensitive and important personal information.

The basic aim of PSP is to use computational means to understand and characterize the ways that different paralinguistic factors shape the speech signal, and to build automatic systems for analyzing and detecting the paralinguistic factors from real speech captured in various settings.

Also note that the distinction between PSP systems and other established areas of speech processing is not always clear-cut. For instance, automated methods for speech intelligibility assessment are also focusing on extralinguistic factors, and the task of speaker recognition was already mentioned at the beginning of this section. In addition, more flexible control of speaking style is an ongoing topic of research in speech synthesis, where the research focus is gradually changing from the production of high-quality to speech to creation of systems capable of richer vocalic expression. However, there is no need for a strict distinction of PSP from other types of processing tasks, but PSP can be viewed as an umbrella term for the increasingly many analysis tasks focused on the various non-literal aspects of spoken language, and where similar data-driven methodology is usually applicable across a broad range of PSP phenomena (see also Schuller and Batliner [2013], for a discussion).

### 8.6.2 Speaker traits and states

Schuller and Batliner [2013] use a distinction into two types of speaker characteristics: traits and states. These are related to the temporal properties of the analyzed phenomena. Speaker traits are long-term and slowly-changing characteristics of the speaker, such as personality traits (e.g., Big Five classification), gender, age, or dialect. On the other hand, speaker states are short- to medium-term phenomena, such as speaker's emotional state, attitude in a conversation, (temporary) health conditions, fatigue, or stress level. When collecting data for PSP research and system development, it is important to consider the time-scale of the phenomenon to be analyzed and how this relates to practical needs of the analysis task (e.g., how much speech can be collected and analyzed before classification decision; does the system have to be real-time). For instance, quickly changing characteristics such as emotional state should be analyzed from relatively short speech recordings where the factor of interest can be assumed to be stable. For example, recognition of speaker's emotional state during a single utterance is a widely adopted approach in speech emotion recognition. In contrast, analysis of speaker health (e.g., COVID-19 symptoms) from just one utterance is likely to be inaccurate, but speaker-dependent data collection and analysis across longer stretches of speech will likely produce more reliable analysis outcomes. Moreover, longitudinal monitoring of a subject across longer periods of time is likely to be more accurate in detecting changes in the speaker's voice, as the system can be adapted to the acoustic and linguistic characteristics of that specific speaker. For instance, subject-specific monitoring of the progression of a neurodegenerative disease based on speech is likely to be more accurate than automatic classification of disease severity from a bag of utterances from a random collection of speakers.

### 8.6.3 Typical applications of PSP

Some possible applications of paralinguistic tasks include, but are not limited to:

- Emotion classification
- Personality classification (e.g., Big Five traits)
- Sleepiness or intoxication detection
- Analysis of cognitive or physical load
- Health-related analyses (cold, snoring, neurodegenerative diseases etc.)
- Speech addressee analysis (e.g., adult- vs. infant-directed speech)
- Age and gender recognition
- Sincerity analysis
- Attitude analysis

### 8.6.4 Basic problem formulation and standard solutions

The basic goal of paralinguistic analysis is to extract paralinguistic information of interest while ignoring the signal variability introduced by other factors, such as linguistic content, speaker identity, background noise or transmission channel characteristics (aka. *nuisance factors*). However, for some tasks, it may also be useful to analyse the lexical and grammatical content of speech in order to infer information regarding the phenomena of interest.

Typical PSP systems follow the two classical tasks of machine learning: classification and regression. In classification, the goal is to build a system that can assign a speech sample (e.g., an utterance) into one of two or more categories (e.g., intoxicated or not intoxicated). In regression, the target is a continuous (or at least ordinal) measure (e.g., blood alcohol concentration percentage).

Fig. 1 illustrates a standard PSP system pipeline, which follows a typical supervised machine learning scenario. First, a number of features are extracted from the speech signal. These features, together with the corresponding class labels, are then used to train a classifier model for the training dataset. During testing and actual use of the system, the classifier is used to determine the most likely class of an input waveform. Depending on the classifier architecture, the classification result may or may not be associated with a confidence measure of the classification decision. In regression tasks, the process is otherwise the same, but the categorical class labels are replaced by continous-valued measures and the classifier is replaced by a regression model.

A central characteristic of many PSP tasks is that the analyzed phenomenon (e.g., speaker emotion) is assumed to be fixed at a certain time-scale, such as across one utterance, and therefore classification decisions should also be made at time-scales longer than typical frame-level signal features. In this case, it would desirable to obtain a fixed-dimensional feature representation of the signal even if the duration of the input waveform varies from case to case. This can be achieved by a two-step process: **1)** first extracting regular frame-level features (e.g., spectral features such as FFT) with high temporal resolution (e.g., one frame every 10-ms), sometimes referred to as low-level descriptors (LLDs), and then **2)** calculating statistical parameters ("functionals") of each of the features across all the time-frames (illustrated in Fig. 2). Typical functionals of LLDs include, e.g., *min, max, mean, variance, skewness,* and *kurtosis*, but they can also be measures, such as centroids, percentiles, or different types of means. In order to acquire a more complete picture of the signal dynamics, first- and second-order time-derivatives of the frame-level features ("deltas" and "delta-deltas") are often included in the feature set before calculating the functionals. In neural networks, a varying-length signal can also be represented by a fixed-dimensional embedding extracted from the input waveform, such as taking the output of an LSTM-layer. Once the input signals are represented by fixed-dimensional feature vectors, standard machine learning classifiers can be applied to the data.

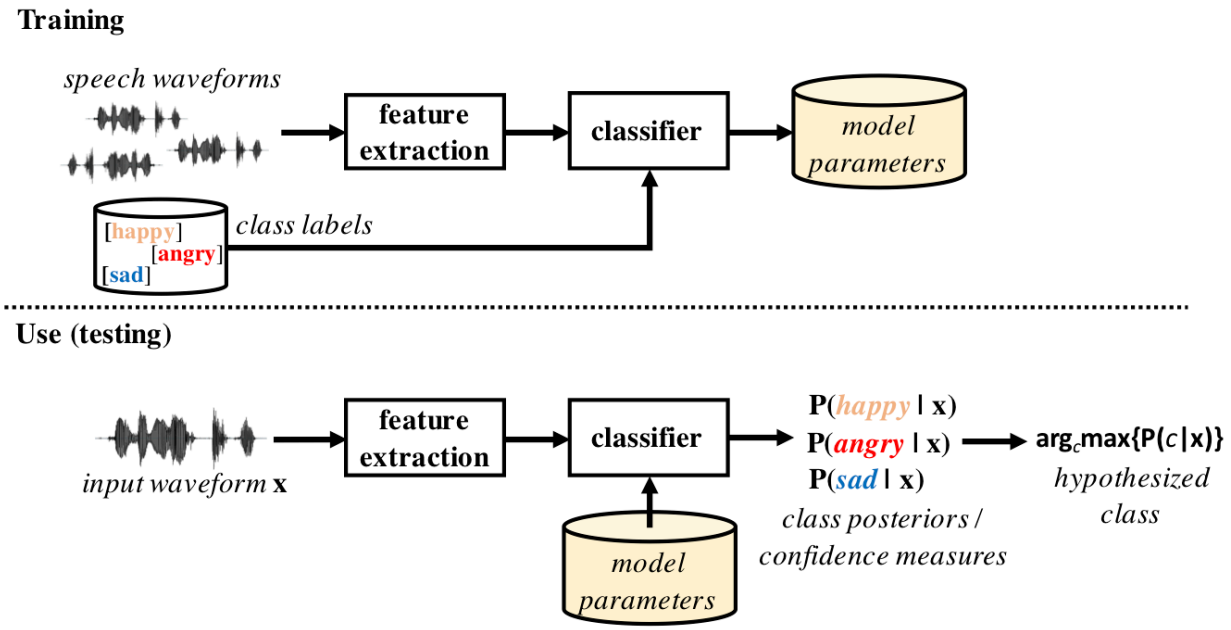



**Figure 1:** An example of a classical PSP processing pipeline with training (top) and usage (bottom). Speech features are first extracted from the original speech waveform, followed by a classifier that has been trained using supervised learning with labeled training samples.

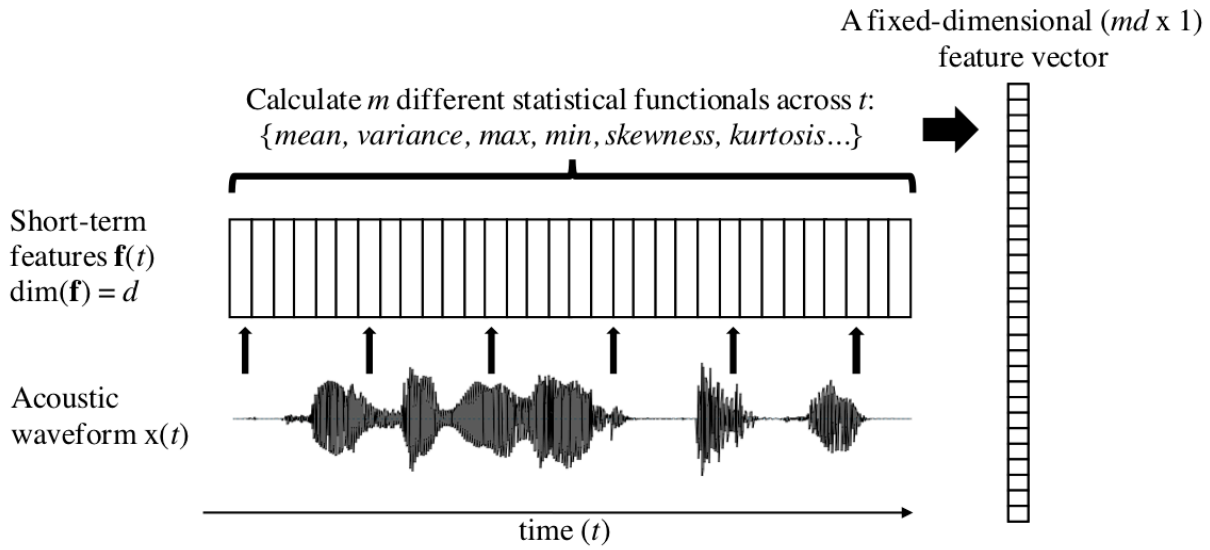



**Figure 2:** An example signal-level feature extraction process where variable-duration utterances become represented by fixed-dimensional feature vectors, consisting of statistical parameters ("functionals") calculated across frame-by-frame speech features (sometimes also referred to as low-level descriptors, or "LLDs").

In the both cases of classification and regression, the challenge is to build a system that can discriminate the key features of the phenomenon without being affected by the other sources of signal variability. High-performance well-generalizing systems can be approached with three basic strategies: *well-designed features*, *robust classifiers*, or *end-to-end learning*, where signal representations ("features") and classifiers are jointly learned from the data. Selection of the approach depends on two primary factors: domain knowledge and availability of representative large-scale training data (Fig. 3).

The basic principle is that well-designed or otherwise properly chosen features require less training data for inference of the machine learning model parameters, but this necessitates that the acoustic/linguistic characteristics of the ana-

lyzed phenomenon are known in order to design features that capture them. If there are only limited data and limited domain knowledge, it is still possible to calculate a large number of potentially relevant features and then use a classifier such as Support Vector Machines (SVMs; Boser et al., 1992) that can robustly handle high-dimensional features, including potentially task-irrelevant or otherwise noisy features. For instance, the baseline systems for annual Computational Paralinguistic Challenges (see also below) have traditionally used a combination of more than 6000 signal-level features together with an SVM classifier, and often these systems have been highly competitive with other solutions. Alternatively, feature selection techniques may be applied in conjunction with data labeling to find a more compact feature set for the problem at hand (see, e.g., Pohjalainen *et al.* [2015]).

If there are plenty of training data available, multilayer neural networks can be used to simultaneously learn useful signal representations and a classifier for the given problem. This type of representation learning can operate directly on the acoustic waveform or using some standard spectral representation with limited additional assumptions, such as FFT or log-Mel spectrum. The obvious advantage is that feature design is no longer needed, and the features and the classifier are seamlessly integrated and jointly optimized. In addition, even if domain knowledge would be available, the assumptions built into manually tailored features may lose some details of the modeled phenomenon, whereas end-to-end neural networks can potentially use all the information available in the input signal. Similarly to many other applications of machine learning, neural networks can therefore be expected to outperform the "more classical" approaches if sufficient training data are available for the task (see also sub-section below for data in PSP). However, besides the data requirement drawback, the standard problems and principles of neural network design and training apply, including the lack of access to the global optimum during the parameter optimization process.

If sufficient training data and domain knowledge are both available, one may also combine representation learning with good initial features or some type of model priors or constraints. This may speed-up the learning process or improve model convergence to a more effective solution due to a more favorable starting point for the optimization process. One particularly interesting new research direction is the use of differentiable computational graphs in feature extraction. In this case, prior task-related knowledge could be incorporated into digital signal processing (DSP) steps used to extract some initial features, but the feature extraction algorithm would be implemented together with the rest of the network as one differentiable computational graph (e.g., using a framework such as TensorFlow). This would allow error backpropagation through the entire pipeline from classification decisions to the feature extractor, thereby enabling task-optimized adaptation of the feature extraction or pre-processing steps. As an example, Discrete Fourier Transform is a differentiable function, allowing error gradients to pass through it, although by default it does not have any free parameters to optimize. However, similar signal transformations but with parametrized basis functions could be utilized and optimized jointly with the rest of the model.

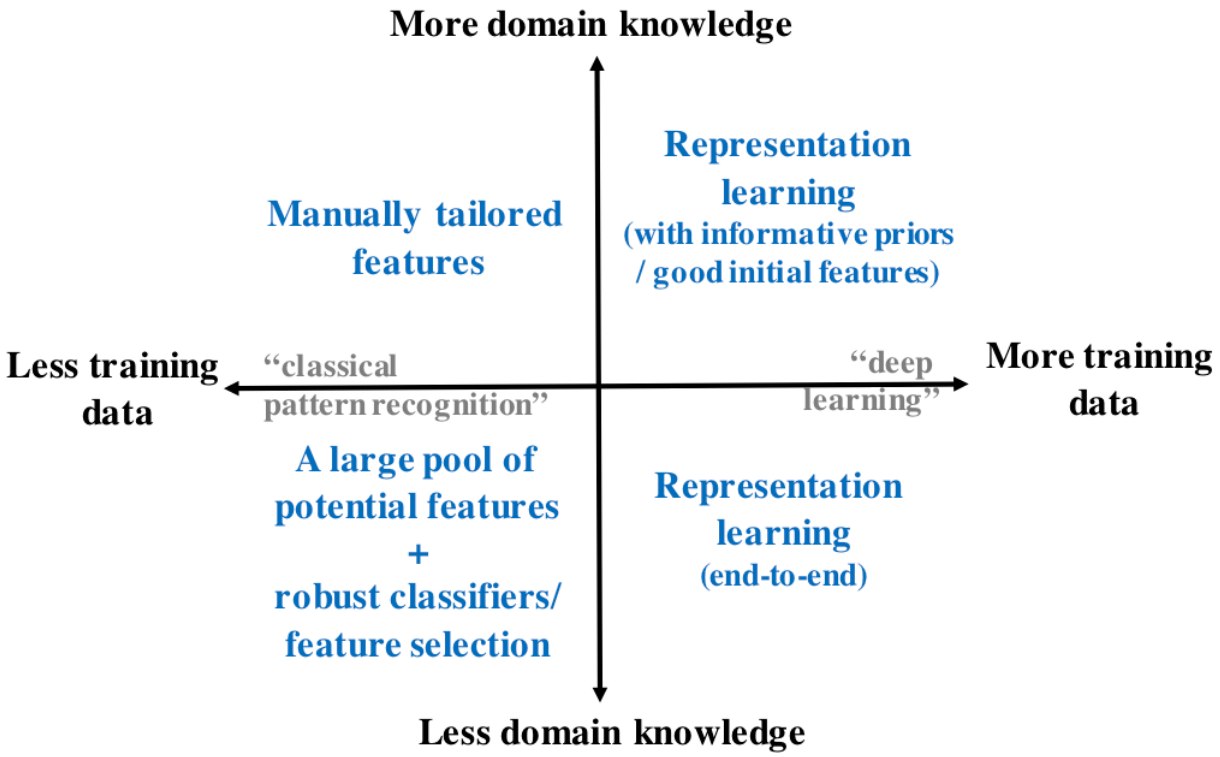



**Figure 3:** Basic strategies for PSP system development given the two main considerations: availability of training data (x-axis) and domain knowledge (y-axis).

Naturally, the division into the four categories described in Fig. 3 is by no means definite. For instance, recent developments in self-supervised representation learning (van den Oord et al., 2018; Chung et al., 2019; Baevski et al., 2020) or multitask learning (e.g., Wu et al., 2015) are still largely unexplored in PSP. Such approaches could enable effective utilization of large amounts of unlabeled speech data with fewer labeled examples in the target domain of interest.

### 8.6.5 Data collection and data sparsity

Since modern PSP largely relies on machine learning, representative training data will be required for the phenomenon of interest. However, access to high-quality labeled data is often be limited for many PSP tasks. First of all, the data collection itself is often challenging and may include important ethical considerations, such as collecting data from intoxicated speakers or subjects with rare diseases, or data where some type of objective (physiological) measurements of emotional state are captured simultaneously with the speech audio. Availability of reliable ground-truth labels for the speech data can also be difficult. For instance, there is no direct way to measure the underlying emotional states of speakers, whereas induced emotional speech by professional actors may not properly reflect the variability of real-world emotional expression. In a similar manner, assessment of severity of many diseases is based on various indirect measurements and clinical diagnostic practices, not on some type of oracle knowledge on a universally standardized scale. Every time humans are used for data labelling (e.g., assessing emotions), there is a certain degree of inter-annotator inconsistency due to differing opinions and general variability in human performance. This is the case even when domain experts are used for the task. Naturally, the more difficult the task or more ambiguous the phenomenon, the more there will be noise and ambiguity in the labeling.

Another limitation hindering PSP progress is that many PSP datasets cannot be freely distributed to the research community due to data ownership and human participant privacy protection considerations. As a clear example, speech with metadata related to factors such as health or IQ of the speakers is highly sensitive in nature, and not all speakers consent to open distribution of their identifiable voice together with such private data of themselves. The data ownership considerations inherently limit the pooling of different speech corpora in order to build more comprehensive databases of speech

related to different phenomena of interest, and generally slow down replicable open science. On the other hand, it is of utmost importance to respect the privacy of human participants in PSP (or any other) research—not only due to ethical considerations, but also since the entire field depends on access to data from voluntary human participants. Commercial interests to data ownership are also often unavoidable.

In total, this means that data is often a limiting factor in system performance, and therefore deep neural networks have still not become the only off-the-shelf-solution for many PSP problems. Efforts for more flexible (but ethical) data sharing and pooling is therefore an important challenge for future research. More powerful technical solutions and deeper understanding of the PSP phenomena could be acquired by combining rich data (and metadata) from various sources, including different languages, cultural environments, and recording conditions.

### 8.6.6 Computational Paralinguistic Challenge

Research in PSP has been strongly advanced by an annual Computational Paralinguistic Challenge (ComParE) held in the context of ISCA Interspeech conferences (see http://www.compare.openaudio.eu/). Every year since 2009, ComParE has included a number of paralinguistic analysis tasks with pre-defined datasets in which participants can compete with each other. Competitive baseline systems, evaluation protocols and results are always provided to the participants as a starting point, enabling low-barrier access to the world of PSP for researchers with various backgrounds. In addition, new tasks and datasets can be proposed to challenge organizers, providing a useful channel for data owners and researchers to obtain competitive solutions to their analysis problems.

### 8.6.7 Further reading and materials on PSP

Schuller, B. et al.: Computational Paralinguistic Challenge. WWW-site: http://www.compare.openaudio.eu/, last accessed 11th October 2020.

CHAPTER

NINE

# SPEECH SYNTHESIS

Speech synthesis systems aim at producing intelligible speech signals from some type of input language representation. Synthesis systems are also often referred to as text-to-speech (TTS) systems, as written text is a natural way to instruct what type of utterances should be produced.

A typical TTS system consists of two basic processing modules: text analysis module and a speech synthesis module.

**Text analysis**

The first module of a TTS pipeline, the text analysis module, is responsible for converting the incoming text into a linguistic representation that encodes the information of how the input text should be spoken.

As the first step, the module must process the text into a standardized format by taking care of any special characters or other inconsistencies. Then the text must be structurally analyzed to identify syntactical components of the sentences. Any inconsistencies between the written and spoken language (e.g., abbreviations and acronyms, proper names, numbers) must be detected and converted into a correct format (e.g., to map the string "100 sq ft" into "*one hundred square feet*", not "*one-zero-zero sq ft*") .

The second step consists of inferring the correct pronunciation of the words, also known as grapheme-to-phoneme conversion. Since the relationship between written and spoken language is not always straightforward, the text analysis module must convert strings of input letters into strings of phonemes based on the pronunciation conventions of the given language. This involves resolving many ambiguities prevalent in languages, such as how to deal with homographs, that is, words with the same written form but different pronunciations. Foreign language words and loan words have to be handled as well.

As the final stage, suprasegmental characteristics of the speech-to-be-produced must be introduced to the linguistic representation. Durations of the phonemes and intermediate pauses must be defined, even though the information does not exist in the text. In order to make the speech natural sounding and intelligible, the process also includes introduction of any potential rhythmic structure, stress patterns, and intonational cues to the linguistic representation. For this purpose, the text analysis module must interpret syntactic properties of the input text. For instance, the system must be able to differentiate different sentence types such as questions from statements and to infer which word should receive focus in the given sentential context.

**Synthesis algorithms**

The second key module consists of the speech synthesizer module. Input to the synthesizer is the linguistic representation produced by the text analysis block while the output consists of acoustic waveforms of synthesized speech. There are several potential technical approaches to the creation of a speech waveform from linguistic instructions. Historically, methods such as formant synthesis or articulatory synthesis have been utilized (where the latter is still used in speech research). However, modern commercial speech synthesizers are based on one of the two alternative techniques: *concatenative synthesis* or *statistical parametric speech synthesis*. Both methods are described in more detail in their respective sub-sections.

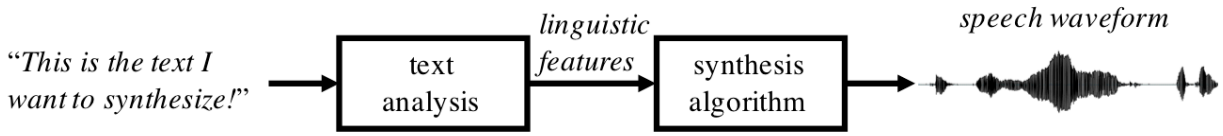



**Figure 1:** The basic structure of a speech synthesis system.**

In general, synthesis algorithms aim at speech output that maximally resembles natural speech, is free of noise and artifacts, and has high intelligibility. Other characteristics may include possibility to use different speaker voices or to speaking styles to account for different use contexts and user preferences. In practical use, computational complexity of the system may also become a relevant design factor. This is especially true if the system must support real-time speech production and/or serve multiple users simultaneously. For instance, speech synthesis used on a standard mobile device or in a car entertainment system must operate with strict latency computational complexity constraints.

Speech quality and intelligibility a speech synthesizer are typically evaluated using subjective listening tests.

**Further readings & material**

Simon King - Using Speech Synthesis to give Everyone their own Voice, University of Edinburgh. Youtube video (link). *Includes an overview of unit selection and statistical parametric synthesis with sound demonstrations.*

Kim Silverman - Speech synthesis lecture at ICSI, Berkeley. Youtube video (link).

Sai Krishna Rallabandi's on-line list of introductory resources for speech synthesis (link)

## 9.1 Concatenative speech synthesis

*Concantenative speech synthesis* (CSS), also known as *unit selection speech synthesis*, is one of the two primary modern speech synthesis techniques together with *statistical parametric speech synthesis*. As the name suggests, CSS is based on concatenation of pre-recorded speech segments in order to create intelligible high-quality speech. The advantage of this approach is extremely high naturalness of the produced speech, as long as the system is well-designed and suitable speech data are available for its development. The drawback is limited flexibility as all the used speech segments have to be pre-recorded, limiting the choice of speaker voices or other modifications to the verbal expression.

The most simple CSS system imaginable could be developed using concatenation of pre-recorded word waveforms. However, as [Rabiner and Schafer, 2007] note, such an approach would suffer from two primary problems. First, concatenation of word-level waveforms would sound unnatural, as coarticulatory effects between words would be absent from the data. In addition, the system would be limited to very restrictive scenarios only, as there can be tens or hundreds of thousands of lexical items and millions of proper names in any language—way more than what can be reasonably pre-recorded by any individual speaker. The problem is even worse for agglutinative languages such as Finnish, where word meanings are constructed and adjusted by extending word root forms with various suffixes. Since words can also participate to utterances in various positions and roles, prosodic characteristics (e.g., F0) of the same word can differ from context to another. This means that pre-recording all possible words is not a practical option.

To solve the issues of scalability and coarticulation, practical modern CSS systems are based on sub-word units. In principle, there are only few phones per language (e.g., around 40–50 for English), but their acoustic characteristics are also highly dependent on the surrounding context due to coarticulation. Context-dependent phones such as diphones (pairs of phones) or triphones (phone triplets) are therefore utilized. In order to build a CSS system, speech dataset has to be first carefully annotated and segmented for the units of interest. These segments can then be stored as acoustic parameters (e.g., speech codec parameters) to save space and to allow easy characterization and manipulation.

### 9.1.1 Steps of concatenative synthesis

Once a database of units exists, synthesis with CSS consists of the following basic steps: **1)** *conversion of input text to a target specification*, which includes the string of phones to be synthesized together with additional prosodic specifications such as pitch, duration, and power, **2)** *unit selection* for each phone segment according to the specification, and **3)** *post-processing* to reduce the impact of potential concatenation artefacts.

While the text processing stage is largely similar to pre-processing in *statistical parametric speech synthesis systems*, the main part of CSS is to perform unit selection in such a manner that the output speech matches the specification with high naturalness of the sound. As described by [Hunt and Black, 1996], unit selection is achieved by cost minimization using two cost functions (Fig. 1): *target cost* $C^t(u_i, t_i)$ and *concatenation cost* $C^c(u_{i-1}, u_i)$. Target cost describes the mismatch between the target speech unit specification $t_i$ and a candidate unit $u_i$ from the database. Concatenation cost describes the mismatch (e.g., acoustic or perceptual) of the join between the candidate unit $u_i$ and the preceding unit $u_{i-1}$. In other words, an ideal solution would find all the target units according to the specification without introducing acoustic mismatches at the edges of concatenated units.

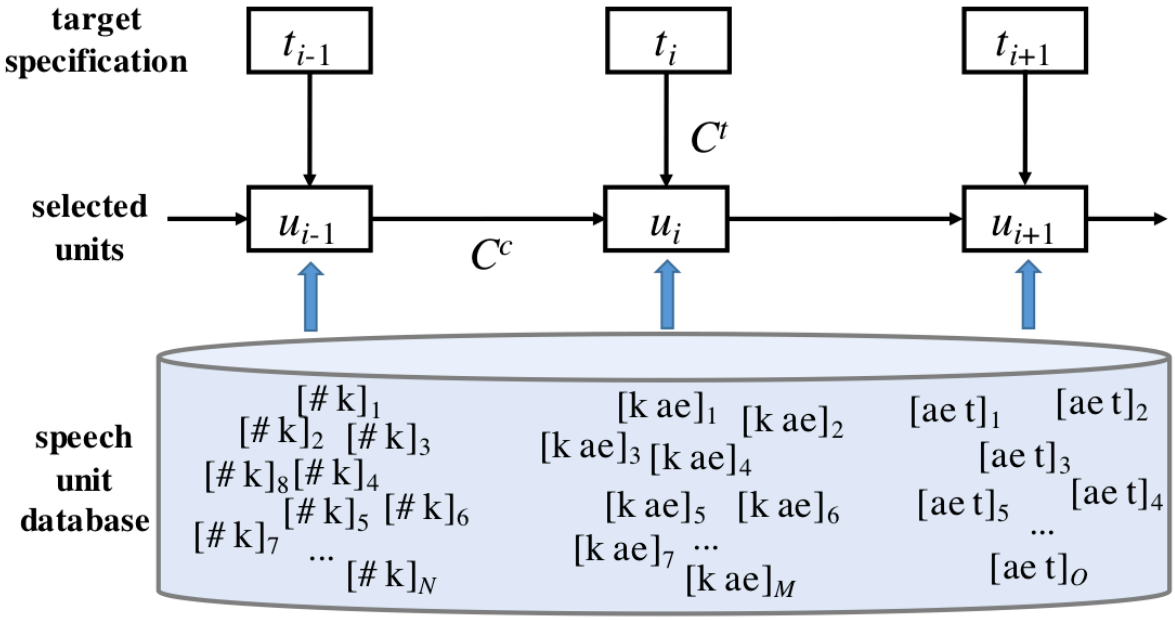



Figure 1: An illustration of the selection cost Ct and concenation cost Cc in diphone-based unit selection for synthesis of word "cat" [k ae t]. In practice, diphones with different initial phones but similar coarticulatory effects on the target phone can be considered in the selection process to overcome the issue of data sparsity. Adapted from [Hunt and Black, 1996].

Since the target specification consists of many characteristics such as target and context phone(s) identity, pitch, duration, and power, the target cost can be divided into multiple subcosts $j$ as $C_j^t(t_i, u_i)$ . For instance, the contextual phone(s) (e.g., [ae] in the last unit [ae t] of "cat" in Fig. 1) can be represented by a number of features describing the manner and place of articulation, so that the specification can be compared to different candidate units in the database [Hunt and Black, 1996]. This enables the use of different context phones with similar coarticulatory effects on the target phone. Similarly, costs for segment power and pitch can be measured, e.g., in terms of the differences in mean log-power and mean F0. Cost for the target phone ([t] in [ae t] of Fig. 1) is usually a binary indicator that forces the phonemic identity of the chosen unit to match with that of the target specification. The total cost can then be written as

$$C^t(t_i, u_i) = \sum_{j=1}^{P} w_j^t C_j^t(t_j, u_i) \tag{1}$$

where $w^t = [w_1^t, w_2^t, ..., w_P^t]$ are the relative weights of each subcost.

Concatenation cost $C^c(u_{i-1}, u_i)$ can be derived in a similar manner to Eq. (1) by decomposing the the total cost to $Q$ subcosts, and then calculating a weighted sum of the subcosts:

$$C^c(u_{i-1}, u_i) = \sum_{j=1}^{Q} w_j^c C_j^c(u_{i-1}, u_i)$$

(2)

Note that the subcosts and their weights $w^c$ for $C^c$ do not need match those of $C^t$, as the concatenation cost specifically focuses on the acoustic compatibility of the subsequent units. Therefore subcosts $C_j^c(u_{i-1}, u_i)$ associated with continuity of the spectrum (or cepstrum), segment power, and pitch in the segment and/or at the concenation point should be considered.

The total cost of the selection process is the sum of the target and concenation costs across all $n$ units:

$$\begin{aligned} C(t_1^n, u_1^n) &= \sum_{i=1}^{n} C^t(t_i, u_i) + \sum_{i=2}^{n} C^c(u_{i-1}, u_i) + C^c(\#, u_1) + C^c(u_n, \#) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{P} w_j^t C_j^t(t_i, u_i) + \sum_{i=2}^{n} \sum_{j=1}^{Q} w_j^c C_j^c(u_{i-1}, u_i) + C^c(\#, u_1) + C^c(u_n, \#) \end{aligned}$$

(3)

where $t_1^n$ are the targets, $u_1^n$ are the selected units, and # denotes silence [Hunt and Black, 1996]. The two extra terms stand for transition from preceding silence to the utterance and from utterance to the trailing silence. The aim of the selection process is then to find units $\bar{u}_1^n$ that minimize the total cost in Eq. (3). The selection process can be represented as a fully connected trellis, as shown in Fig. 3, where each edge to a node has a basic cost of the given node to be chosen (the target cost) and an additional cost depending on the previous unit (the concatenation cost). Given the trellis, the optimal selection can be carried out with *Viterbi search*—a dynamic programming algorithm that calculates the least cost path through the trellis. To make the search computationally feasible for large databases, less likely candidates for each target can be pruned from the trellis. In addition, *beam search* with only a fixed number of most likely nodes for each step can be applied for further speedup.

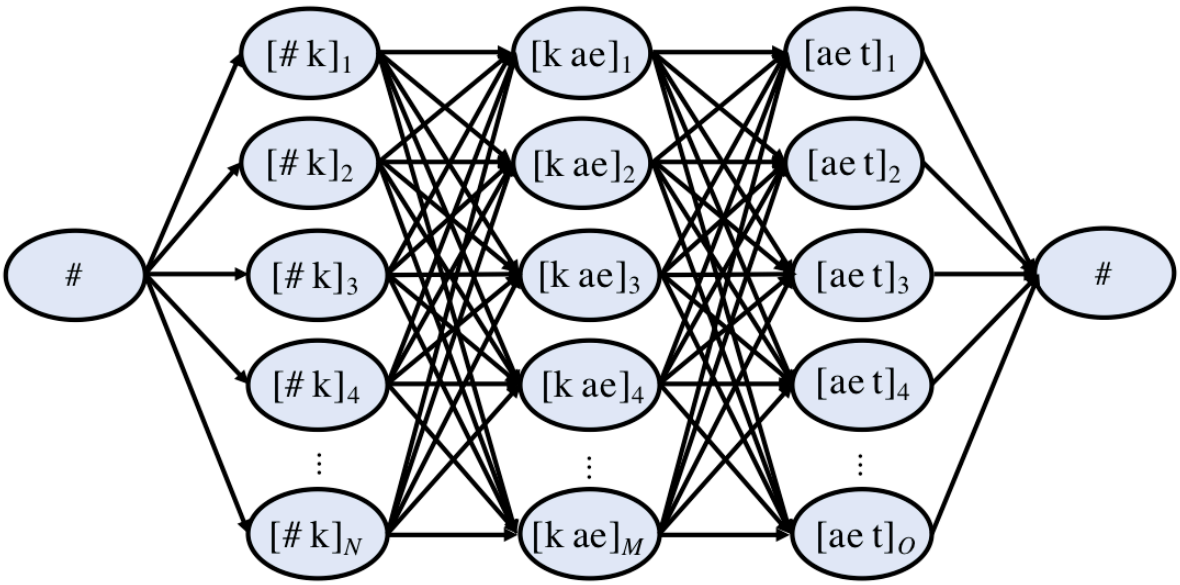



**Figure 3:** An example of unit selection search trellis for word "*cat*" [*k ae t*]. Each edge is associated with basic target cost of the selected unit and concatenation cost dependent on the previous unit. Adapted from [Rabiner and Schafer, 2007]

After the speech units have been concatenated to form the intended utterance, postprocessing techniques can be used to smooth the potential discontinuities in F0, energy and spectrum at the unit boundaries. Note that aggressive signal processing based modification of the segments also often tends to decrease the naturalness of the sound. Straightforward modification of the segments' acoustic parameters (e.g. with vocoding) is not therefore a recommended strategy to overcome the issues of poor unit selection or low quality source data.

### 9.1.2 CSS training

The above formulation enables mathematically principled and optimal unit selection process from a given speech database. However, the synthesis output is highly dependent on the choice of features and functions used for the subcosts, and also on the weights chosen for each feature. While the cost functions and their underlying features can be largely designed based on knowledge in signal processing and speech processing, the weights need to be either adjusted through trial and error, or they can be automatically optimized using some kind of quality criterion.

As examples of automatic weight estimation, [Hunt and Black, 1996] propose two alternative ways to automatically acquire cost function weights:

1) Using a grid search across different weight values by synthesizing utterances using the target specifications of held-out utterances from the training database, and then comparing the synthesized waveform to the real waveform of the held-out utterance using an objective metric. The weights that lead to the best overall performance are then chosen.

2) Using regression models to predict best values for the weight. [Hunt and Black, 1996] report that cepstral distance and power difference across the concanation point can be used as predictors for reported perceptual quality of the concatenation in a linear regression model, and therefore the linear regression weights can be directly used as perceptually motivated the cost weights. For the target weights, they propose and approach where each unit in the database is considered as the target specification at a time, and $K$ best matching other units are then searched for the target using an objective distance measure. Then the sub-costs between the target and the $K$ matches are calculated and recorded. This process is repeated for all exemplars of the same phonetic unit in the database, recording the $K \times Q$ subcosts and the related $K$ distances for each exemplar. Linear regression is then applied to predict the recorded objective perceptual distances using the associated sub-costs, linear regression coefficients again revealing the optimal weights for each of the subcosts. A specific advantage of the regression approach for subcost weight estimation is that it allows estimation of phoneme-specific weights for each subcost, as the perceptually critical cues may differ from a phonetic context to another.

### 9.1.3 Further reading

## 9.2 Statistical parametric speech synthesis

While concatenative synthesis can reach highly natural synthesized speech, the approach is inherently limited by properties of the speech corpus used for the unit selection process. Concatenative systems can only produce speech whose constituent segments (e.g., diphones) have been pre-recorded. In order to make the synthesis sound natural, large amounts of speech from a single speaker must therefore be available. This limits the flexiblity of concatenative systems in producing different voices, speaking styles, emotional expressions, or other modifications to the sound that are common in everyday human communication.

As an alternative to the concatenative approach, statistical parametric speech synthesis (SPSS) is another TTS approach that has become highly popular in the speech technology field. This is because it addresses the main limitation of the concatenative systems — the lack of flexibility — by generating the speech using statistical models of speech instead of relying on pre-recorded segments. These statistical models are learned from speech corpora using machine learning techniques, and they encode information of how speech evolves as a function of time in the context of a given input text. In this respect, SPSS systems can be viewed as a mirror image of ASR systems: while an ASR system tries to convert speech from acoustic features to a string of words using machine learning models, an SPSS system tries to convert a string of words into acoustic features or directly to the acoustic waveform using machine learning models. Both ASR and SPSS

systems are typically trained on a large amount of speech data with their transcriptions, resulting in a set of *parameters* that describe *statistical characteristics* of the speech data (hence "statistical parametric" speech synthesis).

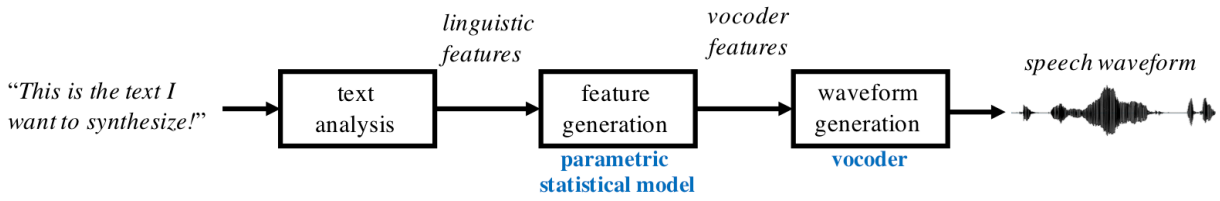



**Figure 1:** A schematic view of an SPSS system.

A full SPSS system consists of text analysis, feature generation, and waveform generation modules. The classical approach to SPSS is based on a combination of a *hidden-Markov model Gaussian mixture model* (HMM-GMM) architecture for feature generation and a *vocoder* for waveform generation, and these will be discussed in more detail below. Recent advances in neural network-based SPSS are then reviewed at the end.

### 9.2.1 Feature generation

Given the linguistic description of the text-to-be-synthesized, the purpose of the feature generation is to transform the linguistic features into a corresponding description of the acoustic signal. Similarly to ASR, this component mediating the two levels is called *an acoustic model*. Technically speaking, the acoustic model converts the linguistic input into a series of acoustic features at a fixed frame-rate (e.g., one feature frame every 10 ms) using a probabilistic mapping between the two. The mapping is learned from a training speech corpus.

A standard approach for the probabilistic mapping has been to use a HMM-GMM as the statistical parametric model. Similarly to an HMM-GMM ASR system, the states $s$ of the HMM correspond to parts of subword units (e.g., parts of a phone, diphone, or triphone). Transition probabilities P(s | st-1) between the states describe how the speech evolves through each subword unit and from a unit to another. Acoustic characteristics associated with each state are modeled with a GMM, where the GMM describes a probability distribution $P(y|s)$ over the possible acoustic feature vectors in that state. Given a sequence of desired subword units (as instructed by the linguistic features), the model can be stochastically or deterministically sampled to produce a sequence of acoustic features. These are then fed to *a waveform generation module* to produce the actual speech signal. In the most basic form, self-transitions from an HMM state to itself account for the duration spent in that state (i.e., how many frames should the same acoustic content be repeated). However, separate more advanced duration models are often used to overcome the limitations of a first-order Markov chain in modeling thetemporal dependencies and durational characteristics of speech.

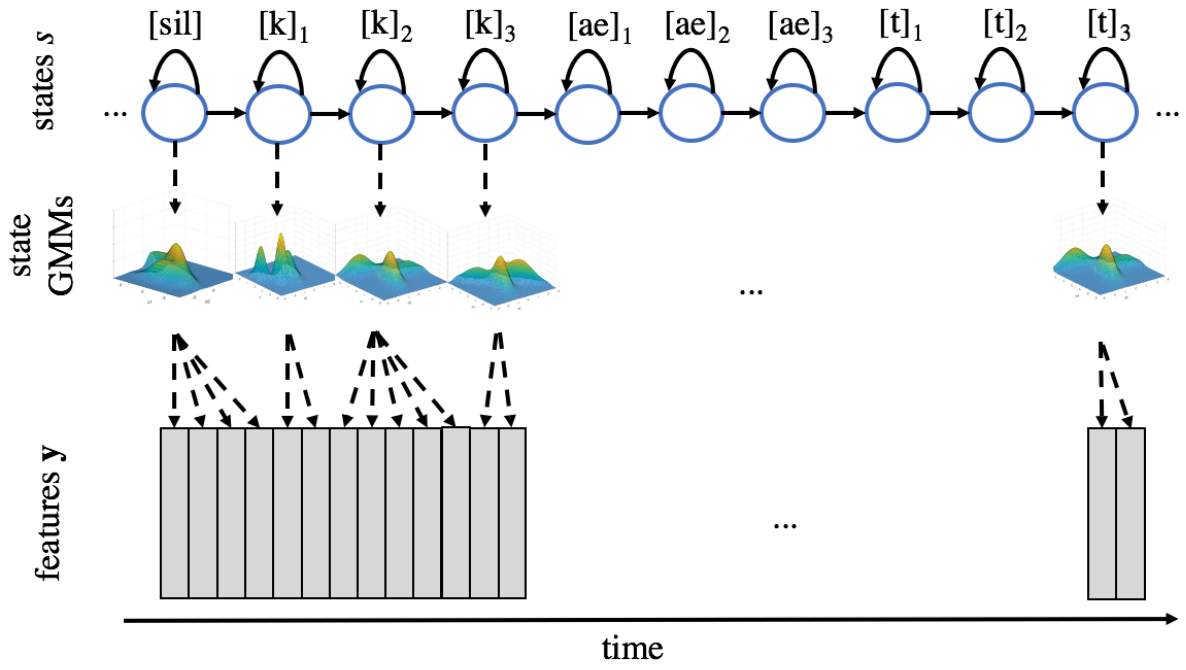



**Figure 2:** A visual illustration of HMM-GMM-based speech feature generation. State sequence $s = \{s_1, s_2, ..., s_{10}\}$ required for word "cat" (/k ae t/) is shown on top, where each phoneme consists of three states: initial, center and final state (e.g., $k_1$, $k_2$, and $k_3$). Each state is associated with an $N$-dimensional Gaussian mixture model (GMM), where $N$ is the dimensionality of the speech features $y$. At each time step, the GMM of the active state is sampled for a feature vector $y_t$. After this, a state transition can occur to a next state or back to the current state, controlling the durational aspects of the speech.

### 9.2.2 Waveform generation with vocoders

A typical high-quality speech waveform consists of "continuous" (e.g., 16-bit quantized) amplitude values sampled at 16 kHz. In addition, the shape of the waveform is affected by several factors that do not directly contribute to the naturalness or intelligibility of speech, such as signal gain or phase and amplitude characteristics of the recording and transmission chain. This means that mere 80 milliseconds of a raw waveform — a typical length of one vowel — would correspond to 0.08 s\16 kHz = 1280-dimensional amplitude vector, and that this vector could take countless of shapes for perceptually highly similar sounds. Moreover, the values encoded in this vector would be highly correlated with each other (see LPC). Given the high dimensionality, variability, and redundant nature of the waveform signal representation, it is not an attractive target for statistical parametric modeling with classical machine learning techniques (but see also Neural SPSS below).

However, as we remember from speech feature extraction (see, e.g., SFFT), speech signal can be considered as quasi-stationary in short windows of approx. 10–30 ms in duration. Speech contents of the signal within these short windows can be described using a set of spectral and source features (such as MFCCs and F0) that are assumed to be fixed for that window. When extracting the features in a sliding window with short (e.g., 10 ms) window steps, the overall structure of the signal can be captured with a much lower dimensional and less variable representation than what the actual waveform would be. **A vocoder, then, is an algorithm that can 1) parametrize a speech waveform into a more compact set of**

**descriptive features as a function of time, but also to 2) synthesize the speech back from the features with minimal loss in speech quality**. In addition, many vocoders use features that are interpretable in terms of speech production or speech acoustics, enabling analysis and manipulation of the speech signal to observe or cause certain phenomena in the speech signal.

Compactness and invariance of the acoustic signal representation is also why vocoding is used in SPSS systems: instead of generating the speech waveform directly, the feature generation module first generates a lower-dimensional set of vocoder features that characterize the speech signal with its essential properties. A vocoder then takes these features as input and generates the corresponding waveform using a series of signal processing operations. These operations are essentially an inverse of the original feature extraction process, combined with some additional mechanisms for re-introducing (or inventing) information lost during the feature extraction process (such as signal phase that is discarded from standard spectral features).

For instance, when using the popular STRAIGHT vocoder (Kawahara et al., 1999), the HMM-GMM model first generates a sequence of feature vectors that encode spectral envelope, F0, and periodicity characteristics of the speech signal to-be-produced, as instructed by the text analysis module. These features are then fed to STRAIGHT that synthesizes the final speech waveform based on the features.

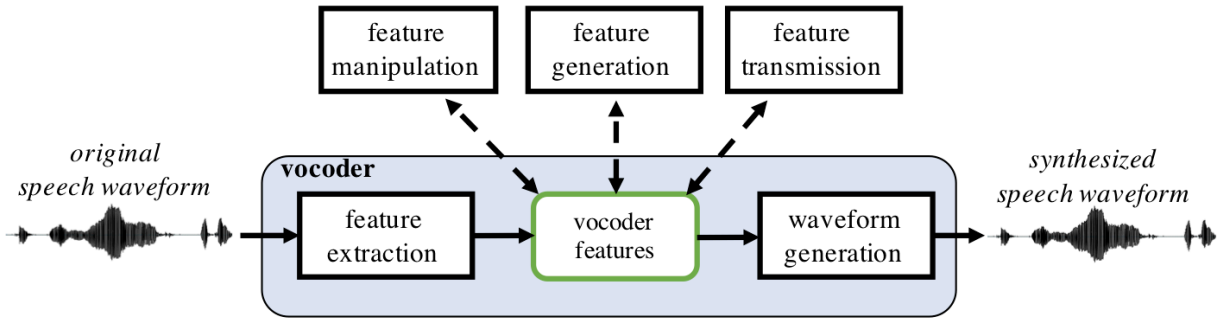



**Figure 3:** A schematic view of a vocoder and typical uses for vocoder features.When used as a part of an SPSS system, vocoder features are generated by the parametric statistical model during the synthesis process.

### 9.2.3 SPSS system training

Training of an SPSS system refers to estimation of the parametric acoustic model (e.g., a HMM-GMM) that is responsible for mapping the linguistic features to the corresponding waveform generation (vocoder) features. This is achieved using a corpus of speech data, where each utterance comes with the corresponding text of what was said, and optionally with phonetic annotation describing the phonetic units and their temporal positions in the waveform. First, the text analysis module is used to create linguistic features of a training utterance while the vocoder is used to extract vocoder features from the corresponding speech waveform. Then the statistical model is trained to minimize prediction error of the given vocoder features when the linguistic features are used as inputs. Access to phonetic annotation allows more accurate temporal alignment between the linguistic features and the speech signal. Since the widely utilized HMM architecture for acoustic modeling is not ideal for modeling speech segment durations, a separate *duration model* is often trained to align the linguistic features (which are agnostic of speaking rate and rhythm in the actual speech data) with the phonetic units realized in the acoustic speech signal. In the figure below, both the acoustic model and the duration model are denoted jointly by the parametric statistical model block.

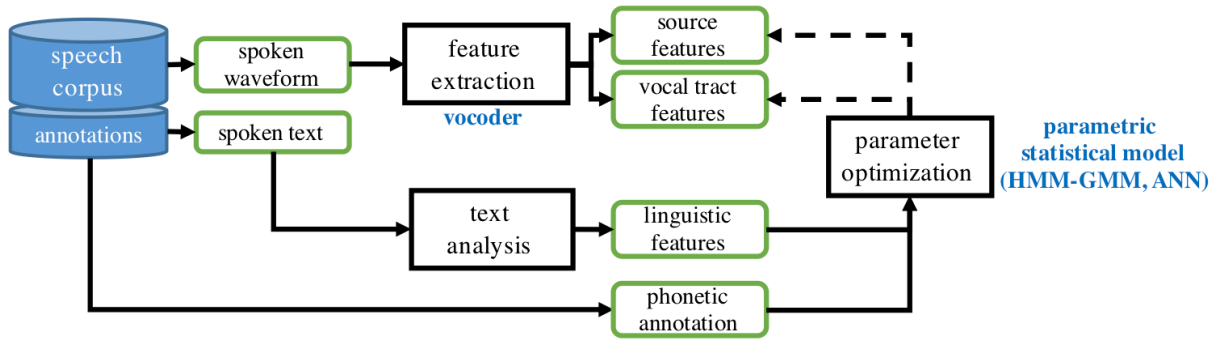



**Figure 4:** A schematic view of SPSS system training.

### 9.2.4 Advantages and disadvantages of the HMM-GMM SPSS compared to concatenative synthesis

Since the "instructions" for speech generation are encoded by parameters of the SPSS model, the model can easily be adapted to produce speech with different characteristics. For instance, the vocal tract characteristics of the training speaker are encoded by the means and variances of the Gaussian distributions in each HMM state whereas durational characteristics are encoded by the transition probabilty matrix of the HMM. Therefore, the system can be adapted to other speakers by simply adapting the pre-trained HMM-GMM using speech from a new talkers. In this case, standard techniques such as Maximum-a-posteriori (MAP) adaptation or Maximum-likelihood linear regression (MLLR) can be used to update the model parameters. In addition, since the parameters of the HMM-GMM are often interpretable in terms of speech spectral envelope or phonation characteristics, it is possible to either modify the models or to post-process the resulting acoustic features in order to achieve desired effects. For example, changing of the speech pitch can be done by simply adjusting the F0 parameter, whereas reduction of some synthesis artifacts such as muffled sound quality due to statistical averaging can be attempted by adjusting the GMM parameters with a chosen transformation.

The potential disadvantages of the statistical approach include sound quality issues (e.g., muffledness) due to statistical smoothing taking place in a stochastic generative model, sound quality of the used vocoders, and potential problems in robust statistical model estimation from finite data.

### 9.2.5 Neural SPSS

Recent advances in artificial neural networks (ANNs) have also led to new developments in SPSS beyond the classical HMM-GMM framework. In terms of vocoding, WaveNet (van den Oord et al., 2016) is a highly influential neural network waveform generator that can produce high-quality speech. It is based on an autoregressive convolutional neural network (CNN) architecture and it operates directly on the speech waveforms. Given a history of previous waveform samples and some "conditioning" information on what type of signal should be produced, the model predicts the most likely next speech waveform sample at each time step. For instance, WaveNet can be trained to produce speech from spectral features such as log-Mel energies and F0 information. Although WaveNet can reach near-human naturalness of the produced speech (with certain limitations), waveform-level autoregressive processing is also computationally extremely expensive. Development of computationally flexible high-quality neural vocoders is therefore still an active research area.

In addition to vocoding, neural networks have become commonplace replacements for the HMM-GMM in the feature generation stage. For instance, deep feed-forward networks or LSTMs can be utilized in the feature generation. Since LSTMs are especially good at modeling temporal dependencies, they can theoretically handle larger temporal ambiguity and variability between the input linguistic specifications and the target vocoder features.

Given sufficient training data, it is also possible to implement the entire chain from written text to the synthesized waveform using a neural network system. Tacotron 2 is an example of such a system, where the input text is processed by a sequence-to-sequence ANN model to directly create a log-Mel spectrogram corresponding to the input text (i.e., without a dedicated text analysis module). The spectrogram is then fed to the WaveNet module (see above) to produce the speech signal. As a

result, Tacotron 2 and the Wavenet vocoder can together achieve highly impressive speech quality. The advantage of these type end-to-end approaches is that there are fewer assumptions regarding what kind of intermediate representations are good for the task at hand, reducing the risk that the pre-specified operations and representations cause a loss of relevant information in the pipeline. There is also no need for deep understanding of the linguistic structure underlying written and spoken language or access to pre-existing text analysis tools, making deployment of the systems possible for any language with sufficient training data (text and corresponding speech). Since all the components are based on differentiable neural network operations, it is also possible to jointly optimize the entire chain from waveform generation to text processing. In principle, neural SPSS systems are also highly flexible, as basically any type of side information can be injected to the system to adjust the characteristics of the produced speech.

Neural systems, however, also have their drawbacks. The amount of data and computational resources required to train these systems can be high. Runtime computational requirements of neural vocoders may also be problematic in some applications, although recent advances in vocoders and in parallelization of the computations has already led to significant advances in this respect. Another issue is the lack of interpretability and transparency of model parameters: while parameters of classical models such as HMMs and GMMs have relatively clear relationship with what are the inputs and outputs of the system, the same is not true for ANNs with multiple layers. This makes it much more difficult to understand the behavior of the model, especially when trying to overcome problems in model performance. Lack of transparency and interpretability also means that manual control of characteristics of the produced speech is more difficult. Finally, adaptation of the models to new data (e.g., a new speaker or speaking style) cannot make use of the well-understood mathematical solutions available to the classical models. In contrast, the design, training, and adaptation of ANNs are much more heuristics-driven, similarly to the use of ANNs in any other machine learning domain.

### 9.2.6 Further reading

Kawahara, K., Masuda-Katsuse, I., and de Cheveigné , A. (1999). Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds, Speech Communication, 27, 187–207. (STRAIGHT vocoder)

Yamagishi, J. (2006). An introduction to HMM-based speech synthesis. https://wiki.inf.ed.ac.uk/pub/CSTR/TrajectoryModelling/HTS-Introduction.pdf (introduction to HMM-based SPSS)

Shen, J. et al. (2017). Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions. ArXiV pre-print: https://arxiv.org/abs/1712.05884 (Tacotron 2)

Tokuda, K., Nankaku, Y., Toda, T., Zen, H., Yamagishi, Y., and Oura, K. (2013). Speech synthesis based on hidden Markov models. *Proceedings of the IEEE*, 101, 1234–1252. (introduction to SPSS)**

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). *WaveNet: A generative model for raw audio.* * ArXiV pre-print: https://arxiv.org/pdf/1609.03499.pdf (WaveNet original paper)**

Wu, Z., Watts, O., and King, S. (2016). Merlin: An Open Source Neural Network Speech Synthesis System. *In Proc. 9th ISCA Speech Synthesis Workshop (SSW9)*, September 2016, Sunnyvale, CA, USA https://github.com/CSTR-Edinburgh/merlin (Merlin toolkit for synthesis)**

Zen, H., Tokuda, K., and Black, A. W. (2009). Statistical parametric speech synthesis. **Speech Communication**, 51, 1039–1064. https://www.sciencedirect.com/science/article/pii/S0167639309000648 (introduction to SPSS)**

CHAPTER

TEN

# TRANSMISSION, STORAGE AND TELECOMMUNICATION

1. *Design goals*
2. Basic tools
    1. *Modified-discrete cosine transform MDCT*
    2. *Entropy coding*
    3. *Perceptual modelling in speech and audio coding*
    4. *Vector quantization (VQ)*
    5. *Linear prediction*
3. *Code-excited linear prediction (CELP)*
4. *Frequency-domain coding*

## 10.1 Design goals

In short, the aim of speech coding methods is primarily to enable natural and efficient spoken communication over a geographical distance, given constraints on available resources. In other words, we want to be able to talk with a distant person with the aid of technology. Usually distance refers to location, but speech coding can be (and is often) used for storing speech signals (such that distance refers to distance in *time*). [Bäckström *et al.*, 2017]

In particular, aspects of quality which we can be included in our design goals are for example:

- *Acoustic quality* in the sense that the the reproduced acoustical signal should be similar to the original signal (measured for example in terms of signal to noise ratio or a perceptually weighted variant thereof).
- *Perceptual transparency* refers to a property of high-accuracy coding systems, where a human listener cannot perceive a difference between the original and reconstructed signals. When discussing transparency, we however need to accurately define the methodology with which we measure transparency. Namely, if we compare small segments of an original and reconstructed signals, we can easily hear minute differences, which would never be perceived in a realistic use-case.
- *Intelligibility *such that a listener can interpret the linguistic meaning of the reproduced signal.
- *Delay* in the communication path, end-to-end, should be within reasonable limits (e.g. below 150 ms). A higher delay can impede the naturalness of a dialogue.
- *Noisyness *caused by low-accuracy quantization and background noises should be minimized.
- *Distortions* of the speech signal to the amount the original signal is perceived to be changed by the processing.
- *Naturalness* of the speech signal refers to high natural vs. non-natural a speech signal sounds. For example, some types of non-natural speech signals could be such which sound robotic, metallic or muffled.

- *Listening effort* refers to the effort a listener needs to use a telecommunications system, and it should be minimized. Effort can be both due to properties of the user-interface, but importantly also, the listener might have to exert effort to understand a distorted speech signal.
- *Annoyance* is closely related to listening effort, in that a signal which is intelligible can have so severe distortions, noisiness or transmission delays that it is really annoying. Usually annoyance thus also increase listening effort.
- *Perception of distance* is the a feeling that the participant has of the distance between participants. The main contributor to the perception of distance is probably room acoustics, such that if the distance between speaker and microphone is large, than the listener feels distant to the speaker. It affects for example the intimacy of a discussion, such that it is hard to feel intimate if the distance is large.

It is clear that different types of quality are prominent at different levels of coding-accuracy, which in turn is a function of available bitrate (bandwidth). As a rough characterization, with current technology, the quality-issues we optimize at different bitrates are:

- At extremely low bitrates (below 1kbp/s), we cannot hope to encode speech at high quality. At best, we can hope to retain *intelligibility*.
- At very low bitrates (1-2 kbp/s), intelligibility can usually be preserved, but speech signals can still be distorted and noisy, such that we want to minimize *listening effort*.
- At low rates (3-8 kbps/s), we often have to balance between avoiding noisyness, distortions, annoyance and naturalness. In other words, we can often reduce noisyness by making the signal more muffled, but that would reduce naturalness. It is then very much a question of individual taste to choose which balance of distortions is best.
- At medium rates (8-16 kbp/s), speech signals can already be coded with a high quality such that we can try to minimize the number of audible (perceivable) distortions. At these rates telecommunication systems can often, in practice, be transparent in the sense that users are not actively aware of any distortions, even if they would clearly notice distortions in a comparison with the original signal.
- At high rates (above 16 kbp/s), it should in general be possible to encode speech signals at a perceptual transparent level. However, if computational resources are limited and in applications which require extremely low delay, distortions can still be audible.

Performance of a codec is however always a compromise between quality and resources. By increasing the amount of computational resources (or bandwidth) we can improve quality ad infinitum. The most important limited resources are

- Bandwidth, that is, the bit-rate at which we can transmit data. It is limited by
  - physical constraints such as available radio channels,
  - power consumption (battery, ecology and price) and
  - infrastructure capacity (investment, complexity, power).
- CPU, that is, the amount of operations per second that can be performed, which is further limited by
  - investment cost and
  - power consumption (battery, ecology and price).
- Memory, that is, the amount of RAM and ROM which is needed for the system, which are limited by
  - investment cost and
  - power consumption (battery, ecology and price).

Furthermore, the use-case of the intended speech (and audio) codec has many important effects on the overall design. For example, the systems configuration can be one of the following:

- *One-to-one*; the classic telephony conversation, where two phones transmit speech between them.
- *One-to-many*; could be applicable for example in a setting like a radio-broadcast or in a storage application, where we encode once and have potentially multiple receivers/decoders. Since there are many receivers, we would then

prefer that decoding the signal does not require much resources. In practice that means that the sender side (encoder) can use proportionally more resources.

- *Many-to-many;* the typical teleconferencing application, often implemented with a cloud-server, such that merging of individual speakers can be done centrally, such that bandwidth to the many receivers can be saved.
- *Many-to-one; *could be a distributed sensor-array scenario, where multiple devices in a room jointly record speech. Since we then have many encoders, they should be very simple and a majority of the intelligence and computational resources should be at the receiver end.

The overall design is also influenced by the type of transmission link. In particular, the first few generations of digital mobile phones operated with circuited-switched networks, where a fixed amount of bandwidth is allocated to every connection. Newer networks are however based on packet-switched designs, where data is transmitted essentially over the internet and capacity and routing is optimized on the fly. Packet-switched networks can in practice be much better optimized for overall cost and performance. However, a packet-switched network cannot guarantee a steady flow of packets, such that the receiver has to tolerate both delayed or missing packets as well as packets which arrive in the wrong order. Clearly this has an impact on both overall transmission delay of the system, as well as increases the computational complexity of the receiver. The costs are however usually balanced by the savings gained in network optimization.

A further important related aspect are assumptions about lost packets in general. In many storage and broadcast applications we can assume that packets are not lost and that all data is available at the receiver. It however much more common that we must assume that some packets are lost. Among the most important consequences of lost packets for the design is that in decoding the signal, we cannot assume that we have access to previous packets. Specifically, if decoding of the current packet depends on the previous packet, then a single lost packet would make us unable to decode any of the following packets. Clearly such sensitivity to lost packets is unacceptable in most real-world transmission systems. However, we could encode speech with much higher efficiency, if we were allowed to use previous packets to predict the current packet. The likelihood of lost packets thus dictates the compromise between sensitivity to lost packets and coding (compression) efficiency.

### 10.1.1 References

## 10.2 Modified discrete cosine transform (MDCT)

### 10.2.1 Introduction

In processing speech and audio, we often need to split the signal into segments or *windows*, because

- audio signals are relatively slowly changing over time, such that by segmenting the signal in short windows allows assuming that the signal is stationary, which is a pre-requisite of many efficient methods such as *spectral analysis*,
- transform-domain analysis, such as spectral analysis, requires that we transform the entire signal in one operation, which in turn requires that we have received the entire signal, before we can start processing. In for example telecommunications, this would mean that we have to wait for a sentence to finish before we can begin sending, and thus reconstruction the sentence at the receiver would have to wait until the whole sentence has finished, incurring a significant delay in communication.

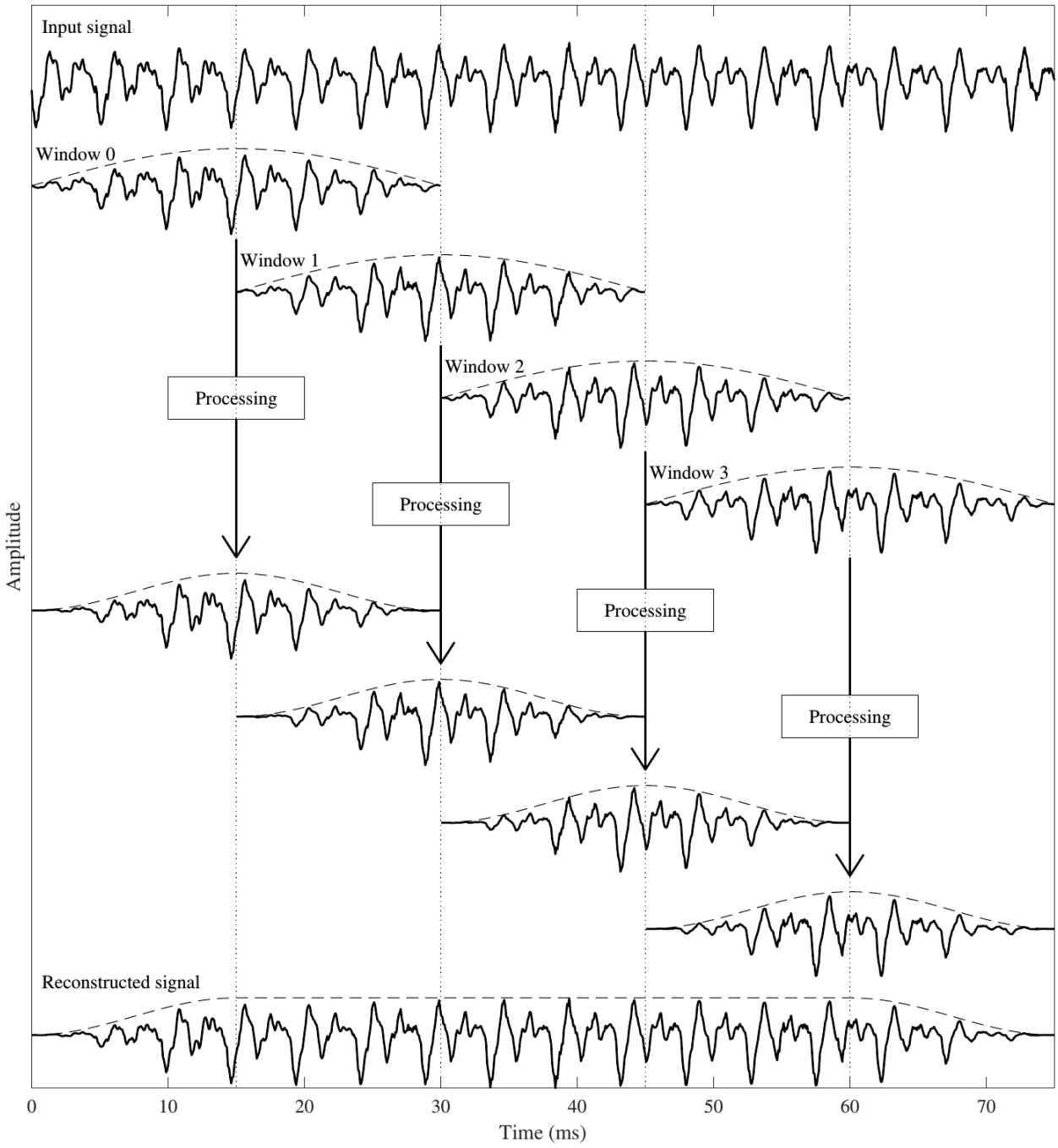
In practical scenarios, we thus always apply windowing before processing.

Applying spectral processing, such as processing in the *STFT* domain, in *telecommunications applications* however has a significant drawback. We need windows which are *overlapping* to allow *perfect reconstruction*, that is, we require that we can perfectly reconstruct the original signal from the windowed signal. Overlaps however mean that consecutive windows share information from the same samples (see Figure on the right). The same information thus has to be encoded in *both windows*, which is inefficient. With coding a signal, our target is to compress the information to the fewest possible bits, but if we encode a part of the information twice, we increase the amount of information we need to encode. The STFT domain thus causes *over-coding*, where in effect, two different bit-streams could represent the same output signal.

The *modified discrete cosine transform* (MDCT) is a solution to the over-coding problem, which uses projection-operators at the overlaps such that the information in consecutive windows are orthogonal to each other. Since information is therefore perfectly retained, the amount of information is not increased nor decreased, and we say that this operator is *critically sampled*.

The MDCT thus provides a time-frequency representation of the input signal, where we can analyse and process the time-evolution of frequency-components. It shares most of the beneficial properties of the STFT, wherein a signal processed in the MDCT-domain, remains continuous in the time-domain and we can use FFT-based operations for efficient implementations. Perhaps the most significant difference between the STFT and MDCT, other than perfect sampling of the MDCT, is that where the STFT is a complex-valued representation, the MDCT is a real-valued representation when the input is real-valued.

Due to these beneficial properties of the MDCT, it is the most commonly used time-frequency transform in speech and audio coding and employed in standardized codecs such as MPEG USAC, 3GPP EVS and Bluetooth LC3.


Input signal
Window 0
Window 1
Window 2
Window 3
Processing
Processing
Processing
Processing
Reconstructed signal
Amplitude
0
10
20
30
40
50
60
70
Time (ms)

### 10.2.2 Executive summary of algorithm

The MDCT is based on taking the symmetric and anti-symmetric parts of the signal at the overlap, such that the symmetric part goes to the one window and the antisymmetric to the other. We furthermore window those symmetric and antisymmetric parts such that they converge smoothly to zero at the borders. Consecutive windows are thus orthogonal at the overlap such that information is split exactly in half and we obtain perfect reconstruction and critical sampling. Moreover, since the segments are windowed, we do not have any difficulties with discontinuities.

Secondly, we take the discrete cosine transform (DCT) of the windowed signal (=of the orthogonal projection). The conventional DCT has to, however, be adjusted such that its phase and symmetries match with the symmetric and anti-symmetric parts. Hence the name *modified DCT or MDCT*. The matching of symmetries is possible only through this modification of the DCT and cannot be generalized to the discrete Fourier transform (DFT).

The (anti)symmetric part of signal then "looks like the signal" which would have this symmetry. In other words, we introduce a *deviation* from the original signal and this deviation is known as a *time-domain aliasing* component, since it is similar to the aliasing effects which occur with the DFT. However, in reconstruction, we add the symmetric and anti-symmetric parts together, such that they cancel each other, which is known as *time-domain aliasing cancellation (TDAC)*. The TDAC is equivalent with the orthogonal projection laying at the basis of the MDCT, and thus a central property of the MDCT.

### 10.2.3 Definition

Suppose $x$ is a $N \times 1$ vector representing the overlapping region shared by two consecutive windows, such as the are between 15 and 30 ms in the figure above and right. Our task is to design a projection matrix $P$ of size $N \times N$, such that we can split $x$ into two orthogonal parts as

$$\begin{cases} x_L &= Px \\ x_R &= P_0 x \end{cases}$$

where the $P_0$ is the kernel of $P$ and thus $P_0 P^T = 0$, such that we can reconstruct $x$ as

$$P^T x_L + P_0^T x_R = P^T P x + P_0^T P_0 x = (P^T P + P_0^T P_0)x = x,$$

since $P^T P + P_0^T P_0 = I$ by definition.

The vectors $x_L$ and $x_R$, corresponding to the left and right windows, are thus both of length N/2 and contain each exactly half the information of $x$.

Though this is a mathematically clean solution, it is not practical, since after processing the reconstructed signal could have discontinuities. For example, suppose that we define $P$ as the operator which extracts the symmetric part of $x$, that is, $P = [I, J]\sqrt{1/2}$, where $I$ is the $(N/2) \times (N/2)$ identity matrix and $J$ its left-right flipped counterpart (reverse diagonal). The kernel is then $P_0 = [I, -J]\sqrt{1/2}$. If we make a small modification to the signal as $x'_L := x_L + d$, then the reconstruction is $x' := P^T x_L + P^T d + P_0^T x_R = x + P^T d$, which means that original signal is changed by $P^T d$. Unfortunately, however, this component does not go to zero at the border such that processed signal is not continuous. Specifically, for example, consider a vector $d = [1,\ 1,\ \ldots,\ 1]^T$, which gives $P^T d = \sqrt{1/2}[1,\ 1,\ \ldots,\ 1]^T$, which is clearly non-zero at both ends of the vector.

The reason for this problem is that the definition did not include a windowing function. Recall that windowing is sample-by-sample multiplication of the signal with a windowing function. In matrix notation, we can implement this by multiplying the input signal $x$ with a diagonal matrix $W$, where the diagonal elements correspond to the windowing function. The windowed signal is then $Wx$. The symmetric projection $P$ can then be applied on the windowed signal such that the windowed left and right signals are

$$\begin{cases} x_L &= PWx := P'x \\ x_R &= P'_0 x = P_0 JWJx \end{cases}$$

where the $P_0'$ is the kernel of $P'$, and $JWJ$ is simply the backwards part of the window (if $W$ is the increasing left part of the window, then $JWJ$ is the decreasing right part of the window). Note that the relationship between the two kernels, $P_0' = P_0 JWJ$ applies only to the symmetric operator defined above, but generally not. In any case, the above definition now includes windowing and the orthonormal projection such that the beneficial properties of perfect reconstruction and critical sampling are retained, but now the signal is also going smoothly to zero at the edges, because we applied a windowing function.

Above we have thus developed a windowing operation which has the properties we need. However, our goal was to obtain a time-frequency transform of the signal, so we are still missing the time-frequency transform.

Our objective is to find a spectral representation of the windowed input signal. Importantly, note that above we studied the properties at the overlap, where we took an orthonormal projection of the overlap to obtain the left-part of the current window $x_L$ and right-part of the preceeding window $x_R$. To avoid confusion, we here have to include the frame indices, such that the overlapping parts are now, respectively, $x_{L,k}$ and $x_{R,k-1}$. The spectral representation, however, should come from a single frame, that is, we define frame $k$ as

$$\hat{x}_k := \left[x_{L,k}\, x_{R,k}\right].$$

To obtain the spectrum of $\hat{x}_k$ we can then multiply it with a time-frequency transform matrix $D$, such that the frequency representation is $y_k = D\hat{x}_k$ and the reconstruction is $\hat{x}_k = D^T y_k$, assuming that $D$ is orthogonal (as time-frequency transforms usually are).

In principle, the transform matrix $D$ could be any time-frequency transform (DFT, DCT-I, DCT-II, DST-I etc.), but we need to choose one specific. To evaluate the sanity of a particular choice, we should check how well the transform works together with the prior orthogonal projections. In particular, let us substitute the defitions of the left and right parts into the above equation to obtain

$$\hat{x}_k := \begin{bmatrix} x_{L,k} \\ x_{R,k} \end{bmatrix} = \begin{bmatrix} PWx_k \\ P_0 JWJ x_{k+1} \end{bmatrix} = \begin{bmatrix} PW & P_0 JWJ \end{bmatrix} \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} := \hat{P} \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix}.$$

The combined windowing+projection+spectral transform is thus $y_k = D\hat{P} \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} = M \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix}$. To evaluate the sanity of a particular time-frequency transform, we thus have to study the basis functions of matrix $M$. By exhaustively going through different options, it is simple to see that the implicit even/odd extensions of DCTs align with the symmetric/antisymmetric projections only for DCT-III. For the other options, we always get discontinuities in the basis functions.

Heuristically speaking, we want the basis functions to "look like" plausible signals, which could appear in a naturally appearing physical system. More formally, such "nice looking" basis functions have less leakage between spectral bins, and thus offer a more accurate description of the spectral characteristics of the input signal. [Bäckström *et al.*, 2017]

See also https://github.com/audiolabs/lapped-transforms

### 10.2.4 References

### 10.2.5 Entropy coding

In transmission and storage of data, it is useful if we can minimize the number of bits needed to uniquely represent the input. With *entropy coding*, we refer to methods which use statistical methods to compress data. The target is *lossless* encoding, where the original data can be perfectly reconstructed from the compressed representation. With *lossy* coding, similarly, we refer to compression where, for example, we have a limited number of bits to use and we try to reproduce a signal as similar as possible to the original, but not necessarily exactly the same. In speech and audio, coding usually refers to lossy coding. The objective is to compress the coded signal such that it remains perceptually indistinguishable from the original or such that the perceptual effect of quantization is minimized. Such coding is known as perceptual coding. Often, perceptual coding is performed in two steps; 1) quantization of the signal such that the perceptually degrading

effect of quantization is minimized and 2) lossless coding of the quantized signal. In this sense, even if lossless and lossy coding are clearly different methods, a lossless coding module is often included also in lossy codecs.

Entropy coding operates on an abstract level such that it can operate on any set of symbols as long as we have information about the probabilities of each symbol. For example, consider a integer-valued scalar $x$, which can attain values -1, 0, and +1, with respecitve probabilities 0.25, 0.5, 0.25. That is, if we repeatedly draw scalars $x$ from this distribution, then on average, 25% of them are -1's. It is then irrelevant what the numerical values of $x$ are, we can equivalently name the distinct elements according to symbols of the alphabet as $a, b$ and $c$.The table on the right demonstrates a possible encoding of these numbers. Clearly we need more than one bit to encode three symbols, and hence this encoding uses 2 bits per symbol. Observe, however, that the bit-string 11 is not used, which means that the encoding is inefficient.

A naive encoding of a set of numbers, with 2 bits per symbol.

| number | symbol | bit-string | length |
|---|---|---|---|
| -1 | a | 00 | 2 |
| 0 | b | 01 | 2 |
| +1 | c | 10 | 2 |

### Vector coding

To take better use of all bits, we can instead of single symbols, consider a vector of symbols $x_1, x_2, x_3$ . With 3 possible symbols for each element, we have $3^3 = 27$ possible combinations. To encode it we thus need ceil$(\log_2(27)) = 5$ bits, or 1.66 bits per sample. In comparison to the original 2 bits per sample above, this is a clear improvement. However, we still have 5 unused bit-strings, which shows that this encoding is sub-optimal.

Note that there are immediate parallels with *vector quantization (VQ)* though the two methods are not the same. In short, vector quantization is lossy coding, which finds the best quantization with a given set of symbols, whereas vector coding is lossless coding of vectors of symbols.

A naive encoding of a set of numbers, with 2 bits per symbol.

| numbers | symbols | bit-string | length |
|---|---|---|---|
| -1, -1, -1 | aaa | 00000 | 5 |
| -1, -1, 0 | aab | 00001 | 5 |
| -1, -1, +1 | aac | 00010 | 5 |
| -1, 0 -1 | aba | 00011 | 5 |
| -1, 0, 0 | abb | 00100 | 5 |
| … | … | … | … |
| +1, +1, +1 | ccc | 11011 | 5 |

### Variable length and Huffman coding

A central concept in entropy coding *variable length* coding, where symbols can be encoded with bit-strings of varying lengths. In our above example, an optimal coding (i.e. optimal bit-strings) is listed in the table on the right.

The average number of bits per symbol is then the sum of the length of each bit-string multiplied with the corresponding probability, that is,

$$E[bits/symbol] = \sum_{k \in \{a,b,c\}} P_k L_k = 0.25 \times 2 + 0.5 \times 1 + 0.25 \times 2 = 1.5.$$

From the bit-strings 00, 01 and 1, we can clearly decode the original symbols; if the first bit is one, then the symbol is $b$, otherwise the second bit determines whether the symbol is $a$ or $c$.

Such variable length codes can be readily constructed when the probabilities are negative powers of 2. This is a classic approach known as *Huffman* coding. It is very simple to implement, which makes it an attractive choice when probabilities are negative powers of 2. However, when the probabilities of the symbols are arbitrary, then Huffman coding is no longer applicable without approximations of probabilities, which make the coding suboptimal.

An illustrative encoding of a set of numbers, with 1.5 bits per symbol.

| number | symbol | probability $P_k$ | bit-string | length $L_k$ |
|---|---|---|---|---|
| -1 | a | 0.25 | 00 | 2 |
| 0 | b | 0.5 | 1 | 1 |
| +1 | c | 0.25 | 01 | 2 |

### Arithmetic coding

To further improve on coding efficiency, we can combine vector coding with Huffman coding in a method known as *arithmetic coding*. It uses the probability of symbols to jointly encode a sequence symbols. For example, consider the set of symbols 0….5 on the right with corresponding occurrence probabilities $P_k$. Further suppose that we are supposed to encode the string "130". The first step is to assign every symbol to a unique segment $[s_k,\ s_{k+1}]$ of the interval $[0,\ 1]$ such that the width of the segment matches the probability of the symbol $P_k = s_{k+1} - s_k$ .

The first symbol is "1" whereby we are assigned to the interval 0.40 … 0.67, which we will call the current interval. The central idea of arithmetic coding is that next symbol is encoded inside the current interval. That is, we shift and scale the $s_k$'s such that they perfectly fit within the current interval 0.40 … 0.67. In mathematical terms, if the current symbol is $h$, then the current interval is $s_h ... s_{h+1}$, and the intervals of the next symbol are shifted and scaled as

$$s'_k = s_h + s_k(s_{h+1} - s_h) = s_h + s_k P_k.$$

The second symbol was "3", such that the current interval is 0.6133 … 0.6349. For the third symbol, the intervals are then shifted and scaled as

$$s''_k = s'_3 + s'_k(s_4 - s_3) = 0.6133 + s'_k \times 0.08 \times 0.27.$$

The new intervals are listed on the right. The third symbol is "0" such that the last current interval is 0.6133 … 0.6219, which we will denote as $s_{left} ... s_{right}$ .

The remaining step is to translate the last interval into a string of bits. Let us divide the whole interval 0 … 1 into $2^B$ quantization levels on a uniform grid. such that the $k$th level is $k2^{-B}$. We then find the largest $B$ such that there is a $k$ with which $k2^{-B}$ is inside the last current segment $s_{left} ... s_{right}$ that is, $s_{left} \leq k2^{-B} \leq s_{right}$ . Then $k$ is the index to our quantization position, that is, it uniquely describes the interval and thus uniquely describes the sequence of symbols "130". Specifically, with $B = 7$, we find that $k = 79$, fulfills the criteria

$$s_{left} = 0.6133 \leq k2^{-B} = 0.6172 \leq s_{right} = 0.6219.$$

Decoding the sequence is then straightforward at the decoder.

A few additional points:

- The average bit-rate is $-\sum_k P_k \log_2 P_k \approx 2.2$ bits per sample. In the example above we needed B=7 bits to encode 3 samples, which gives 2.3 bits per sample. The actual number of bits thus does not perfectly coincide with the average bit-rate.
- In a practical implementation of a decoder, we need to either know the number of symbols or bits, transmit the number of symbols or bits separately, or we need to use a special symbol which signifies end-of-string.
- Usually the last current segment does not exactly align with $k2^{-B}$ , which means that there are small unused spaces in between the bitstring and the last current segment. This is an inherent inefficiency of arithmetic coding. Heuristically it is easy to understand that we must send an integer number of bits, but the data might not be exactly

an integer number of bits. The loss is always less than 1 bit for the whole string, which is acceptable when we send a large amount of symbols in a string.

- Direct implementation of the above description would be cumbersome since the intervals rapidly become smaller than what can be expressed by discrete arithmetic (fixed or floating point). Usually therefore algorithms are designed to use an intermediate interval, from which we output bits once they become known. For example, in the above example, after the first symbol we already know that the interval is above 0.5, such that the first bit has to be 1, corresponding to the interval 0.5 … 0.1.
- The specific implementation is rather involved and sensitive to errors.
- Algorithmic complexity of an arithmetic coder is usually reasonable, provided that the probabilities $P_k$ are readily available. If the probabilities need to be calculated online (parametric probability model), then complexity increases considerably.

Illustrative set of symbols and their corresponding probabilities.

| symbol $k$ | probability $P_k$ | interval $s_k \ldots s_{k+1}$ | bits per symbol $\log_2(P_k)$ |
|---|---|---|---|
| 0 | 0.40 | 0.00 … 0.40 | 1.32 |
| 1 | 0.27 | 0.40 … 0.67 | 1.89 |
| 2 | 0.12 | 0.67 … 0.79 | 3.05 |
| 3 | 0.08 | 0.79 … 0.87 | 3.64 |
| 4 | 0.07 | 0.87 … 0.94 | 3.84 |
| 5 | 0.06 | 0.94 … 1.00 | 4.06 |

The intervals of the second symbol.

| symbol $k$ | interval $s'_k \ldots s'_{k+1}$ |
|---|---|
| 0 | 0.4000 … 0.5080 |
| 1 | 0.5080 … 0.5809 |
| 2 | 0.5809 … 0.6133 |
| 3 | 0.6133 … 0.6349 |
| 4 | 0.6349 … 0.6538 |
| 5 | 0.6538 … 0.6700 |

The intervals of the third symbol.

| symbol $k$ | interval $s''_k \ldots s''_{k+1}$ |
|---|---|
| 0 | 0.6133 … 0.6219 |
| 1 | 0.6219 … 0.6278 |
| 2 | 0.6278 … 0.6304 |
| 3 | 0.6404 … 0.6321 |
| 4 | 0.6321 … 0.6336 |
| 5 | 0.6336 … 0.6439 |

### Parametric coding

In the examples above, we used a table to list the probabilities of each symbol. That leads to a rigid system, which cannot adapt to changes in the signal. If we want to, for example, use a perceptual model to choose the quantization accuracy of different samples, we need the ability to adapt quantization bin widths and consequently, to adapt the probabilities of each quantization bin. A simple approach is to model the probability distribution of the signal and calculate probabilities of each symbol on-line. We thus use a parametric model of the probability distribution and correspondingly, we call a coding with parametric models a *parametric coding*.

In speech coding, parametric coding is typically used in frequency-domain coding to encode individual spectral components. We can then assume that spectral components follow a Laplacian distribution and derive the probabilities $P_k$ using that distribution. More refined alternatives include for example Gaussian mixture models (GMMs).

### Algebraic coding

Suppose we would like to encode a string like "00010000", where "0"s happen with a high likelihood and there is only a single "1". We could then use arithmetic and parametric coding to encode the probabilities of 0's and 1's to develop an output string. It quickly becomes awfully complex, when it would be much simpler to just encode the position of the single "1". In this case, if the first position is 0, then the single "1" is at position 3. There are a total of 8 positions, such that we need 3 bits to encode the position. Very simple.

This approach can be readily extended to encompass for example both +1 and -1's. Just encode the sign with a single extra bit. There exists straightforward algorithms to include multiple non-zero elements as well, as long as the number of non-zeros is small.

Such encoding algorithms are known as *algebraic coding*, where we use an algebraic rule or explicit algorithms to encode strings. This is one type of vector coding, since it encodes jointly a string of symbols. Usually algebraic coding is fixed bitrate coding, since the number of bits is decided in advance.

Algebraic coding works efficiently when there are a low number of non-zeros and the number of quantization levels is very low. Unfortunately these methods become increasingly complicated when higher accuracy is required. Moreover, for higher accuracy quantization, it becomes increasingly difficult to find the best quantization of a given vector $x$. Still, due to its simplicity and efficiency at low bitrates, algebraic coding is so popular in speech coding that the most commonly used codec type is known as Algebraic code-excited linear prediction (ACELP), since it uses algebraic coding to encode the residual signal.

## 10.2.6 Perceptual modelling in speech and audio coding

Humans are usually the intended recipients of speech signals in telecommunication, such that the quality of a transmission should be measured in terms of how good a human listener would judge its quality. *Perceptual models* refer to methods which try to approximate or predict the judgement of auditory quality perceived by human listeners. In coding applications we can thus define perceptual models as *evaluation models*, with which we approximate the perceptual effect of distortions.

An another type of models which are frequently used in speech coding are *source models*, which describe the inherent characteristics of the source, which is the speech signal. You can think of a source model as for example 1) physical models, which describe the physiological processes which cause speech sounds or 2) the probability distribution of speech signals. The important distinction is that source models do not care about who is observing, but they only describe the objective reality. In contrast, perceptual models are applied when *we subjectively observe* the signal, to evaluate properties of the signal.

In speech and audio coding applications, practically all distortions caused by the algorithms are due to quantization of the signal. The objective of perceptual modelling is then to choose the quantization accuracy such that the perceptually degrading effect of quantization is minimized. Roughly speaking, this means that those signal components which are more important to a human listener are quantized with a higher accuracy than those which are less important.

### Frequency masking

If we play two sinusoid with slightly different frequencies, then the louder of the two can *mask* the second sinusoid such that it becomes inaudible. This effect is known as *frequency masking*. In other words, people are less sensitive to sounds which are near in frequency to other sounds. In particular, when quantizing a signal, we can use a lower quantization accuracy in frequency-regions which have more energy. The effect is reduced the further away we are in frequency.

In practice, frequency masking models are similar to spectral (energy) envelopes. That is, the shape of the frequency masking model is similar to the spectral envelope, but a smoothed and less pronounced version thereof. More accurate versions of the model can be generated based on psychoacoustic theory.

Frequency masking models are used in two ways:

- In frequency-domain codecs, where a frequency-domain representation of the signal is quantized, we choose the quantization accuracy in different regions of the spectrum based on a perceptual model. Typically high-energy regions are quantized with less accuracy than low-energy regions.
- In time-domain codecs such as CELP, we typically use a analysis-by-synthesis loop, where different quantized versions are synthesized and the error between original and quantized signal is determined with perceptual weighting. The weighting is here based on a frequency masking model. Out of the different possible quantizations, the one with the smallest perceptually weighted error is chosen.

### Frequency scale

The sensitivity of human hearing depends on the frequency range where the sound is present. We are more sensitive in the "low" regions and less sensitive at "high" frequencies. There is however some ambiguity in how sensitivity is defined, and we have two prominent different interpretations:

- In the cochlea of the human ear, sounds are processed in spectral bands, which are independent such that sounds in separate bands do not interfere with each other, but sounds within the same band *do interfere* with the perception of each other. This is known as auditory masking. The width of these bands is frequency dependent and increases with increasing frequency. This aspect of perception has been approximated with several models, including the Bark and ERB scales.
- The distance between pitches are perceived differently depending on their frequencies. In short, a perceptually small step in pitch (measured in frequencies) is much larger at higher frequencies than at low frequencies. This aspect of perception can be approximated with the Mel scale.

### Temporal masking

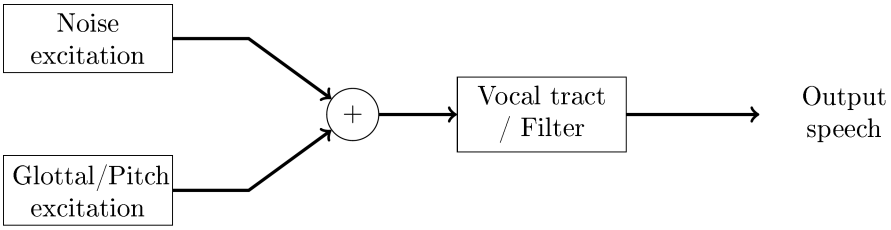
The variation in accuracy and sensitivity of perception is interesting also across time. In particular, a loud sound can make imperceptible a second, weaker sound which comes later in time. Say, if we have two impulses, consecutive in time, and such that the second one is weaker, and their distance in time is sufficiently short, then we cannot hear the second impulse. Surprisingly, such temporal masking can occur also the other way around, a *later* loud sound can mask a preceding weaker sound.

## 10.3 Code-excited linear prediction (CELP)

The most famous speech coding paradigm is code-excited linear prediction (CELP). It was first invented in 1985 and is the basis of all main-stream codecs dedicated to speech. Its most prominent variant is the algebraic CELP, which uses an algebraic codebook to encode the noise residual. Codecs such as AMR, EVS, G.718 and Speex (superseded by Opus) are all based on variants of CELP.

As an overview, CELP is based on a source-filter model of speech, where linear prediction is used to model the filtering effect of the vocal tract (and other effects) and this filter is excited by the speech source, viz. the glottal source and turbulent noise. Typically, the pitch or fundamental frequency model is a long-term prediction (LTP) filter, which is just a linear predictor with a long delay. To model noise, CELP codecs usually use a vector codebook. The codebook contribution is often optimized with analysis-by-synthesis, where the output of different quantizations are synthesised and the synthesised outputs are evaluated to choose the best quantization. The evaluation uses *perceptual weighting* such that the subjective, perceptual quality can be compared.

Since linear prediction and fundamental frequency modelling are described in detail elsewhere, below we will discuss only overall encoder/decoder structure, perceptual evaluation, noise modelling and analysis-by-synthesis.




The source-filter model of CELP codecs.

### 10.3.1 Encoder/decoder structure

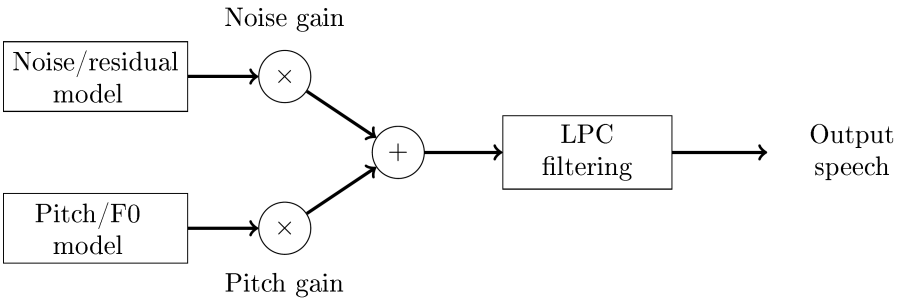
The decoder (see image on the right) very closely implements the idea of the source-filter model (see above). The only refinement are two multiplications with scalar gains, where the noise codebook and pitch contribution are scaled to the desired magnitude. Observe that here we abbreviate linear predictive coding with LPC.

The encoder and decoder typically operate in frames of 20 ms length, which are further subdivided into 5 ms subframes. Operations described above are thus performed on vectors whose length correspond to 5 ms, which at a sampling-rate of 12.8 kHz corresponds to 64 samples.

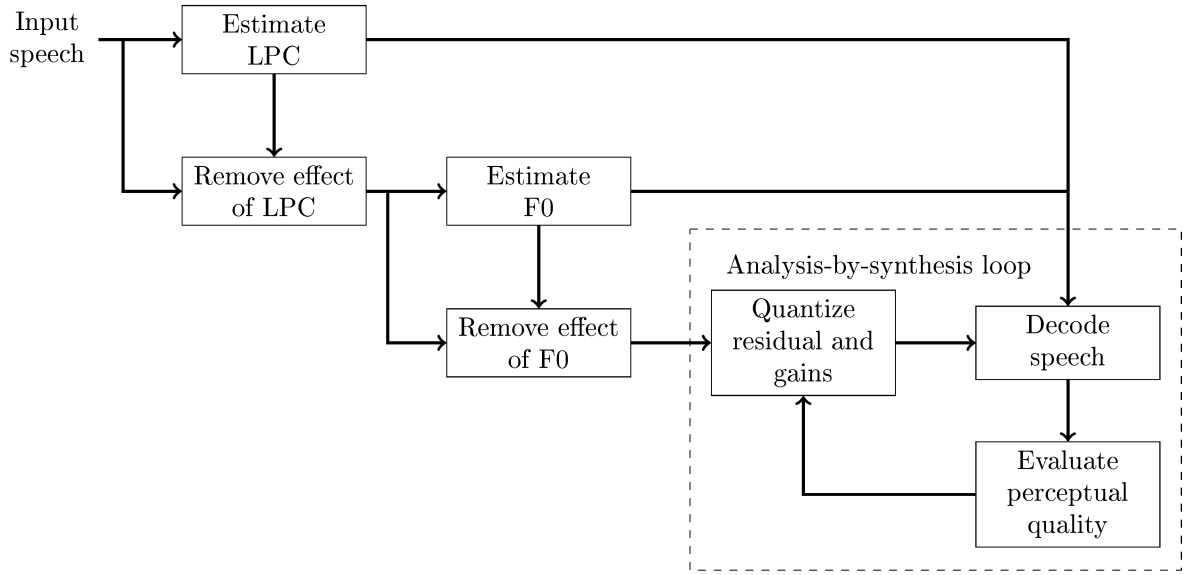
The encoder (see figure on the right) first estimate the linear predictive (LPC) model, then removes its effect from the input. In other words, since linear prediction is IIR-filtering, we can remove the effect from the speech signal with the corresponding FIR-filter to obtain the LPC-residual. We can then similarly estimate the fundamental frequency (F0) from the residual and again remove its effect to obtain the F0-residual.

The F0-residual closely resembles white noise (following the Laplacian distribution). We can thus quantize it with a noise-quantizer (described below) as well as the pitch and noise gains. To evaluate the output quality of the signal, we then decode the quantized signal and calculate a perceptually weighted error. Since LPC-filtering is autoregressive (IIR), it however has a non-linear effect on the output such that quantization has a non-linear effect on the output. We therefore cannot know which quantization is the best one without trying out *all of them*. To get best possible performance, in theory, we should try every possible quantization! However, in practice, we choose a group of potentially-good quantization and find the best out of them. This is known as the *analysis-by-synthesis* method.

Analysis-by-synthesis is a celebrated method because it enables optimization of CELP and achieves relatively high quality. Since CELP is arguably more efficient than competing frequency-domain codecs, and analysis-by-synthesis enables optimization of CELPs, it is important. However, observe that this is a brute-force method, which has an inherent penalty in computational complexity.




CELP decoder structure




CELP encoder structure

### 10.3.2 Perceptual quality evaluation

Perceptual quality in CELP codecs is evaluated with a weighted norm. Suppose $W$ is a convolution matrix corresponding to the perceptual weighting filter, then the weighted norm between the true and quantized residual signals $x$ and $\hat{x}$ , respectively, is

$$d_W(x, \hat{x}) := \|W(x - \hat{x})\|^2 = (x - \hat{x})^T W^T W (x - \hat{x}).$$

Though this is a quadratic form, whose minimization is simple, notice that we consider quantized vectors, such that the minimization is an integer-valued minimization problem, which does not have an analytic solution.

Further, the quantized signal is the sum of noise and pitch contributions, both multiplied with scaling factors

$$\hat{x} := \gamma_{F0} x_{F0} + \gamma_{noise} x_{noise}.$$

When estimating the F0, we can set the noise contribution to zero, such that we minimize

$$\arg\min_{x_{F0}} d_W(x, \gamma_{F0} x_{F0}) := \arg\min_{x_{F0}} (x - \gamma_{F0} x_{F0})^T W^T W (x - \gamma_{F0} x_{F0}).$$

To compare different pitch contributions, we further need to exclude the gain from the problem, which is achieved by setting the derivative with respect to $\gamma_{F0}$ to zero (left as an exercises), which gives the optimal gain as

$$\gamma^*_{F0} = \frac{x^T W^T W x_{F0}}{x^T_{F0} W^T W x_{F0}}.$$

Substituting back to the original problem, after removing constants, yields

$$x^*_{F0} := \arg\min_{x_{F0}} d_W(x, \gamma^*_{F0} x_{F0}) := \arg\max_{x_{F0}} \frac{\left(x^T W^T W x_{F0}\right)^2}{x^T_{F0} W^T W x_{F0}}.$$

Observe that this equation thus evaluates the weighted correlation between the original signal $x$ and the pitch contribution. In other words, different F0's can be evaluated with this function and the one with the highest correlation is chosen as the F0.

Once the F0 has been chosen, we calculate the optimal gain and subtract it from the original residual signal, $x' := x - \gamma^*_{F0} x^*_{F0}$. This F0-residual is then approximately white noise and can be modelled with the noise codebook. Similarly as above, we assume that the noise-gain is optimal such that the noise codebook can be optimized with

$$\gamma^*_{noise} = \frac{x^T W^T W x_{noise}}{x^T_{noise} W^T W x_{noise}}$$

and

$$x^*_{noise} := \arg\min_{x_{noise}} d_W(x', \gamma^*_{noise} x_{noise}) := \arg\max_{x_{noise}} \frac{\left(x^T W^T W x_{noise}\right)^2}{x^T_{noise} W^T W x_{noise}}.$$

We have thus quantized the pitch and noise contributions, but for the two gains we have optimal values, but not optimal *quantized* values. Again, since quantized values are not continuous, we do not have an analytic solution but must search for the best quantization among all possible values. The optimization problem is

$$\arg\min_{\gamma_{F0},\gamma_{noise}} d_W(x, \gamma_{F0} x^*_{F0} + \gamma_{noise} x^*_{noise}) = \arg\min_{\gamma_{F0},\gamma_{noise}} (x - \gamma_{F0} x^*_{F0} - \gamma_{noise} x^*_{noise})^T W^T W (x - \gamma_{F0} x^*_{F0} - \gamma_{noise} x^*_{noise}).$$

We note that the above equation is a polynomial of the two scalar gains and all vector and matrix terms reduce to constants, such that

$$\arg\min_{\gamma_{F0},\gamma_{noise}} d_W(x, \gamma_{F0} x^*_{F0} + \gamma_{noise} x^*_{noise}) = c_0 + \gamma_{F0} c_1 + \gamma^2_{F0} c_2 + \gamma_{noise} c_3 + \gamma_{noise}\gamma_{F0} c_4 + \gamma^2_{noise} c_5.$$

In difference to the optimization of the residual vectors, this optimization is computationally relatively simple such that we can exhaustively search for the best gains. The gains are usually quantized with 8 to 10 bits, such that this involves only 256 to 1024 polynomial evaluations.The final quantized residual is then

$$\hat{x}^* = \gamma^*_{F0} x^*_{F0} + \gamma^*_{noise} x^*_{noise}.$$

### 10.3.3 Noise modelling and algebraic coding

As mentioned above, the residual after LPC filtering and F0 modelling is approximately stationary white noise, that is, it is constant variance and samples are uncorrelated. We would like to quantize this effectively. White noise signals have however no structure left, except their probability distribution. We can assume that the residual samples $\xi_k$ follow the Laplacian distribution with zero mean,

$$f(\xi_k) = C \exp\left(-\frac{\|\xi_k\|}{s}\right).$$

The joint log-likelihood is

$$\log \prod_k f(\xi_k) = C' - \sum_k \frac{\|\xi_k\|}{s} = C' - \frac{1}{s}\|x_{noise}\|_1,$$

where $x_{noise} := [\xi_1, \ldots, \xi_K]$. and $\|x\|_1$ is the 1-norm (absolute sum). In other words, if we model constant-probability vectors $x_{noise}$, then that is equivalent with modelling vectors with a constant 1-norm, $\|x_{noise}\|_1 =$ constant. We can thus

build a codebook which has constant 1-norm. For example, if we quantize to integer values, then the absolute sum of the quantized signal is a fixed integer.

In the simplest case, we can quantize $x_{noise}$ to have one signed pulse at location $k$, and otherwise all samples are zero. The location of the pulse can be encoded with $\log_2 K$ bits, and the sign with one bit, such that the overall bit-consumption is $1 + \log_2 K$. This encoding strategy can be readily extended by adding more pulses. The bit-consumption of multi-pulse vectors however becomes more complicated. The issue is that if apply a naive encoding where we directly encode the position and sign of each pulse, then we use more bits than necessary for two reasons. Firstly, if two pulses overlap, then they must have the same sign otherwise they would cancel. Secondly, pulses are indistinguishable, such that their ordering does not matter, such that if we encode pulses one by one, changing their order would give different bit-streams but the encoded signal would remain the same. Both imply that we are using too many bits when using such encodings for multiple pulses. Solutions exist for optimal encoding of such multi-pulse vectors, but the algorithm becomes involved.

In any case, the outcome is that it is possible to generate quantizations of residual signals with algorithmic methods. That is, we have an algorithm or algebraic rule which defines all possible quantizations and consequently, such quantization is known as algebraic coding. The encoding can be chosen to have optimal bit-consumption for a given number of pulses and it is thus (with loose assumptions) the best possible quantization for the residual vector when using a fixed bitrate. It is computationally efficient since the residual vectors are mostly zeros, such that evaluation of the optimization function is straightforward to calculate. It also efficient in the sense that codebooks do not have to be stored, but can be generated on the fly by an algorithm.

Algebraic coding is so central to CELP codecs that CELP codecs using algebraic coding are known as algebraic CELP or ACELP. Most main stream codecs, such as AMR, EVS and USAC use ACELP. Some codecs use also other residual codebooks, but even then, algebraic codes are always the first choice.

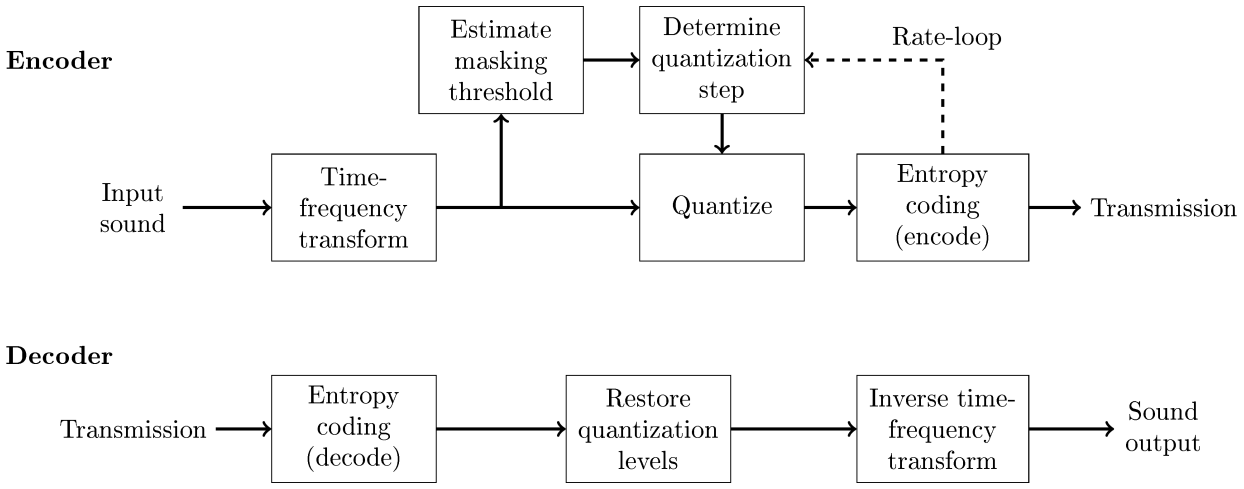
## 10.4 Frequency-domain coding

In audio coding, the classical approach is based on coding in the frequency domain, which means that we are coding a time-frequency representation of the signal. Such coding methods are especially suitable for signals which have prolonged stationary parts, such as many instrument-sounds, which are often stationary for the duration of a note, more or less. Frequency-domain codecs are based on *entropy coding* where the quantization accuracy is chosen with a *perceptual model*.

Classical speech coding is however based on code-excited linear prediction (CELP), which is a fundamentally different paradigm. In modern (mobile) applications, these two modalities, speech and audio are often intertwined and we would like to be able to encode all types of sounds. For example, a movie can have music, speech, speech with music in the background, music with spoken and sung parts, and there are frequent and seamless transitions between such modalities. For a unified codec, which can encode both speech, music, generic audio and their mixtures, we therefore needs codecs which support also frequency-domain coding. Besides, stationary parts of speech such as fricatives can sometimes be more efficiently encoded with frequency-domain coding anyway.

Such codecs, which encode both speech, music and generic audio are often collectively called *speech and audio codecs*. Most recent codecs such as MPEG USAC, 3GPP EVS and Opus are speech and audio codecs. Internally, they contain elements of both CELP and frequency-domain coding and they switch between these modes depending on the content.

More specifically, frequency-domain codecs are based on a time-frequency transform such as the MDCT, *perceptual modelling* to choose the quantization accuracy, and *entropy coding* to transmit the quantized signal with the least amount of bits. At the decoder, the process is reversed (as is obvious). In many designs, the bitrate is variable, such that with the perceptual model we choose the desired perceptual accuracy, and entropy coder compresses that as much as possible. Then we always have approximately the desired quality, but we however do not in advance know how many bits we will use. In other designs, we strive for fixed bitrate, such that we should reach the highest quality as long as the bit-consumption remains under a chosen limit. With most entropy codecs this means that we have to implement a *rate-loop*, where we iteratively search for the best quantization accuracy which remains within the chosen limit on bit-consumption.


Encoder
Estimate masking threshold
Determine quantization step
Rate-loop
Input sound
Time-frequency transform
Quantize
Entropy coding (encode)
Transmission
Decoder
Transmission
Entropy coding (decode)
Restore quantization levels
Inverse time-frequency transform
Sound output

CHAPTER

# ELEVEN

# SPEECH ENHANCEMENT

When using speech technology in real environments, we are often faced with less than perfect signal quality. For example, if you make a phone call at cafeteria, typically you have plenty of other people speaking in the background, there could be music playing and the room itself can have reverberation. Such effects distort the desired speech signal such that the receiving end, the desired speech sounds less pleasant, requires more effort to understand or at the worst case, it becomes less intelligible. *Speech enhancement* refers to methods which try to reduce such distortions, to make speech sounds more pleasant, reduce listening effort and improve intelligibility.

The most prominent categories of speech enhancement are:

1. *Noise attenuation*, where we try to extract the desired speech signalm when distorted by background noise(s).
2. *Echo cancellation* and feedback cancellation are used when the sound played from a loudspeaker is picked up by a microphone distorting the desired signal.
3. *Bandwidth extensions* refers to methods which convert a signal at a lower sampling rate to a higher sampling rate and fills the missing range of the spectrum with some plausible content.
4. Dereverberation refers to methods which attenuate the effect of room acoustics on the desired signal.
5. Source separation methods try to extract sounds of single sources from a mixture, for example, in the classical cocktail-party problem, we would like to isolate single speakers when multiple people are talking at the same time.
6. *Beamforming* refers to spatially selective methods, where the objective is isolate sounds coming from a particular direction, by using the information about the spatial separation of a set of microphones.

The objective of speech enhancement however requires a bit more consideration. In its most classical form, the objective is to extract a clean speech signal from a distorted mixture, where the distortions can be background and sensor noises, as well as room reverberation. Here the clean reference signal is considered to be that signal which would be rerecorded with a microphone close to the speaker, which does not contain said noises or reverberation. It is then clear that it will be challenging to obtain realistic data, since even a microphone close to the speaker will usually contain background noises and effect of reverberation. For development of methods, it is therefore often difficult to obtain data which would accurately correspond to a realistic situation. In any case, a typical objective would be to improve the signal to noise ratio (with or without perceptual weighting) as much as possible.

A more challenging scenario is when two or more persons are speaking in the same acoustic environment. The second speaker can then be viewed as a competing speaker (undesired source) or as a discussion partner (desired source). Even if the two speakers are in an interaction with each other, then often they will speak on top of each other, even if stereotypically we think of a dialogue as a non-overlapping back and forth exchange of non-overlapping arguments. If we want to separate between the two speakers, then overlaps are difficult, because the statistics of the both speech signals will be rather similar, whereas noise signals with distinct statistics are easier to attenuate.

Sometimes we do not want to remove all distortions entirely, but just attenuate their effect. Completely removing artefacts can sometimes make the signal sound unnatural and besides removing distortions, processing methods also almost always distorts the desired signal. Therefore, to retain a natural-sounding signal and to minimize distortion of the desired speech signal, we often limit the extent to which distortions are removed.

A further aspect of enhancement is intelligibility and pleasantness; as a starting point, observe that the speech of some people is by nature difficult to understand or otherwise just annoying (unpleasant). It then conceivable that we device some processing which improves the speech signal to better than the original. What "sounds better" is however a difficult concept, since we do not have unambiguous measures for "how good it sounds" and opinions between listeners will certainly diverge.

Intelligibility with regard to human listeners is similarly complicated as pleasantness, but luckily, we can use speech recognition engines to obtain objective measures. That is, if we give noisy and improved speech signals to a speech recognizer, we can determine the recognition performance in both cases to estimate the benefit obtained with our processing.

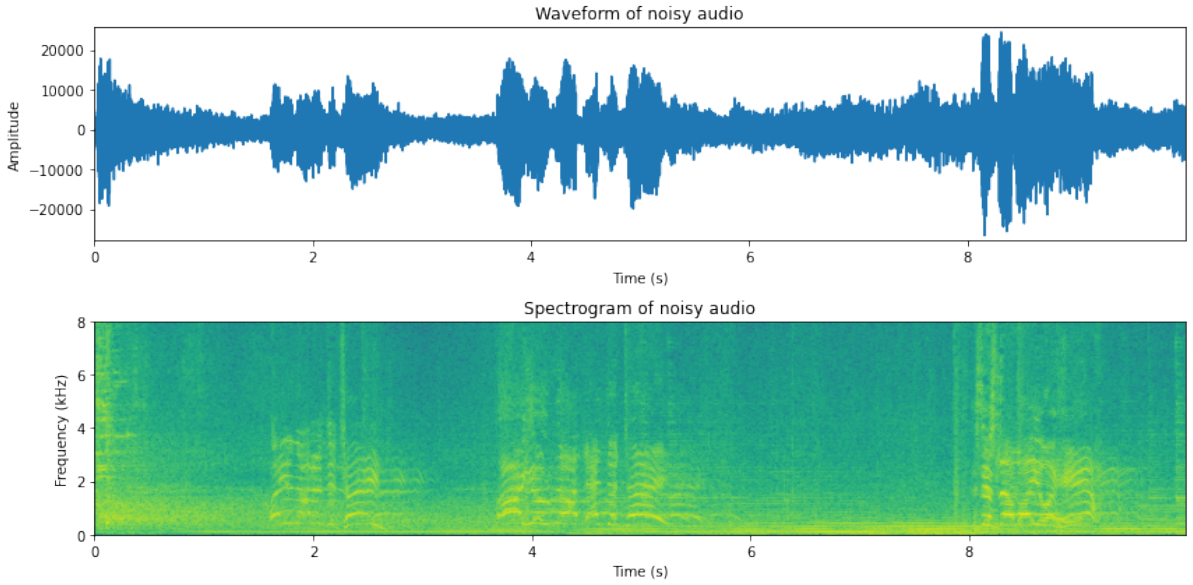
Photo by Jezael Melgoza on Unsplash

## 11.1 Noise attenuation

When using speech technology in realistic environments, such as at home, office or in a car, there will invariably be also other sounds present and not only the speech sounds of desired speaker. There will be the background hum of computers and air conditioning, cars honking, other speakers, and so on. Such sounds reduces the quality of the desired signal, making it more strenuous to listen, more difficult to understand or at the worst case, it might render the speech signal unintelligible. A common feature of these sounds is however that they are *independent* of and *uncorrelated* with the desired signal. [Benesty *et al.*, 2008]

That is, we can usually assume that such noises are *additive*, such that the observed signal $y$ is the sum of the desired signal $x$ and interfering noises $v$, that is, $y = x + v$. To improve the quality of the observed signal, we would like to make an estimate $\hat{x}$ of the desired signal $x$. The estimate should approximate the desired signal $x \approx \hat{x}$ or conversely, we would like to minimize the distance $d(x, \hat{x})$ with some distance measure $d(\cdot, \cdot)$.

```
<IPython.lib.display.Audio object>
```




### 11.1.1 Noise gate

Suppose you are talking in a reasonably quiet environement. For example, typically when you speak on a phone, you would go to a quiet room. Similarly, when attending an on-line lecture, you would most often want to be in a room without background noise.

What we perceive as quiet is however never entirely silent. When we play a sound recorded in a "quiet" room, then in the reproduction you then hear the *local* and the *recorded* background noises. Assuming the two noises have similar energies, then their sum has twice the energy, viz. 6dB higher than the original noises. In a teleconference with multiple participants, the background noises add up such that each contributes with an 6dB increase in the background noise level. You do not need many participants before the total noise level becomes so high that communication is impossible.

The **mute**-button in teleconferences is therefore essential. Participants can silence their microphones whenever they are not speaking, such that only the background noise of the *active* speaker(s) is transmitted to all listeners.

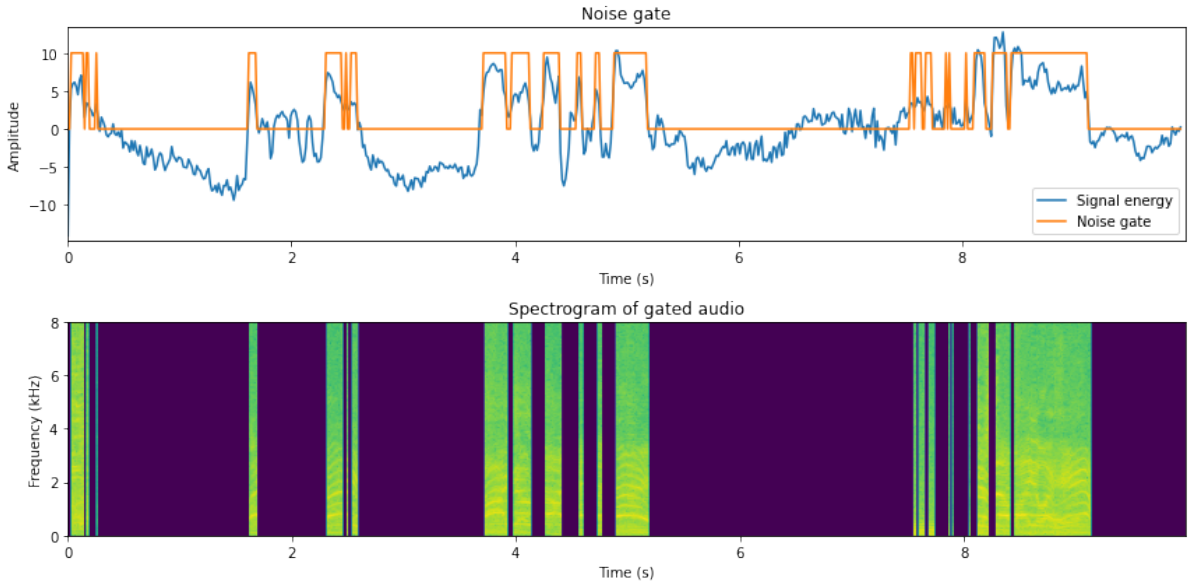
While the mute-button is a good user-interface in the sense that it gives control to the user, it is however an annoying user-interface in that users tend to forget to mute and unmute themselves. Would be better with an automatic mute.

**Noise gating** is a simple auto-mute in the sense that it thresholds signal energy and turns reproduction/transmission off if energy is too low. Typically it also features a hysterisis functionality such that reproduction/transmission is kept off for a while after the last speech segment. Moreover, to avoid discontinuities, there is a fade-in/fade-out functionality at the start and end.

Note that noise gating with a energy threshold is a simple implementation of a *voice activity detector (VAD)*. With more advanced features than mere energy we can refine voice activity detects quite a bit, to make them more robust especially in noisy and reverberant environments. In addition to enhancement of signal quality, such methods are often used also to preserve resources, such as transmission bandwidth (telecommunication) and computation costs (recognition applications such as speech recognition).

For the noise-gate, we first need to choose a threshold. Typically that the threshold is chosen relative to the mean (log) energy $\sigma^2$ such that the threshold is $x^2 < \sigma^2\gamma$, where $\gamma$ is a tunable parameter. Moreover, we can implement the gate such that if we are below the threshold, we set a gain value to 0 and otherwise to 1. If we want fade-in/fade-out, we can ramp that gain value smoothly from 0 to 1 at the attack and from 1 to 0 at the release.

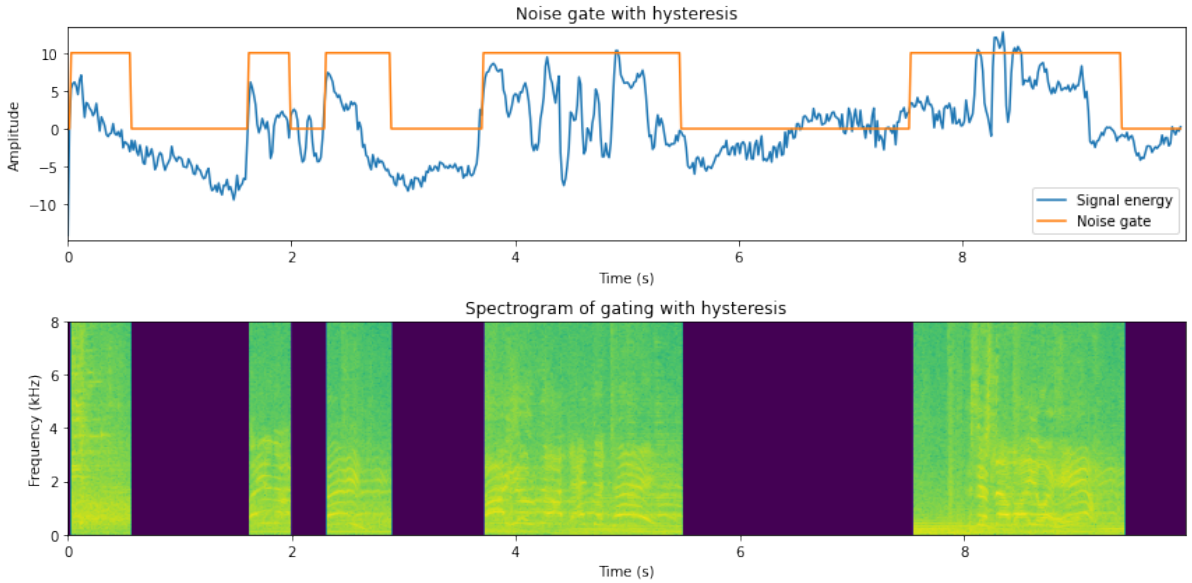
```
frame_energy = np.sum(np.abs(spectrogram_matrix)**2,axis=1)
frame_energy_dB = 10*np.log10(frame_energy)
mean_energy_dB = np.mean(frame_energy_dB) # mean of energy in dB

threshold_dB = mean_energy_dB + 3. # threshold relative to mean

speech_active = frame_energy_dB > threshold_dB
```




```
<IPython.lib.display.Audio object>
```

This is quite awful isn't it? Though we did loose many stationary segments of noise, we also distorted the speech signal significantly. In particular, typically we loose plosives at the beginning of words. Overall the sound also sounds odd when it turns on an off again.

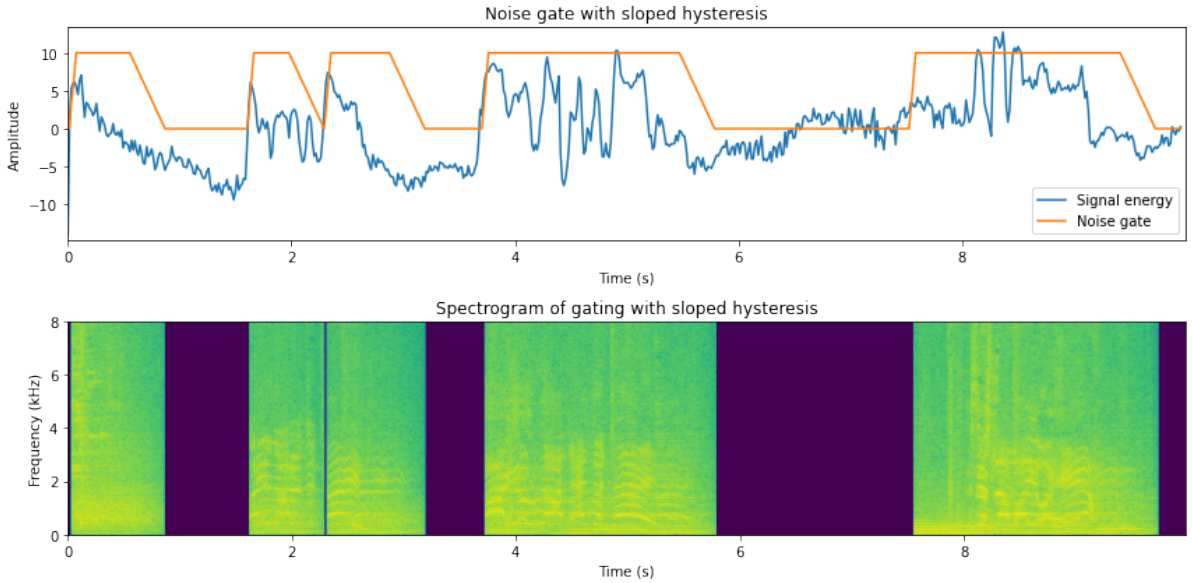
```
hysteresis_time_ms = 300
hysteresis_time = int(hysteresis_time_ms/window_step_ms)

speech_active_hysteresis = np.zeros([window_count])
for window_ix in range(window_count):
    range_start = max(0,window_ix-hysteresis_time)
    speech_active_hysteresis[window_ix] = np.max(speech_active[range(range_start,
↪window_ix+1)])
```




```
<IPython.lib.display.Audio object>
```

This sounds quite a bit better already. There are only some sudden muted areas (depending on your sound sample), but overall the sound is clearly better.

```
# Fade-in and fade-out
fade_in_time_ms = 50
fade_out_time_ms = 300
fade_in_time = int(fade_in_time_ms/window_step_ms)
fade_out_time = int(fade_out_time_ms/window_step_ms)

speech_active_sloped = np.zeros([window_count])
for frame_ix in range(window_count):
    if speech_active_hysteresis[frame_ix]:
        range_start = max(0,frame_ix-fade_in_time)
        speech_active_sloped[frame_ix] = np.mean(speech_active_hysteresis[range(range_
↪start,frame_ix+1)])
    else:
        range_start = max(0,frame_ix-fade_out_time)
        speech_active_sloped[frame_ix] = np.mean(speech_active_hysteresis[range(range_
↪start,frame_ix+1)])
```




```
<IPython.lib.display.Audio object>
```

This doesn't sound all too bad! The sudden on- and off-sets are gone and the transitions to muted areas sound reasonably natural.

Now we have implemented gating for the full-band signal. Gating can be easily improved by band-wise -processing. Depending on the amount of processing you can afford, you could go all the way and applying gating on individual frequency-bins in the STFT.

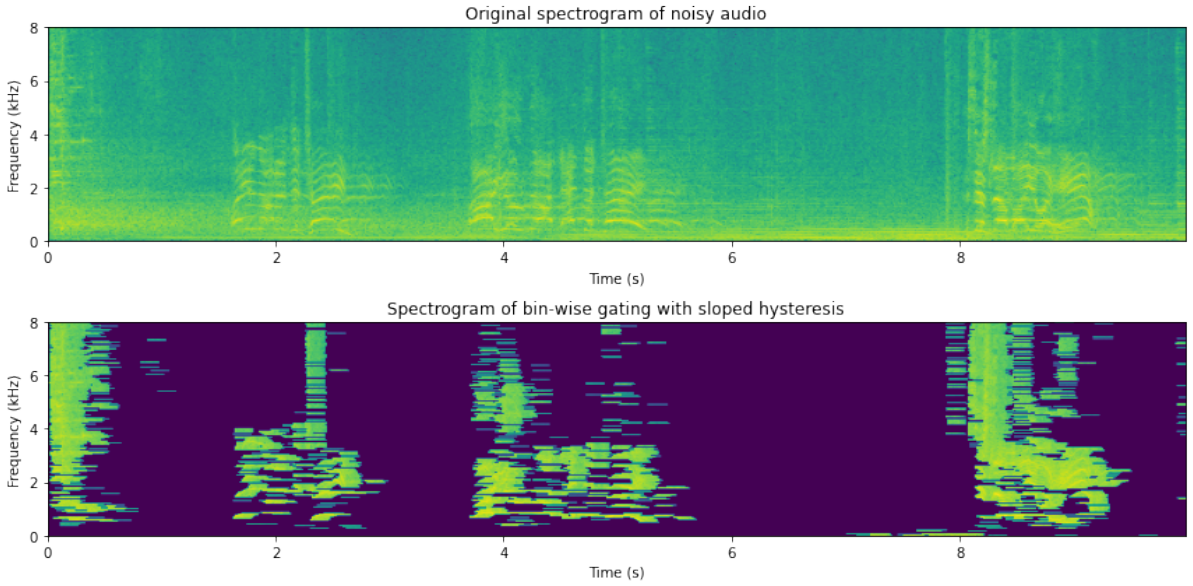
```
hysteresis_time_ms = 100
hysteresis_time = int(hysteresis_time_ms/window_step_ms)

fade_in_time_ms = 30
fade_out_time_ms = 60
fade_in_time = int(fade_in_time_ms/window_step_ms)
fade_out_time = int(fade_out_time_ms/window_step_ms)


# NB: This is a pedagogic, but very slow implementation since it involves multiple␣
 ↪for-loops.
spectrogram_binwise = np.zeros(spectrogram_matrix.shape,dtype=complex)
for bin_ix in range(fft_length):
    bin_energy_dB = 10.*np.log10(np.abs(spectrogram_matrix[:,bin_ix])**2)
    mean_energy_dB = np.mean(bin_energy_dB) # mean of energy in dB
    threshold_dB = mean_energy_dB + 16. # threshold relative to mean
    speech_active = bin_energy_dB > threshold_dB

    speech_active_hysteresis = np.zeros_like(speech_active)
    for window_ix in range(window_count):
        range_start = max(0,window_ix-hysteresis_time)
        speech_active_hysteresis[window_ix] = np.max(speech_active[range(range_start,
 ↪window_ix+1)])

    #speech_active_sloped = np.zeros_like(spe
    for frame_ix in range(window_count):
        if speech_active_hysteresis[frame_ix]:
```

(continues on next page)

(continued from previous page)

```
            range_start = max(0,frame_ix-fade_in_time)
            speech_active_sloped[frame_ix] = np.mean(speech_active_
↪hysteresis[range(range_start,frame_ix+1)])
        else:
            range_start = max(0,frame_ix-fade_out_time)
            speech_active_sloped[frame_ix] = np.mean(speech_active_
↪hysteresis[range(range_start,frame_ix+1)])

    spectrogram_binwise[:,bin_ix] = spectrogram_matrix[:,bin_ix]*speech_active_sloped
```

```
<IPython.lib.display.Audio object>
```




```
<IPython.lib.display.Audio object>
```

This is clearly again a step better, but you should note two things:

- The parameters are now quite a bit different; hysteresis and ramps shorter as well as higher threshold. This was required to get acceptable quality.
- Some musical noise left in the low and high frequencies, where isolated areas are turned on for a while.

If possible, try listening to this sound with headphones and from loudspeakers. Is there a difference? Which one sounds better?

A possible impression is that, with good-quality headphones, the result is too clean. It sounds unnatural. With loudspeakers, however, it may sound quite ok. When listening to headphones, you better perceive the absence of background noise, whereas on the loudspeaker, there is more background noise present locally, such that the absence of reproduced background noise is not noticeable. With loudspeakers, any reproduced background noise would also interact with the local room, generating a double room reverberation (assuming that the reproduced loudspeaker sound already had reverberation).

In any case, it seems therefore clear that muting the background noise entirely is not always desirable (at least when listening with headphones to a single speaker with background noise). We should therefore apply some more intelligent gain factor (see more in the spectral subtraction section below).

### 11.1.2 Statistical model

The *STFT spectrum* of a signal is a good domain for noise attenuation because we can reasonably safely assume that spectral components are uncorrelated with each other, such that we treat each component separately. In other words, in the spectrum, we can apply noise attenuation on every frequency bin with scalar operations, whereas if the components would be correlated, we would have to use vector and matrix operations. The benefit of scalar operations is that they are computationally simple, $O(N)$, whereas matrix operations are typically at least $O(N^2)$.

The sources are, according to our assumption, uncorrelated, which means that the expected correlation is zero,

$$E[xv] = 0.$$

A consequence of this assumption is that, for the mixture $y = x + v$, we have the energy expectation

$$E\left[y^2\right] = E\left[x + v^2\right] = E\left[x^2\right] + E\left[v^2\right] + 2E\left[xv\right] = E\left[x^2\right] + E\left[v^2\right].$$

In other words, in terms of expectations, the energy is the sum of component energies, which will become a very practical property.

To find the energy of the speech signal, we then just need to estimate $E\left[v^2\right]$.

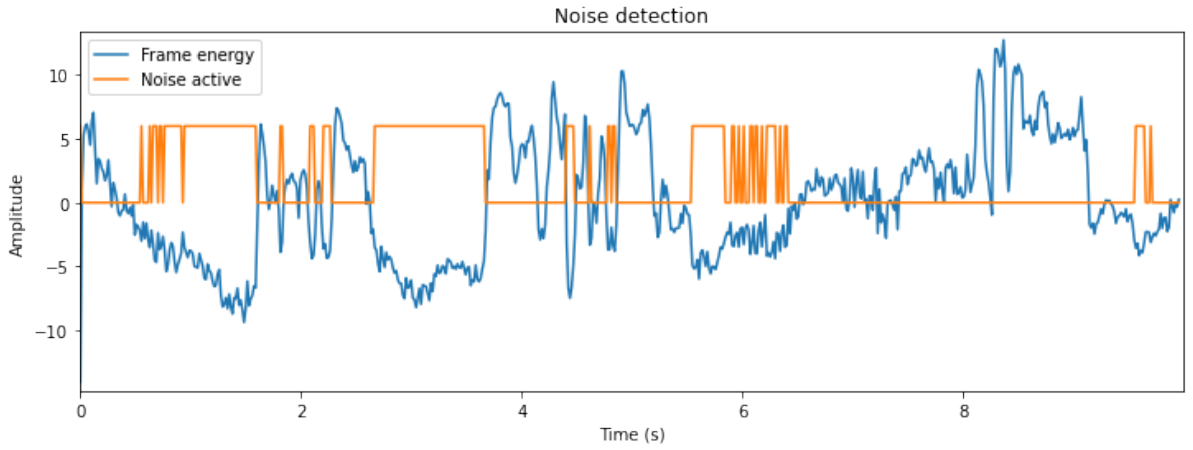
#### Noise estimation and modelling

#### Mean-energy with voice activity detection

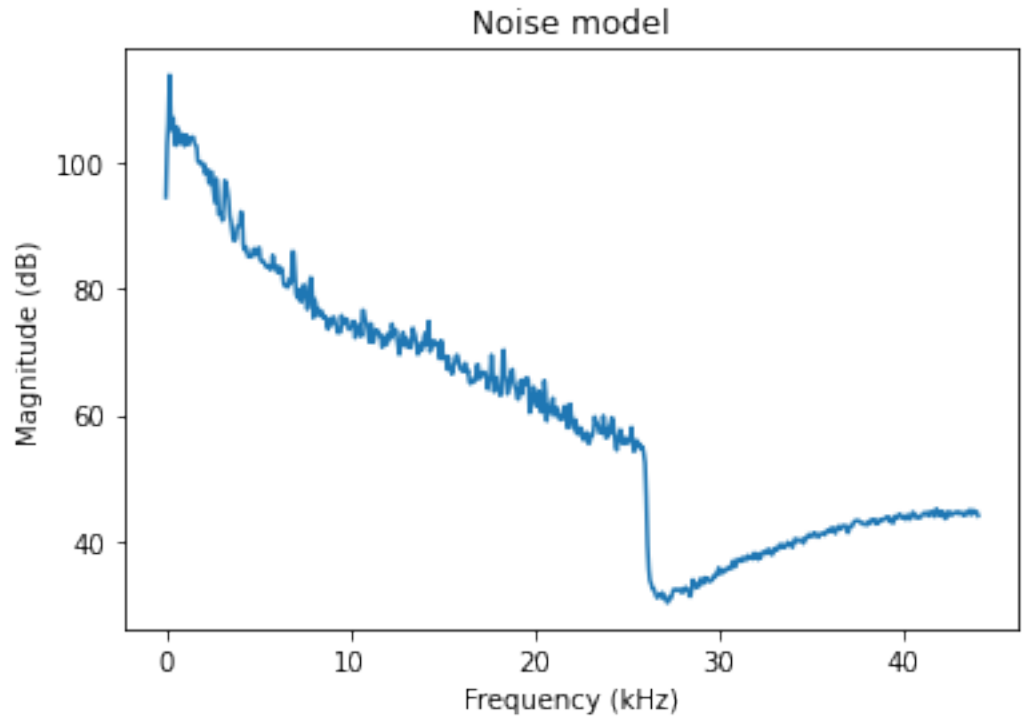
To be able to remove noise, we first need to estimate noise characteristics or statistics. We thus need to find sections of the signal which have noisy only (non-speech). One approach would be to use *voice activity detection* (VAD) to find non-speech segments. Assuming we have a good VAD this can be effective. It works if we assume that noise is stationary, such that the noise on non-speech parts are similar to the noise in speech-parts. In general, VADs are accurate only at low noise levels.

```
<IPython.core.display.SVG object>
```

```
# VAD through energy thresholding
frame_energy = np.sum(np.abs(spectrogram_matrix)**2,axis=1)
frame_energy_dB = 10*np.log10(frame_energy)
mean_energy_dB = np.mean(frame_energy_dB) # mean of energy in dB

noise_threshold_dB = mean_energy_dB - 3. # threshold relative to mean

noise_active = frame_energy_dB < noise_threshold_dB
```




```
# Estimate noise in noise frames
noise_frames = spectrogram_matrix[noise_active,:]
noise_estimate = np.mean(np.abs(noise_frames)**2,axis=0)
```

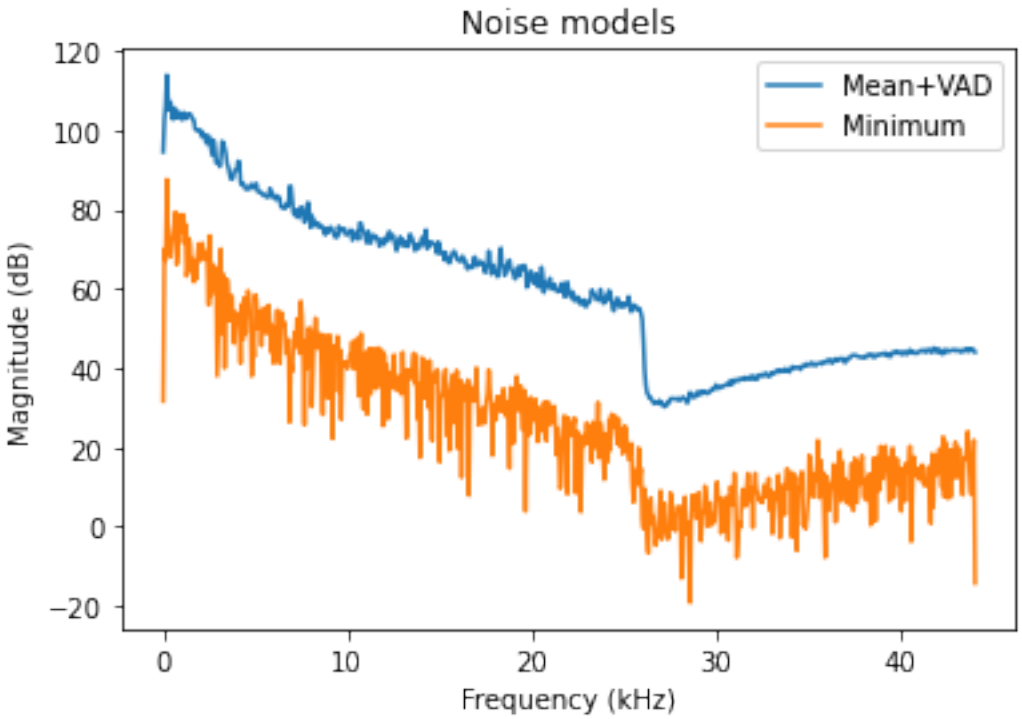



In typical office environments and many other real-world scenarios, the background noise is dominated by low-frequencies, that is, the low frequencies have high energy. Often low quality hardware also leaks analog distortion into the microphone, such that there can be visible peaks in the higher frequencies.

### Minimum statistics

An alternative estimate would be the *minimum-statistics* estimate, where we take the minimum energy of each component over time. Since speech+noise has more energy than just noise alone, the minimum most likely represents only noise. [Martin, 2001]

```
noise_estimate_minimum = np.min(np.abs(spectrogram_matrix)**2,axis=0)
```




We see that the minimum statistics estimate of noise is much lower than the mean. Duh, it is by definition lower. However, the shape of both spectra is the same. Thus, the minimum statistics estimate is biased, but preserves the shape. The bias is something we can correct by adding a fixed constant. The benefit of minimum statistics is that it does not depend on a threshold. We thus replace the threshold parameter by a bias parameter, which is much less prone to errors.

In the following, we will use the mean as our noise model for simplicity.

## 11.1.3 Spectral subtraction

The basic idea of spectral subtraction is that we assume that we have access to an estimate of the noise energy $E[\|v\|^2] = \sigma_v^2$ , and we subtract that from the energy of the observation, such that we define the energy of our estimate as [Boll, 1979]

$$\|\hat{x}\|^2 := \|y\|^2 - \sigma_v^2.$$

Unfortunately, since our estimate of noise energy is not perfect and because we have hiddenly made an inaccurate assumption that $x$ and $v$ are uncorrelated, the above formula can give negative values for the energy estimate. Negative energy is not realizable and nobody likes pessimists, so we have to modify the formula to threshold at zero

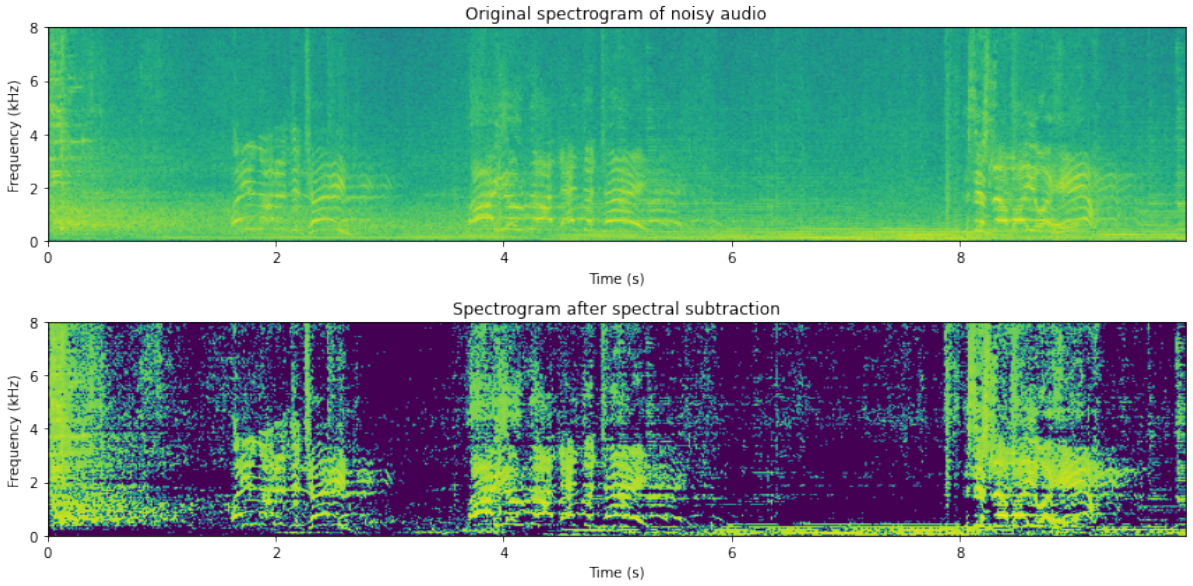
$$\|\hat{x}\|^2 := \begin{cases} \|y\|^2 - \sigma_v^2 & \text{if } \|y\|^2 \geq \sigma_v^2 \\ 0 & \text{if } \|y\|^2 < \sigma_v^2 \end{cases}.$$

Since STFT spectra are complex-valued, we then still have to find the complex angle of the signal estimate. If the noise energy is small $\|v\|^2 \ll \|x\|^2$ , then the complex angle of $x$ is approximately equal to the angle of $y$, $\angle x \approx \angle y$ , such that our final estimate is (when $\|y\|^2 \geq \sigma_v^2$ )

$$\hat{x} := \angle y \cdot \|\hat{x}\| = \frac{y}{\|y\|}\sqrt{\|y\|^2 - \sigma_v^2} = y\sqrt{\frac{\|y\|^2 - \sigma_v^2}{\|y\|^2}}.$$

```
energy_enhanced = np.subtract(np.abs(spectrogram_matrix)**2, np.expand_dims(noise_
↪estimate,axis=0))
energy_enhanced *= (energy_enhanced > 0)  # threshold at zero
enhancement_gain = np.sqrt(energy_enhanced/(np.abs(spectrogram_matrix)**2))
spectrogram_enhanced = spectrogram_matrix*enhancement_gain;
```

```
<IPython.lib.display.Audio object>
```




```
<IPython.lib.display.Audio object>
```

Typically spectral subtraction does improve the quality of the signal, but it also leaves room for improvement. In particular, usually we hear musical noise in the higher frequencies, which are components which alternate between on and off. Because these components appear independetly from the speech signal, they are perceived as independent sounds and because they are essentially isolated sinusoids, they are perceived as tones; thus we hear musical noisy "tones".

### 11.1.4 Minimum-mean Square Estimate (MMSE) aka. Wiener filtering

Observe that the form of the relationship above is $\hat{x} = y \cdot g$, where $g$ is a scalar scaling coefficient. Instead of the above heuristic, we could then derive a formula which gives the smallest error, for example in the minimum error energy expectation sense or minimum mean square error (MMSE). Specifically, the error energy expectation is

$$E\left[\|e\|^2\right] = E\left[\|x - \hat{x}\|^2\right] = E\left[\|x - gy\|^2\right] = E\left[\|x\|^2\right] + g^2 E\left[\|y\|^2\right] - 2gE\left[xy\right].$$

If we assume that target speech and noise are uncorrelated, $E\left[xv\right] = 0$ ,

then $E\left[xy\right] = E\left[x(x+v)\right] = E\left[\|x\|^2\right]$ and

$$E\left[\|e\|^2\right] = E\left[\|x\|^2\right] + g^2 E\left[\|y\|^2\right] - 2gE\left[\|x\|^2\right] = (1-2g)E\left[\|x\|^2\right] + g^2 E\left[\|y\|^2\right].$$

The minimum is found by setting the derivative to zero

$$0 = \frac{\partial}{\partial g} E\left[\|e\|^2\right] = -2E\left[\|x\|^2\right] + 2gE\left[\|y\|^2\right],$$

such that the final solution is

$$g = \frac{E\left[\|x\|^2\right]}{E\left[\|y\|^2\right]} = \frac{E\left[\|y\|^2\right] - \sigma_v^2}{E\left[\|y\|^2\right]}.$$

and the Wiener estimate becomes

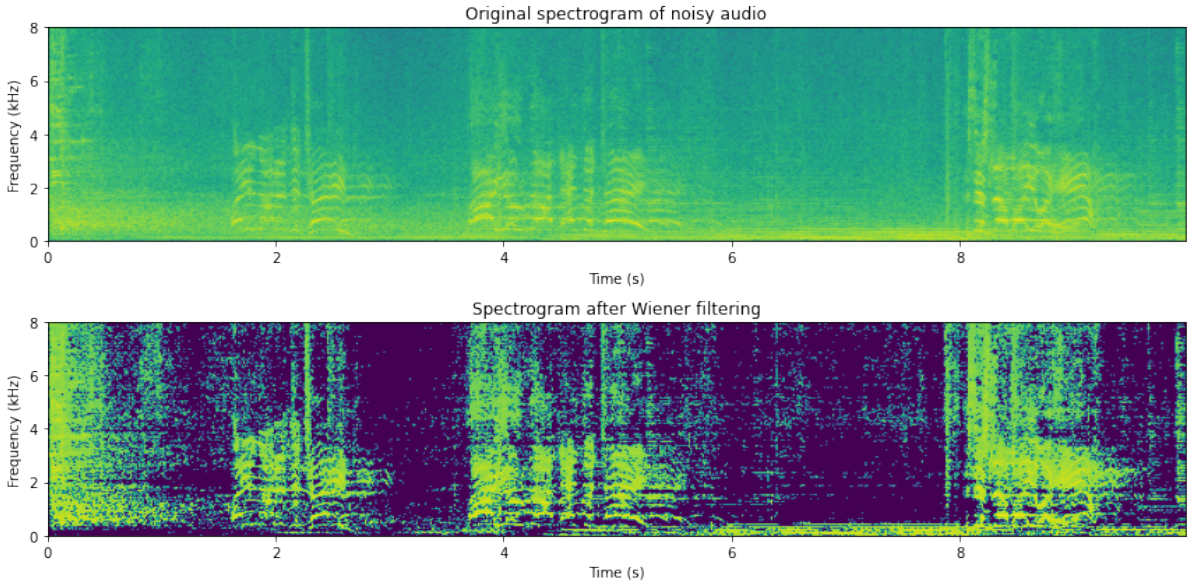
$$\hat{x} := y\left(\frac{\|y\|^2 - \sigma_v^2}{\|y\|^2}\right).$$

Observe that this estimate is almost equal to the above, but the square root is omitted. With different optimization criteria, we can easily derive further such estimates. Such estimates have different weaknesses and strengths and it is then a matter of application specific tuning to choose the best estimate.

Overall, it is however somewhat unsatisfactory that *additive* noise is attenuated with a *multiplicative* method. However, without a better model of source and noise characteristics, this is probably the best we can do. Still, spectral subtraction is a powerful method when taking into account how simple it is. With minimal assumptions we obtain a signal estimate which can give a clear improvement in quality.

```
mmse_gain = energy_enhanced/(np.abs(spectrogram_matrix)**2)
spectrogram_enhanced_mmse = spectrogram_matrix*mmse_gain
```

```
<IPython.lib.display.Audio object>
```




```
<IPython.lib.display.Audio object>
```

The quality of the MMSE estimate is not drastically different from the spectral subtraction algorithm. This is not surprising as the algorithms are very similar.

In comparison to noise gating, we also notice that it may not be entirely clear which one is better. To a large extent, this will happen with good quality signals, where the SNR is high. With worse SNR, energy thresholding becomes more difficult and noise gating will surely fail. MMSE is more robust and therefore typically has better quality at low SNR. These methods can also be combined to give the best of both worlds.

### Treating musical noise

For spectral subtraction -type methods (which includes MMSE), we often encounter musical noise (described above). Similar problems are common in coding applications which use frequency-domain quantization. The problem is related to discontinuity in spectral components over time. Musical noise is perceived when spectral components are randomly turned on and off.

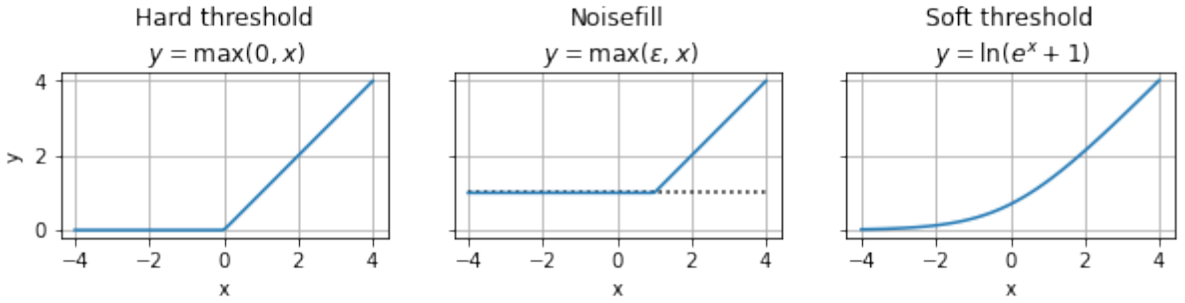
The solution to musical noise must thus be to avoid components going on and off. Possible approaches include:

### Noise filling

Spectral holes (zeros) can be avoided by a noise floor, where we add random noise any time spectral components are below a threshold.

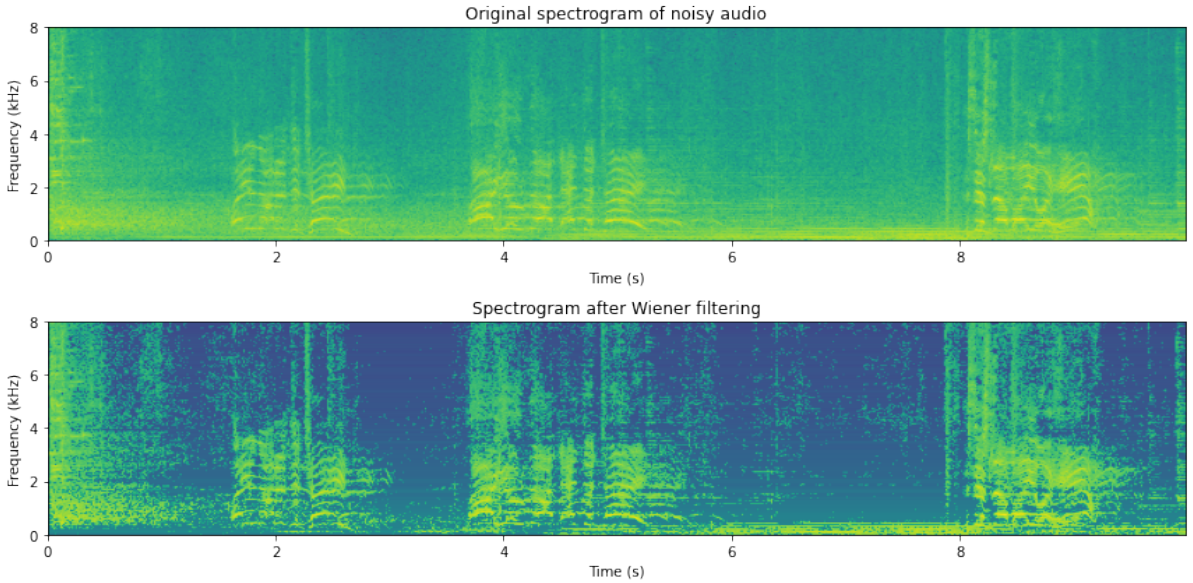
### Mapping

In spectral subtraction -type methods, the gain value was thresholded at zero before multiplication by the spectrum. It is this thresholding which causes the problem. Therefore, we could replace the hard threshold with a soft threshold. One such soft threshold is the soft-plus, defined for an input $x$ as $y = \ln(e^x + 1)$.




```
# Noise fill with min(eps,x)
noisefill_threshold_dB = -60    # dBs below average noise
noisefill_level = (10**(noisefill_threshold_dB/10))*noise_estimate
#noisefill_level = 0.2

energy_enhanced = np.abs(spectrogram_matrix)**2 - noise_estimate

#energy_noisefill = noisefill_level + 0.5*(energy_enhanced - noisefill_level + np.
 ↪abs(energy_enhanced - noisefill_level))
energy_noisefill = noisefill_level + ((energy_enhanced - noisefill_level) > 0) *␣
 ↪(energy_enhanced - noisefill_level)
mmse_noisefill_gain = np.sqrt(energy_noisefill)/np.abs(spectrogram_matrix)
spectrogram_enhanced_noisefill = spectrogram_matrix*mmse_noisefill_gain;
```

```
<IPython.lib.display.Audio object>
```




```
<IPython.lib.display.Audio object>
```

We find that noisefilling -type methods offer a compromise between amount of noise removed and the amount of musical noise. Usually we try to tune the parameter such that we are just above the level where musical noise is perceived, since it is a non-natural distortion and therefore more annoying than an incomplete noise attenuation. However, closer tuning is dependent on application and preference.

### Wiener filtering for vectors

Above we considered the scalar case, or conversely, the case where we can treat components of a vector to be independent such that they can be equivalently treated as a collection of scalars. In some cases, however, we might be unable to find an uncorrelated representation of the signal or the corresponding whitening process could be unfeasibly complex or it can incur too much algorithmic delay. We then have to take into account the correlation between components.

Consider for example a desired signal $x \in \mathbb{R}^{N\times 1}$ , a noise signal $v \in \mathbb{R}^{N\times 1}$ and their additive sum, the observation $y = x + v$, from which we want to estimate the desired signal with a linear filter $\hat{x} := a^H y$. Following the MMSE derivation above, we set the derivative of the error energy expectation to zero

$$\begin{aligned} 0 &= \frac{\partial}{\partial a} E\left[\|e\|^2\right] = \frac{\partial}{\partial a} E\left[\|x-\hat{x}\|^2\right] = \frac{\partial}{\partial a} E\left[\|x-a^H y\|^2\right] \\ &= \frac{\partial}{\partial a} E\left[\|x-a^H(x+v)\|^2\right] = 2E\left[(x+v)\left(x-a^H(x+v)\right)^H\right] \\ &= 2\left[E[xx^H] - \left(E[xx^H]+E[vv^H]\right)a\right] = 2\left[R_{xx} - \left(R_{xx}+R_{vv}\right)a\right] \end{aligned}$$

Where the covariance matrices are $R_{xx} = E[xx^H]$ and $R_{vv} = E[vv^H]$ , and we used the fact that $x$ and $v$ are uncorrelated $E[xv^H] = 0$ . The solution is then clearly

$$a = \left(R_{xx}+R_{vv}\right)^{-1} R_{xx} = R_{yy}^{-1}\left(R_{yy}-R_{vv}\right),$$

where we for now assume that the inverse exists. This solution is clearly similar to the MMSE solution for the scalar case.

A central weakness of this approach is that it involves a matrix inversion, which is computationally complex operation, such that on-line application is challenging. It furthermore requires that the covariance matrix $R_{yy}$ is invertible (positive definite), which places constraints on the methods for estimating such covariances.

In any case, Wiener filtering is a convenient method, because it provides an analytical expression for an optimal solution in noise attenuation. It consequently has very well documented properties and performance guarantees.

### 11.1.5 Masks, power spectra and temporal characteristics

As seen above, we can attenuate noise if we have a good estimate of the noise energy. However, actually, both the spectral subtraction and Wiener filtering methods use models of the speech and noise energies. The models used above were rather heuristic; noise energy was assumed to be "known" and speech energy was defined as energy of observation minus noise energy. It is however not very difficult to make better models than that. Before going to improved models, note that we did not use speech and noise energies independently, but only their ratio. Now clearly the ratio of speech and noise is the signal-to-noise ratio (SNR) of that particular component. We thus obtain an estimate of the SNR of the whole spectrum. Conversely, we would need only the SNR of the spectrum to attenuate noise with the above methods. The SNR as a function of frequency and time is often referred to as a *mask* and in the following we will discuss some methods for generating such masks. It is however important to understand that mask-based methods are operating on the power (or magnitude) spectrum and thus do not include any models of the complex phase. Indeed, efforts have in general focused mostly on the power spectrum and much less on the phase. On one hand, the motivation is that characteristics of the power spectrum are much more important to perception than the phase (though the phase is also important), but on the other hand, the power spectrum is also much easier to work with than the phase. Therefore there has been both much more demand and supply of methods which treat the power spectrum.

To model speech signals, we can begin by looking at spectral envelopes, the high-level structure of the power spectrum. It is well-known that the phonetic information of speech lies primarily in the shape of the spectral envelope, and the lowest-frequency peaks of the envelope identify the vowels uniquely. In other words, the distribution of spectral energy varies smoothly across the frequencies. This is something we can model and use to estimate the spectral mask. Similarly, we know that phonemes vary slowly over time, which means that the envelope varies slowly over time. Thus, again, we can model envelope behaviour over time to generate spectral masks.

A variation of such masks is known as *binary* masks, where we can set, for example, that the mask is 1 if speech energy is larger than noise energy, and 0 otherwise. Clearly this is equivalent with thresholding the SNR at unity (which is equivalent to 0 dB), such that an SNR-mask can always be converted to a binary mask, but the reverse is not possible. The benefit of binary masks is that it simplifies some computations.

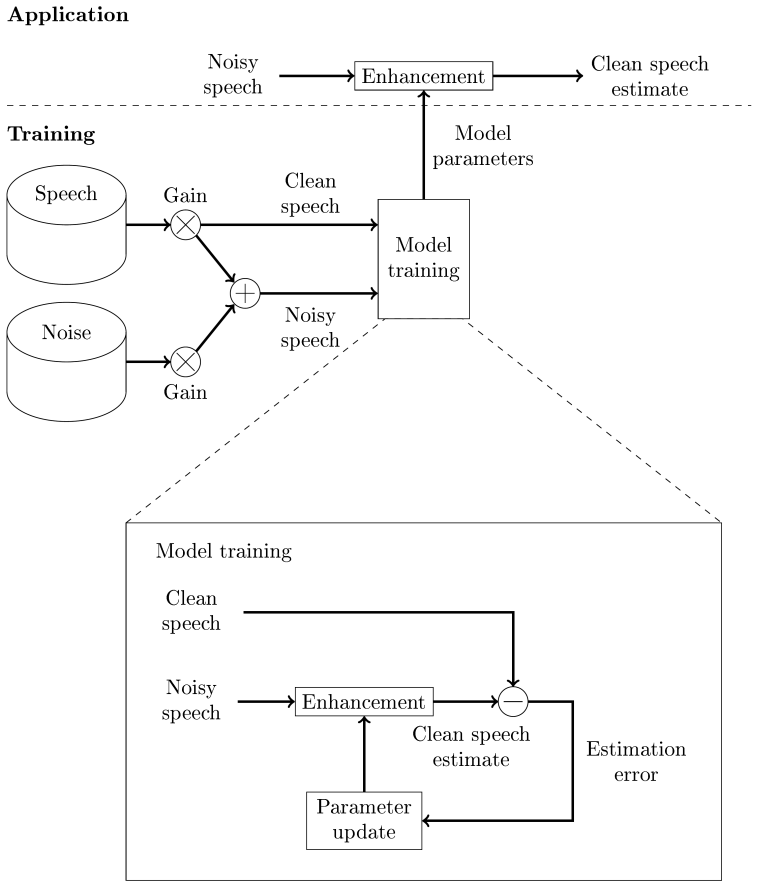
If we then want to attenuate noise in a particular frame of speech it is then useful to use as much of the surrounding data (context) as possible. For best quality, we can model, say, a second of the speech signal both before and after the target frame. Though this can improve quality of the estimate, this has two clear negative consequences. First of all, including more data increases computational complexity. The level of acceptable complexity is though highly dependent on the application. Secondly, if we look into *future* frames to process the current frame, then we have to have access to such data. In a on-line system, this means that we have to wait for the whole look-ahead data to arrive before processing, such that the overall system has a delay corresponding to the length of the look-ahead. We can extrapolate the current frame from past frames, but interpolating between past and future frames does give much better quality. The amount of acceptable delay is also an application dependent question.

### 11.1.6 Machine learning methods

The first choice in designing machine learning methods for noise attenuation and other speech enhancement tasks is the overall systems architecture. The application is usually simply a neural network which takes noisy speech as input and outputs an estimate of the clean speech. A natural choice would then be to train the network with a large database of noisy speech samples and minimize the distance of the output to the clean speech signal. Since we assume that noise is additive, we can create synthetic samples by adding noise to speech. By varying the intensity (volume) of the noise samples, we can further choose the signal to noise ratio of the noise samples. With reasonable size databases of speech and noise, we thus get a practically infinite number of unique noisy samples such that we can make even a large neural network to converge.

A weakness of this model however is that even if the database is thus large, it has only a limited number of unique noises and unique speakers. There is no easy way of getting assurance that unseen noises and speakers would be enhanced effectively. What if we receive a noisy sample where a 3 year-old child talks with annoying vuvuzelas playing in the background. If our database contained only adult speakers and did not contain vuvuzela-sounds, then we cannot know whether our enhancement is effective on the noisy sample.

Machine learning configuration for speech enhancement with noisy and target clean speech signal.

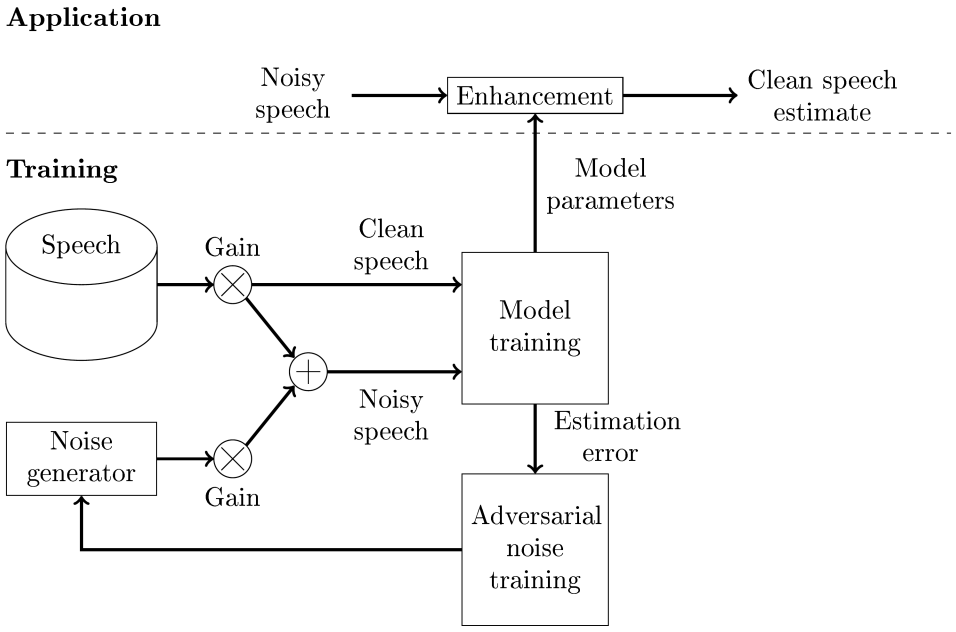



To overcome the problem of inadequate noise databases, we can take an *adversarial* approach, where we have a *generative* network which generates noises and an enhancement network which attenuates noises which corrupt speech. This approach is known as a *generative adversarial network (GAN)*. We then have two optimization tasks;

1. To optimize the enhancement network (minimize estimation error) to remove the noise generated by the generative network and
2. to optimize the generative network (maximize estimation error) to generate noises which the enhancement network is unable to remove.

These two tasks are adversial in the sense that they work against each other. In practical application we would use only the enhancement network, so the generative network is used only in training.

Application and training with a generative adversarial network (GAN) structure for speech enhancement.




### 11.1.7 References

## 11.2 Echo cancellation

A frequently occurring distortion in speech telecommunication scenarios is echoes, which have either an electric or acoustic cause. Electric echoes appear in analogue networks at points of impedance mismatch. Since a majority of telecommunication networks today are digital, such electric echoes are mostly of historical interest and not discussed here further. Acoustic echoes however is a problem which has become more important, especially with the increasing use of teleconferencing services. Such acoustic echoes appear when the speech of a person A is played through the loudspeakers for a person B, such that the loudspeaker sound is picked up by the microphone of person B and transmitted back to person A. If the delay would be mere milliseconds, then person A would perceive the feedback signal as reverberation, which is *not* too disturbing (in some circumstances such an effect, known as *side-talk*, could actually be a useful feature indicating that the microphone is recording). However, typically the transmission delay is above 100ms, such that the feedback is perceived as an echo, which is usually highly disconcerting and disturbing. Even worse, sometimes the acoustic echo signal completes the feedback loop and goes in a circle, again and again. If the feedback signal is attenuated on each loop, then the feedback slowly diminishes, but sometimes the feedback loop can amplify the signal such that the signal quickly escalates up to the physical limit of hardware or until the loudspeakers blow up. It is thus clear that echoes must be avoided in real-life systems.

Fortunately, *echo cancellation* is well-understood problem with "standard" solutions available. In short, these methods are based on estimating the acoustic room-impulse-response (RIR), filtering the loudspeaker signal with the RIR to obtain an estimate of the acoustic echo, and subtracting the estimated echo from the microphone signal. Though it is a classic problem with off-the-shelf solutions available, it remains a difficult problem. Central difficulties include:

- The RIR is not stationary and must therefore be estimated on-line as an *adaptive filter*. The RIR changes whenever a door or window is opened or closed, furniture are moved or people move within the room, and even when the temperature of the room changes. Changes are even more rapid when using mobile, handheld or wearable devices, when the location of the device can change rapidly. In the worst case, the microphone and loudspeaker can be on different devices such that their acoustic distance can change rapidly.
- When the RIR has a long tail, that is, when the room has a long reverberation, then the corresponding filters must be long, which increases *computational complexity*. Fast adaptation to changing RIRs also increases computational requirements.

Observe that echo *cancellation* methods refer to *subtracting* the estimated echo from the microphone signal. In contrast, *noise attenuation* methods *multiply* the microphone signal with a positive scalar, such that the output signal approximates the echo-free signal. The essential difference is that multiplicative methods generally estimate only the energy/magnitude of the signal, while subtractive methods try to match both magnitude and phase. The benefit of subtractive methods is that the output quality is generally better since the removal-method matches the physical process which creates the signal. Multiplicative methods however are much more robust than subtractive methods; in particular, if the phase is incorrectly estimated, then a subtractive method can *increase* the amount of noise in a signal. In the worst case, an echo canceller poorly matched to the room response can generate a catastrophic feedback loop, whereas multiplicative methods can be designed to never increase the amount of noise.

*Echo suppression* methods use this insight to remove acoustic echo with a multiplicative method similar to that of spectral subtraction or Wiener filtering used in noise attenuation.

### 11.2.1 Echo cancellation solutions

As it is mentioned above, the problem that echo cancellation needs to solve is the effect of the room in the path from a loudspeaker to the microphone used in the communication. This means that the signal played by the loudspeaker enters the system with a certain delay and multiple echo paths (Room impulse response). From now on, we will refer to the received signal played by the loudspeaker as far-end (*x(n)*), and the useful speech signal from the user will be called near-end signal (*s(n)*). The mixture recorded by the microphone that would be sent to the other end of the call can be represented as:

$$d(n) = s(n) + h(n) * x(n) + v(n)$$

where *v(n)* represents additive noise in the scene and, for simplicity, will be considered part of *s(n)*. The effect of the room on the far-end signal can be modelled as an FIR filter *h(n)*, added to the far-end signal using a convolution. The idea of echo cancellation is to find an estimation of this FIR filter, and applying it to the received far-end signal, we can then subtract it from *d(n)* to extract the useful near-end signal.

The estimation of the echo path model is done by *minimizing* the *mean square error* (MMSE) between the recorded signal and the estimated filtered far-end, assuming that s(n) is not present. However, only one estimation is not possible due to multiple factors.

1. The echo path is unknown at the beginning of the communication, additionally, any changes in the echo path can be catastrophic in the estimation.
2. The non-stationariety of the speech signal makes the estimation of the echo path a complex task.
3. It is very likely that *s(n)* is present in the recorded signal, causing the estimation to be flawed (Double-talk).

These three issues require a continuous monitoring of the quality of the estimation and the filter must be periodically updated in order to accurately represent the echo path. For that reason the MMSE method must be calculated iteratively as we receive more information from the far-end and recorded signals. The most popular algorithm to calculate this adaptive filter is Least Mean Squares (LMS) and its multiple variants. The expression to minimise is:

$$MMSE = min\|d(n) - \hat{h}(n) * x(n)\|^2$$

The objective is to find the coefficients of the estimated filter that minimise the previous equation, and a simple depiction of the iterative LMS algorithm used to update the filter can be divided in three steps:

- Estimate the recorded echo signal:

$$\hat{y}(n) = \sum_i \hat{h}(n-i) * x(n-i)$$

- Estimate the error:

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

- Update the filter weights:

$$\hat{h}_{t+1}(n) = \hat{h}_t(n) + \mu \cdot e^H(n) \cdot x(n)$$

The term $\mu$ represents the learning rate of the algorithm. Higher values will make the algorithm converge faster, but might converge to a bigger error value, while smaller learning rate will converge slower and it might not be able to follow the changes in the echo path. It has been proven that an adaptive learning rate provides the best results and most modifications on this method, focus on handling an adaptive learning rate according to the required specifications. As we can see in the second equation, the estimation error of the adaptation algorithm is also the signal that will be used as output and that will contain the corresponding *s(n)* in an optimal case.

This first approach of the LMS adaptive filter is proposed to be applied on the time domain of the audio signal. If the filter updates on every sample of the received audio signal, accurately modelling a room impulse response will require a filter of a few thousand samples. Updating the filter on every new sample becomes computationally expensive, and for these reasons other methods are proposed based on this approach:

- BlockLMS: Reduce the rate of updates in the algorithm, such that the update is only applied once every certain number of samples can help considerably reduce the complexity of the algorithm. However, the block processing can affect the convergence of the algorithm and reduce its responsiveness.
- Frequency Domain Adaptive Filters (FDAF): The simplest case of FDAF consists in converting the audio signal to the frequency domain using an STFT, and then apply an independent LMS filter on each of the frequency components of the signal. This alllows to represent the echo path with a much smaller amount of samples. Considering the algorithm with quadratic complexity, it is preferable to have many short filters than having just a long one.

Finally, as we mention above, the main problem that adaptive filters face in real-life communication applicatoins is double talk. In a common interaction between two people, it is very likely that both speakers are active at the same time. In the echo cancellation framework, that means that both far-end signal *x(n)* and near-end *s(n)* will be present in the mixture. As the adaptive filter tries to minimize the error between the far-end signal and the recorded one, in the presence of double-talk the filter will likely diverge and remove or distort the near-end signal instead of reducing the echo feedback.

To reduce the effect of double-talk in the adaptation process, it is important to detect when double-talk starts and stop the adaptation. Assuming that the filter had converged before the double-talk segment, the the far-end signal should still be removed, while the near-end speaker would pass through. Many methods have been presented to control the learning rate of the adaptative filter depending on the detected double-talk, but most of them are based on three main ideas:

- Energy based detection (Geigel detector):
  - This detector assumes that, if the energy ratio between the far-end signal and the recorded one will remain almnost constant until an additional voice is added from the near-end signal. When the ratio between far-end and recorded signals changes, we can assume that there is double-talk.
  - It is a very simple method that barely adds any computational complexity to the system.
  - The detector assumes that the echo level is clearly different than the near-end speech. Therefore, this method is highley influenced by noise and signal misalignment.
  - The method also requires to be tuned for each specific configuration and changes in the echo path during communication might trigger the double-talk detection.
- Normalized Cross-Correlation (NCC):
  - This method measures the similarity between the input and processed signals.
  - It is more robust to noise than the energy levels.
  - The value of NCC will be close to 1 when double-talk is present, and 0 when it is not. Therefore, the NCC can be used as a scaling factor for the learning rate.

$$NCC = \frac{\sigma_{ed}^2(n)}{\sigma_e(n)\sigma_d(n)}$$

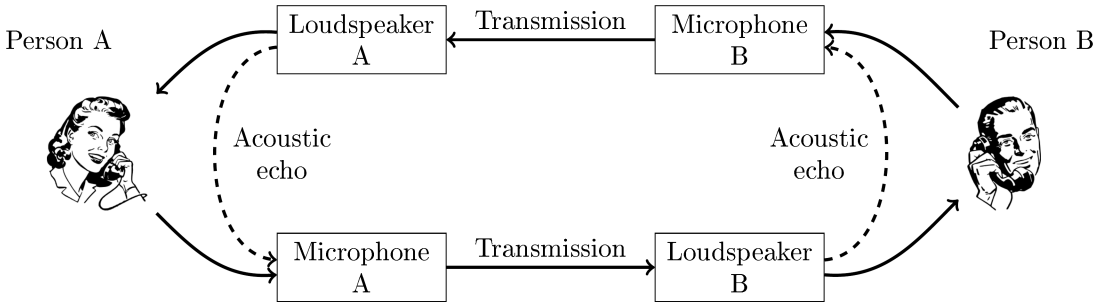
$$\sigma_x^2 = E[xx^H]$$

$$\sigma_x^2(n) = \lambda\sigma_x^2(n-1) + (1-\lambda)\sigma_x^2$$

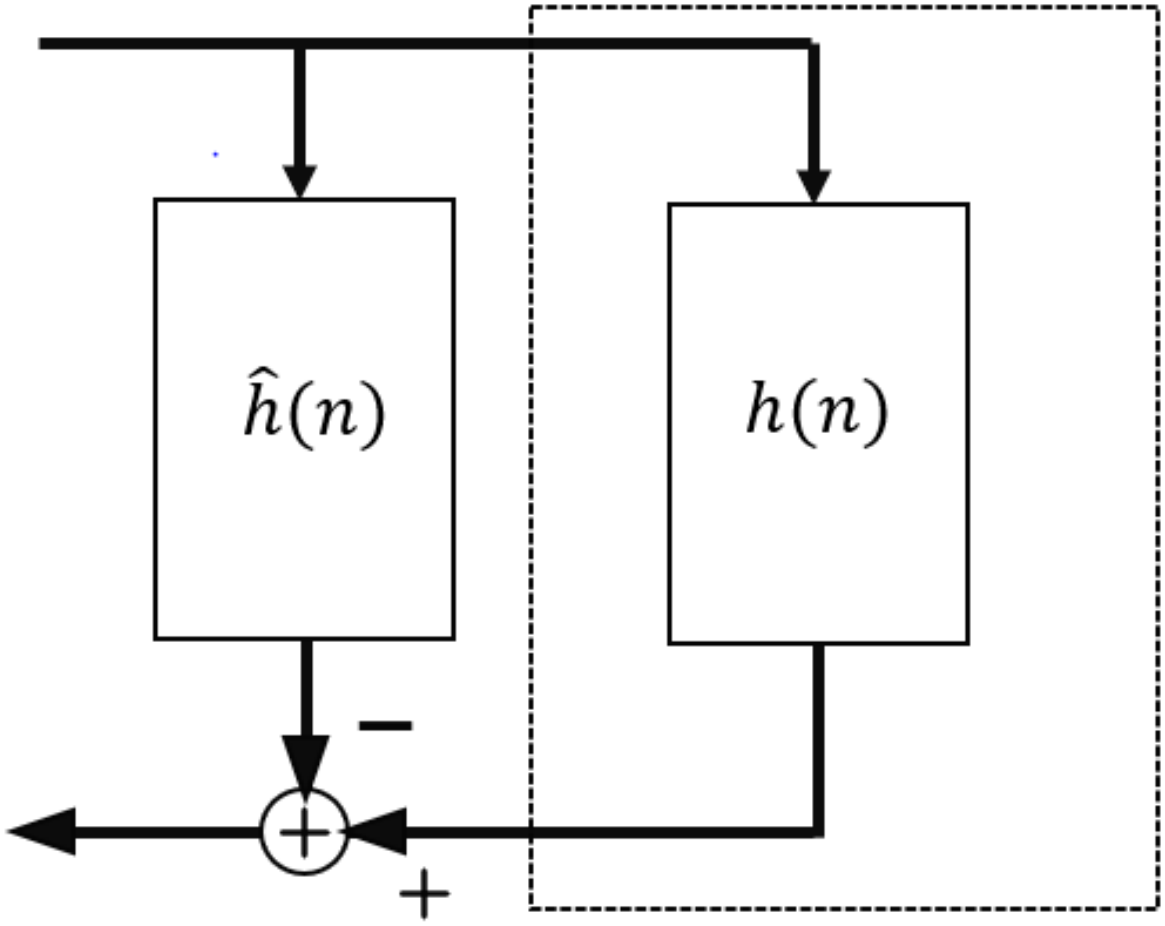
$$\mu_{NCC} = (1 - NCC)\mu_{max}$$

- Two-path Echo Cancellation:
  - Two filters, background and foreground, process the echo signal simultaneously. The background filter is continuously adapting on every step, while the foreground one remains fixed. A control module then decides if there is double-talk and the better solution is to update the coefficients or keep them unmodified based on the output of the backround and foreground filters.
  - It is the most robust of the presented methods.
  - Requires additional computational complexity, as the far-end signal needs to be filtered twice before deciding if ther is double-talk.

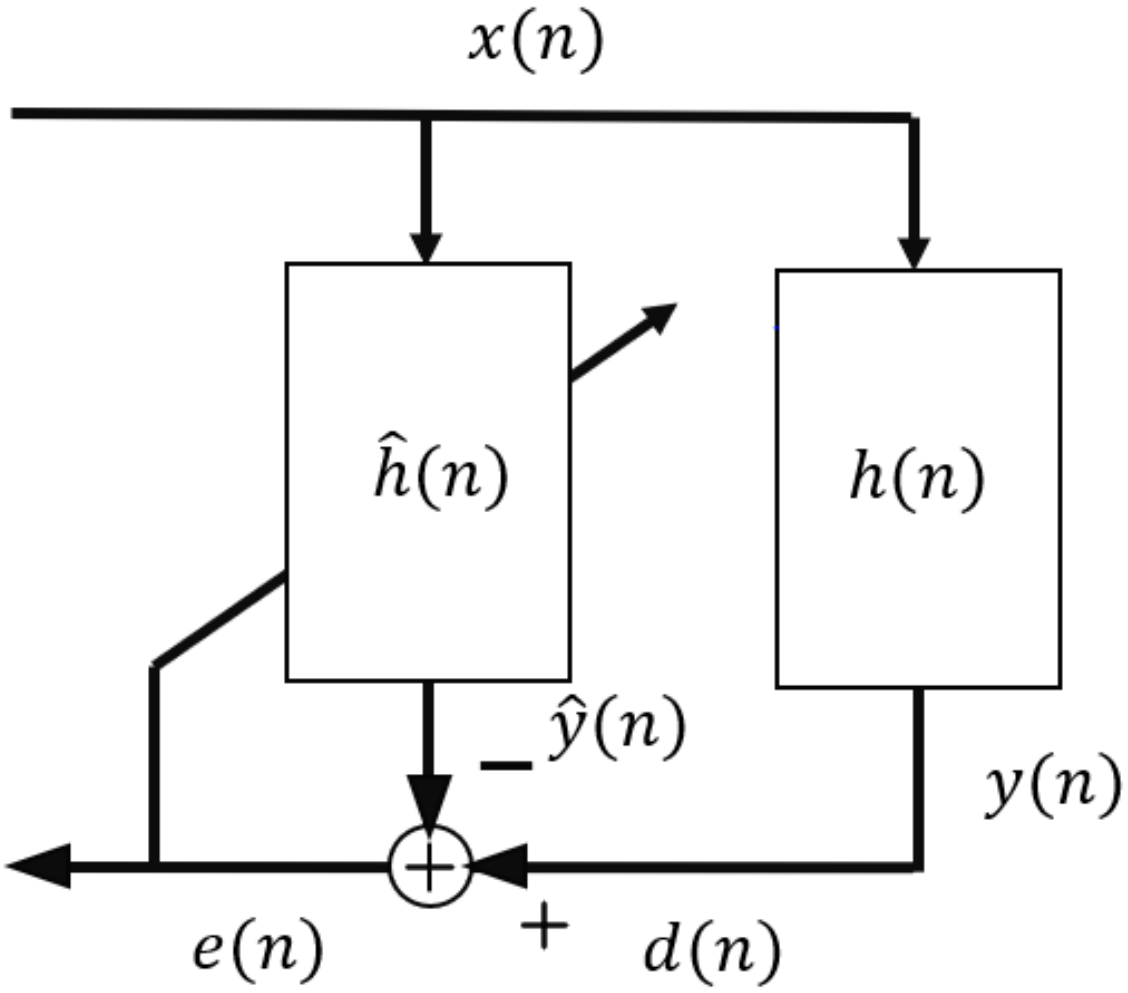
The acoustic feedback loop in telecommunication applications.



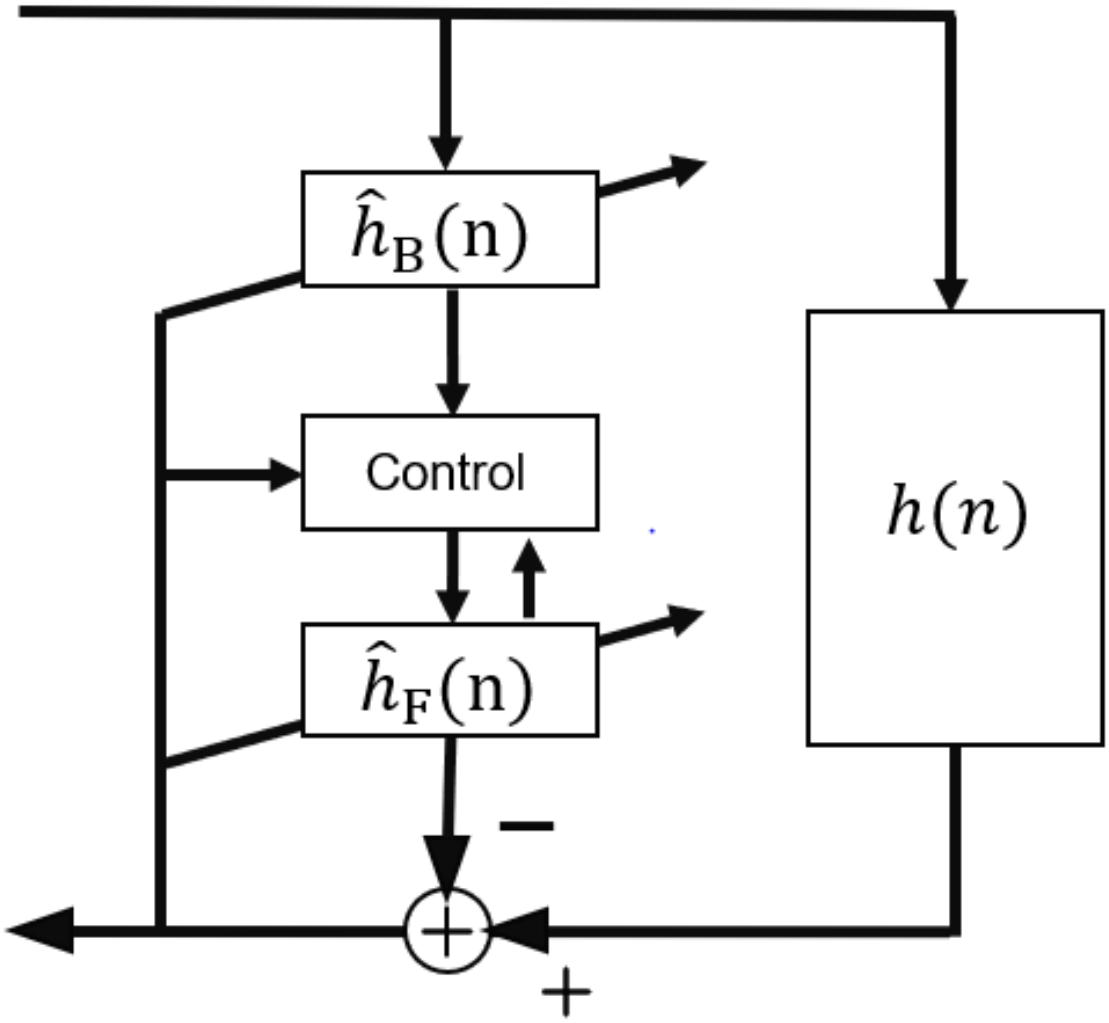

Model of the echo path and estimated filter




Feedback loop for echo cancellation




Two-path echo cancellation model




## 11.3 Bandwidth extension (BWE)

## 11.4 Multi-channel speech enhancement and beamforming

Simple, single-channel *noise attenuation* and dereverberation is often not sufficient for acceptable quality, especially in very noisy environments, such as in a car or on a busy sidewalk. To reach better quality, we can then add more microphones. The benefit of added microphones includes at least:

- Time-of-arrival differences between sources enables the use of beamforming filters, which use such time (phase) differences to separate between sources.
- Intensity level differences between sources; In for example a mobile phone, we can have forward and backward

facing microphones such that the backward facing microphone is used primarily for estimating background noise, while the forward facing microphone records the desired speech signal. By using the background-noise estimate from the backward facing microphone, we can then use noise attenuation on the forward facing microphone to gain better quality.

- Each microphone will feature some level of sensor-noise, that is, the hardware itself causes some inaccuracies to the signal. Sensor-noises are for most parts independent across microphones such that with each additional microphone we can better separate desired sources from noise.

The most-frequently discussed approach is to use microphone arrays, typically in either a linear configuration, where microphones are equi-spaced on a straight line, or in a circular configuration, where microphones are equi-spaced on a circle. The benefits include that a linear configuration makes analytical analysis easier, whereas a circular array can have an almost uniform response in all directions.

### 11.4.1 Delay-and-sum beamforming

As an introduction to beamforming consider a linear array of $K$ microphones with input signals $x_k(t)$, where $k$ and $t$ are the microphone and time indices. We assume that the desired source is sufficiently far away that we can approximate it with a plane wave. Then the signal will arrive at the microphones at different times $\Delta t_{x_k}$ and we can calculate time time difference of arrival (TDOA) of each microphone $t_{x_k} = \Delta t_{x_k} - \Delta t_{x_1}$ where we used microphone $k = 1$ as a reference point. The delayed signals thus have $x_k(t) = x(t - \Delta t_{x_k}) = x\left(t - \Delta t_{x_k} + \Delta t_{x_1} - \Delta t_{x_1}\right) = x_1(t - t_{x_k})$. Similarly, for the noise sources we have $y_k(t) = t_1(t - t_{y_k})$.

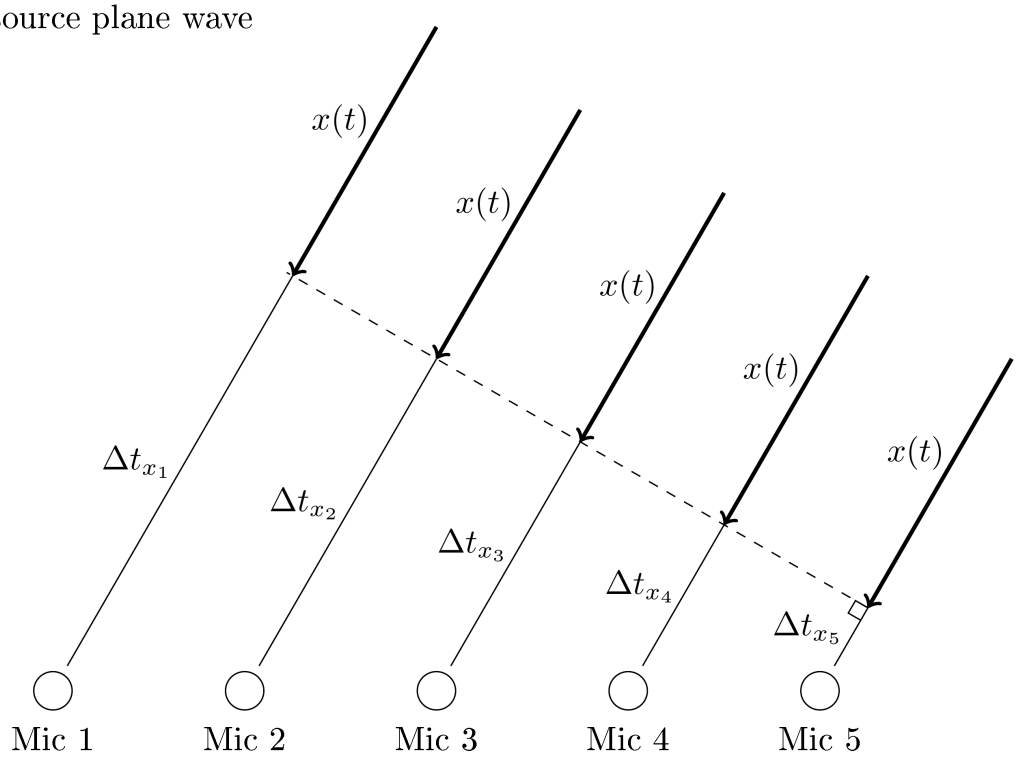
If the desired and noise sources appear at different angles, then their corresponding delays will be different. Moreover, if we add a signal $z(t)$ with itself at a random offset $\delta$, then the summation is destructive, that is, smaller than the original $\frac{1}{2}\left\|z(t) + z(t+\delta)\right\| \leq \left\|z(t)\right\|$ . Addition without an offset is obviously constructive, such that we can form the *delay and sum estimate* as

$$\hat{x}(t) = \frac{1}{K}\sum_{k=1}^{K} x_k(t - t_{x_k}).$$

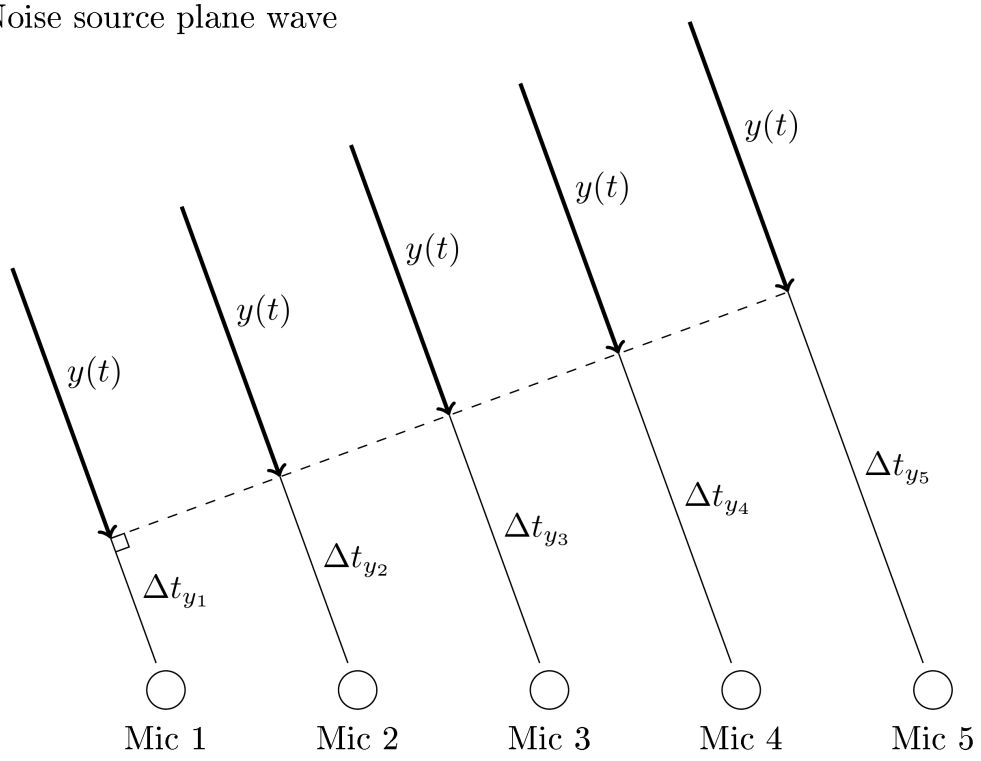
In this summation, all signals approaching from the same direction as the desired source will be additive (constructive) and other directions will be (more or less) destructive.

Trivial as it is, the delay-and-sum should however be treated as a pedagogical example only. It does not ideally amplify the desired source nor attenuate the noise source, and it is sensitive to errors in the TDOAs.

(a) Desired source plane wave




(b) Noise source plane wave



CHAPTER

TWELVE

# COMPUTATIONAL MODELS OF HUMAN LANGUAGE PROCESSING

One area of research making use of speech technology is the study of human language learning and processing. Language is a highly complex phenomenon with physical, biological, psychological, social and cultural dimensions. Therefore it is also studied across several disciplines, such as linguistics, neuroscience, psychology, and anthropology. While many of these fields primarily focus on empirical and theoretical work on language, computational models and simulations provide another important aspect to the research: capability to test theoretical models in practice. Implementation of models capable of processing real speech data requires techniques from speech processing and machine learning. For instance, techniques for speech *signal representation* and *pre-processing* are needed to interface the models with acoustic speech recordings. Different types of *classifiers and machine learning algorithms* are needed to implement learning mechanisms in the models or to analyze behavior of the developed models. In addition, model training data may be generated with *speech synthesizers* (e.g., [Havard *et al.*, 2017]), whereas linguistic reference data for model evaluation may be extracted from speech recordings using *automatic speech recognition*.

The basic idea of computational modeling is to understand how humans learn and process language by implementing human-like learning and speech processing capabilities as computational algorithms. The models are then exposed to inputs similar to what humans observe, and the model behavior is then recorded and compared to human data (Fig. 1). Computational models can focus on questions such as how adult speech perception operates (e.g., the highly-influential TRACE model of speech perception; McClelland and Elman [1986]), how language learning takes place in young children (native language aka. L1 learners; e.g., [Dupoux, 2018, Räsänen, 2012]) or in second-language (L2) learners, or they may study the emergence and evolution of language through communicative coordination between multiple agents (see, e.g., Kirby [2002], Steels [1997], for overviews).

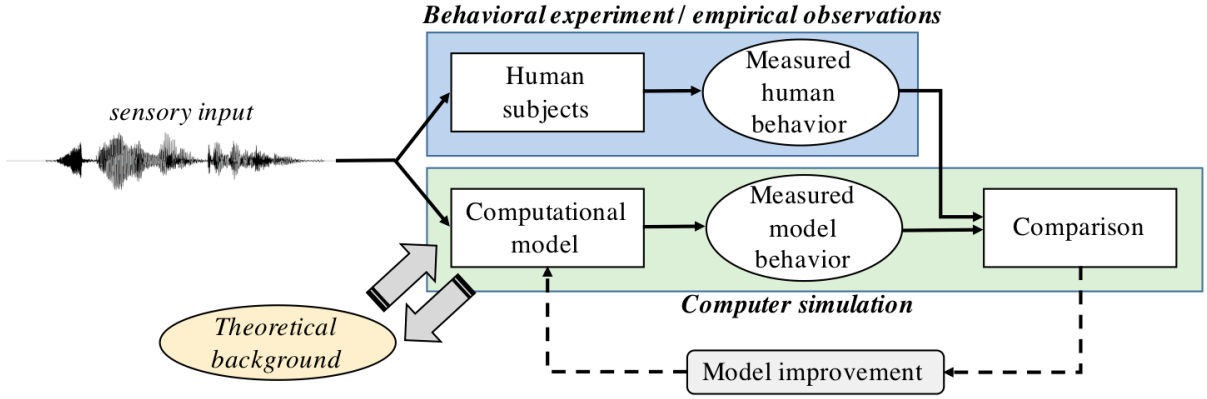



**Figure 1:** A high-level schematic view of a typical computational model development and evaluation process.

## 12.1 Human cognition as a sensorimotor information processing system

Computational modeling research is based on the metaphor of human brain as a computational information processing system. From an external observer viewpoint, this system perceives the environment using a number of input channels (senses), processes the information using some type of processing steps (the nervous system), and creates outputs (motor actions) based on the available sensory information and other internal states of the system. This input/output-relationship is affected by developmental factors and learning from earlier sensorimotor experience, realized as changes in the connectivity and structure of the central nervous system. Computational research attempts to understand the components of this perception-action loop by replacing the human physiology and neurophysiology with computational algorithms for sensory (or sensorimotor) information processing. Typically the aim is not to replicate information processing of the brain at the level of individual neurons, but to focus on the *computational and algorithmic principles* of the process, i.e., the *information representation*, *flow* and *transformation* within the system (see Marr's levels of analysis; Marr [1982]). These processing steps could then be implemented in infinitely many ways using different hardware (biological neurons, silicon chips architectures, CPU instruction sets, quantum computing etc.) or translations from computational description to implementation-specific instructions (consider, e.g., different programming languages with the same CPU instruction set). Despite the implementation differences, the observed behavior of the system in terms of inputs and the resulting outputs can still be similar.

To give an example, a model of adult spoken word recognition could focus on explaining the acoustic, phonetic and/or other linguistic factors that affect the process of word recognition. Such a model could focus on the details of how word recognition process evolves over time when a spoken word is heard, describing how alternative word candidates are being considered or rejected during this process (see, e.g., Magnuson *et al.* [2020], Weber and Scharenborg [2012], for examples). Even if the model would not focus on modeling neurons of the human brain, it could still explain how our minds decode linguistic information from speech input. This explanation could include how the process is affected by factors such as noisy environments, misprounciations, distributional characteristics of the input, or non-native language background of the listener—all useful information to understand both theoretical underpinnings and practical aspects of speech communication.

Another central aspect of the modeling is the relationship between human learning and computational methods trying to characterize the process. According to the present understanding, *human language learning is largely driven by* the interaction of *statistical regularities in the sensory input* available to the learner (e.g., Maye *et al.* [2002], Saffran *et al.* [1996], Saffran and Kirkham [2018], Werker and Tees [1984]), *innate mechanisms, constraints, and biases for perception and learning* from such input, and *other mechanisms responsible for social, communicative and exploratory needs o*f the learner. By extracting the statistical regularities from their sensorimotor linguistic enviroment, children are capable of learning any of the world's languages while fundamentally sharing the same basic cognitive mechanisms. A central topic in computational modeling of language acqusition is therefore to understand how much of language structure can be learned from the input data, and how much language-related prior knowledge needs to be built-in to the hard-coded mechanisms of these models. Note that human statistical learning is closely related to machine learning in computers, as both aim to extract statistical regularities from data using some sort of pre-specified learning principles. However, unlike standard speech technology systems such as *automatic speech recognition*, humans learners do not have access to data labels or consistent reward signals. For instance, a computational model of early infant word learning is essentially trying to find a solution to unsupervised pattern discovery problem: how to learn words from acoustic or multimodal input when there is no data labeling available. By applying a combination of speech processing and machine learning techniques to data representative of infant language experiences, explanation proposals for such a process can be created.

### 12.1.1 Computational modeling versus cognitive computationalism

Note that computational modeling and *representations* often studied in the models should not be confused with classical computationalism. The latter is loaded with certain assumptions regarding the nature of the entities processed by the computational system (e.g., *content of the representations, symbols*) and what are the basic computational operations (e.g., *symbol manipulation* using Turing machines). In contrast, computational models are simply descriptions of the studied process in terms of the described assumptions, inputs, outputs, and processing mechanisms without prescribing further *meaning* to the components (unless otherwise specified). For instance, *representations* of typical DSP and machine-learning -based models can simply be treated as quantifiable states, such as artificial neuron/layer activations, posterior distributions, neural layer weights, distribution parameters. In other words, the representations are scalars, vectors, or matrices that are somehow causally related to the inputs of the system. Behavior of these representations can then be correlated and compared with theoretical concepts regarding the phenomenon of interest (e.g., comparing selectivity of neural layer activations towards known phoneme categories in the acoustic input to the model; see, e.g., Nagamine *et al.* [2015]) or comparing the overall model behavior to human behavior with similar input (e.g., Räsänen and Rasilo [2015]). As long as the models are able to explain the data or phenomena of interest, the *models are a computational and hypothetical explanation* to the phenomenon without loading the components with additional theoretical or philosophical assumptions. Additional theoretical loading comes from the *data* and *evaluation protocols* chosen to investigate the models and in terms of how the modeling findings are interpreted.

## 12.2 Role of computational models in scientific research

Computational modeling has a role in scientific theory development and hypothesis testing by providing the means to test high-level theories of language processing with practical simulations (Fig. 2). This supports the more traditional approaches to language research that include collection of empirical data on human language processing, conducting brain research, or running controlled behavioral experiments in the laboratory or as real-world intervention studies. By implementing high-level conceptual models of language processing using real algorithms operating on real-world language data, one can test whether the models scale up to complexity of real-world sensory data accessible to human listeners. In addition to explaining already collected data on human language processing, computational models can also lead to new insights and hypotheses about the human processing to be tested in behavioral experiments.

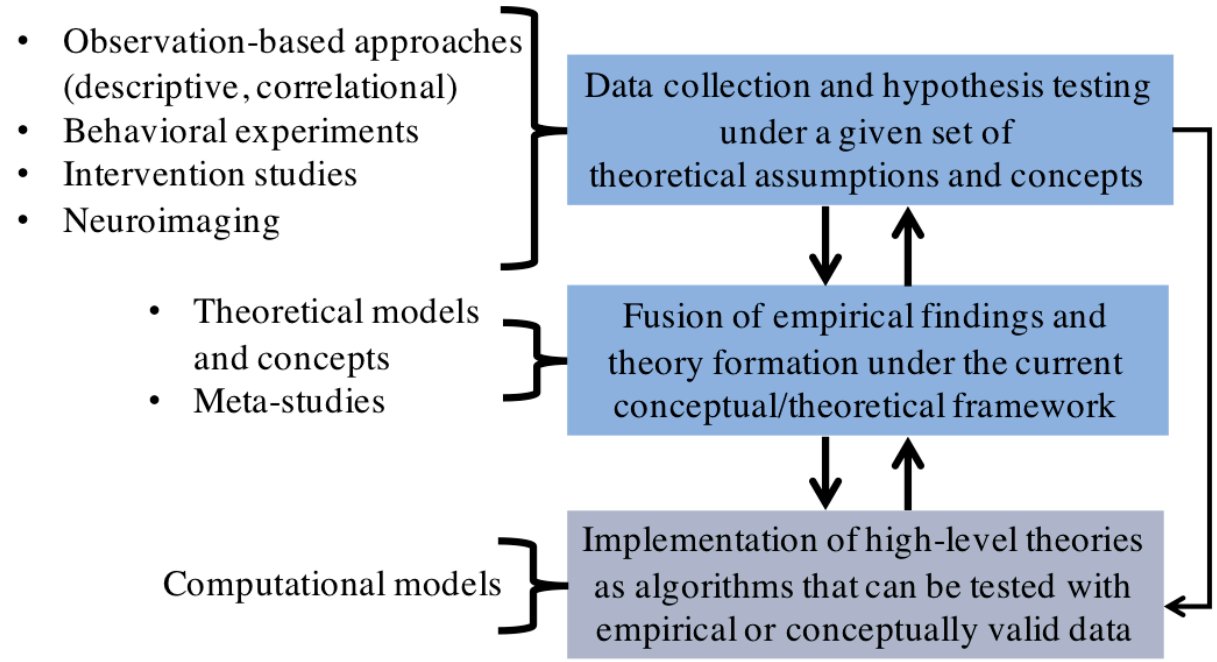



**Figure 2:** Different aspects of human language processing research and how they interact.Computational modeling uses

data from empirical research to test and inform high-level theories related to the given topic.

One potential advantage of computational modeling is its capability to address multiple processing mechanisms and language phenomena simultaneously. This is since *computational models can, and must, explicitly address all aspects of the information processing chain* from input data to the resulting behaviour. By first formulating theories of language processing in terms of computational goals and operations, then implementing them as functional signal processing and machine learning algorithms, and finally exposing them to realistic sensory data comparable to what real humans experience, ecological plausibility and validity of the underlying theories can be explicitly tested (cf. Marr [1982]). In contrast, behavioral experiments with real humans—although necessary for the advancement of our scientific understanding and for general data collection—can usually focus on only one phenomenon at a time due to the need for highly-controlled experimental setups. The fragmentation of focus also makes it difficult to combine knowledge from individual studies into holistic theoretical frameworks (e.g., understanding how phonemic, lexical, and syntactic learning are dependent on each other in early language development).

## 12.3 Examples of computational modeling research

***Computational models of child language development***: Computational models of language learning aim at understanding how human children learn to perceive and produce their native language. The basic idea is to simulate the learning of a human child, either starting from "birth" or from a specific stage of language development. Individual models typically aim to answer questions such as: how phonemic categories are learned, how word segmentation is achieved, how spoken words are associated with their referential meanings, or how syntax can be acquired? The grand challenge is to understand how the adult-like understanding of language as a discrete, symbolic, and compositional system can emerge from the exposure to noisy and inherently continuous sensorimotor environment. Typical computational modeling research questions include: 1) *to what extent are languages learnable from the statistics of sensory experiences*, 2) *what type of learning mechanisms or constraints are needed for the process*, and 3) *what kind of and how much data* ("experiences") are required in the process (quality and quantity of speech, uni- vs. multimodal input etc.). A broader view takes into account the fact that the children are not just passive receivers of sensory information but can interact with their caregivers and their environment as active explorers and learners. Therefore it is also of interest 4) *what type of additional interaction-related mechanisms and dynamically created experiences are critica*l. The big and yet unaswered question is what are the critical ingredients for successful language learning, as all normally developing children with very different language experiences, environments, and also somewhat differing cognitive skills still manage to converge to a shared communicative system of their native language. As the short-term outcomes, models of language learning can test and propose different hypotheses for different aspects of language learning. They also produce functional algorithms for processing acoustic or multimodal language data in low-resource settings, where access to data labels is limited (e.g., Kakouros and Räsänen [2016], Kamper *et al.* [2017], Räsänen *et al.* [2018]). Long-term outcomes from language acquisition modeling contribute to both basic science and practice. In terms of basic science, the research tries to answer the question of how one of the most advanced aspects of human cognition, i.e., language, operates. Long-term practical goals include understanding the impact of external factors in language development and how to ensure equally supportive environments for children in different social settings, understanding different types of language-related disorders and how to best respond to them, but also how to develop autonomous AI systems capable of human-like language learning and understanding without supervised training, i.e.., development of systems ultimately capable of *understanding the intentions and meaning in communication*. Computational modeling of early language acquisition is closely related to zero-resource speech processing (see http://www.zerospeech.com/) that aims at algorithms capable of unsupervised language learning from speech data.

***Models of spoken word recognition****:* Another widely studied topic is speech perception in adults. Computational models developed for this purpose attempt to explain how the brain processes incoming speech in order to recognize words in the input. Models in this area may focus on explaining the interaction between sub-word and word-level units in perception, on how words compete with each other during the recognition process, or, e.g., on how the speech perception is affected by noise in native and non-native listeners. Since word recognition is essentially a temporal process, particular attention is typically paid to the evolution of the recognition process as a function of time (or proportion of input word or utterance perceived). For an overview, see Weber and Scharenborg [2012]. For some examples of models, see Magnuson *et al.* [2020], McClelland and Elman [1986], Norris [1994].

***Models of speech production***: This line of research attempts to explain how human speech production works in terms of articulators and their motor control. Some studies also focus on the acquisition of speech production skills. Typical speech production models involve an articulatory *speech synthesizer*—an algorithm capable of producing audible speech signals by modeling the physical characteristics of the vocal apparatus—and some type of motor control algorithms that are responsible for phonation and articulator movements. Sometimes hearing system is simulated as well. These models have various uses from general understanding of the articulatory basis of speech to understanding speech pathologies, articulatory learning in childhood or adulthood, or special types of sound production such as singing. For classical and more recent examples of articulatory models of speech production, see, e.g., Birkholz [2005], Birkholz *et al.* [2015], Maeda [1988]. For models of infant learning of speech production, see, e.g.,Howard and Messum [2014], Rasilo and Räsänen [2017], Tourville and Guenther [2011].

***Multi-agent models of language learning, evolution and communication:*** Languages are essentially cultural conventions based on social activity, enabled by genetically coded cognitive and physiological mechanisms, and learned through interactions between people. One branch of computational modeling focuses on understanding how languages emerge, evolve, and are learned through multi-agent communication and interaction. These simulations, sometimes referred to as *language games* or *iterated learning* (see Kirby [2002]), focus on non-linear dynamical systems that result from the interaction of multiple communicative computational agents. These agents can be purely based on simulation, or they can be based on physical robots interacting in a shared physical environment. By providing the agents with different types of innate goals, mechanisms, learning skills and environmental conditions, one can study the extent that language-like signaling systems (as a social system) or language skills (as subjective capabilities) can emerge from such conditions. For overviews, see Kirby [2002], Steels [1997].

## 12.4 References and further reading

Birkholz, P.: VocalTractLab: http://www.vocaltractlab.de/ [for work on articulatory synthesis]

Dupoux, E. et al.: Zero Resource Speech Challenge: http://www.zerospeech.com/ [a challenge on unsupervised speech pattern learning]

CHAPTER

# THIRTEEN

# SECURITY AND PRIVACY IN SPEECH TECHNOLOGY

***DISCLAIMER:*** *This document is meant to be an introduction to questions in security and privacy in speech technology for engineering students, such that they would understand the main problematic. In particular, this is not a legal document. In real-life application of technology and data collection, you must consult legal experts to determine whether you follow the law. You are responsible.*

The right to privacy is a widely accepted concept though its definition varies. It is however clear that people tend to think that some things are "theirs", that they have ownership of *things*, including information about themselves. A possible definition of privacy would then be "the absence of attention from others" and correspondingly security could be defined as the protection of that which one owns, including material and immaterial things. It however must be emphasised that there are no widely shared and accepted definitions and in particular, the legal community has a wide range of definitions depending on the context and field of application.

From the perspective of speech technology, security and privacy has two principal aspects;

- Security and privacy of data related to speech signals and
- Protection against attacks which use speech signals as a tool.

The latter aspect is mainly related to speaker identity; fraudsters can for example synthesise speech which mimics (spoofs) a target person to gain access to restricted systems, such as access to the bank account of the target person. Such use cases fall mainly under the discussions under *speaker recognition and verification*, and not discussed further here.

Observe that in the isolated category of telephony (classical telephone connections) privacy and security already have well-established ethical standards as well as legislation. In typical jurisdictions, telephone calls are private in the sense that only the "intended" participants can listen to them and sometimes even recording them is restricted. Covert listening is usually allowed only for the police and even for them only in specially regulated situations, such as with a permission granted by a court or judge.

[Lareo, 2019, Nautsch *et al.*, 2019]

## 13.1 System models

### 13.1.1 All-human interaction

Speech is a tool for communication such that it is generally sensible to always discuss interactions between two agents, say, Alice and Bob. The interaction between them is the desired function such that the information exchanged there is explicitly permitted. By choosing to talk with each other, they both reveal information to the extent speech contains such information.

#### Primary interaction

```
<IPython.core.display.SVG object>
```

Though Alice and Bob knowingly and intentionally interact, they might reveal private things. This is the classic "*slip of the tongue*".

#### Secondary interactions

A second-order question are third parties, who are not part of the main speech interaction. The pertinent question is the degree to which the third party is allowed to partake in an interaction. As a practical example, suppose Alice and Bob have a romantic dinner at a restaurant. To which extent is the waitress Eve allowed to interact with the discussion of Alice and Bob? Clearly Eve has some necessary tasks such that interaction is unavoidable. Will Alice and Bob, for example, pause their discussion when Eve approaches?

```
<IPython.core.display.SVG object>
```

Observe that we have here labelled Eve as a "*restricted*" and not as an "*unauthorized*" interactor. If access is unauthorized, then it is clear that Eve should not have any access to the speech interaction, which is generally straightforward to handle. The word restricted, on the other hand, implies that unimpeded access should not be granted, but that some access can be allowed. It is thus not question of "*if*" access should be granted but "*how much?*".

#### Ownership and personal privacy

Privacy is closely connected to ownership of immaterial property, that is, information. Such ownership can also be translated to the question of *who has the control* over some information? In terms of *personal privacy*, it is clear that it relates to information to which a single person can claim ownership.

Speech is more complicated. Speech is a form of communication and thus relates to an interaction between two parties. Dialogues can also commonly lead to co-creation of meaning, where new information is generated through the dialogue in a form which none of the involved parties could have alone produced [Gasiorek, 2018]. None of the users can thus claim sole ownership of the information, but the ownership is shared. Currently we do not have the tools for handling such shared ownership.

### 13.1.2 Interactions which involve devices or services

#### Telecommunication

```
<IPython.core.display.SVG object>
```

Talking over the phone and video conference involve transmission of speech through a telecommunication service. Here we consider scenarios where the telecommunication device does not include any advanced functionalities or artificial intelligence. Most countries have clearly definied rules that specify the situations when such communication can be eavesdropped. In most jurisdictions, only the police is allowed to intercept such traffic and only in specific situations.

```
<IPython.core.display.SVG object>
```

Such interception of private communication is interesting primarily from ethical and legal perspectives, but does not contain technological challenges related to the communication itself. The main technological challenges are related to forensics;

- What information can the police extract (e.g. speaker identity, emotions and health)?
- How can speakers (and service providers) protect themselves from unauthorized interception by, for example, stripping away information such as speaker identity, from the transmitted signal?

### Discussion in the presence of speech interfaces

A commonly occuring scenario is one where two or more users engage in a discussion such that there is one or more speech operated devices nearby. For example, a user could have their mobile phone or there could be a smart speaker nearby.

```
<IPython.core.display.SVG object>
```

Often, one of the users will be the primary user of said devices (e.g. it is *their* phone), so the question is how the devices should relate to the other users. For example, suppose Alice has a smart speaker at home and Bob comes to visit. What would be the appropriate approach then for both Alice and the agent? Should Alice or the agent notify Bob of the presence of an agent? Or should the agent automatically detect the presence of Bob and change its behaviour (e.g. go to sleep)?

We seem to lack both the cultural habits which dictate how to handle such situations, the legal tools which regulates such situations as well as the technical tools to manage multi-user access.

### Interaction with a speech interface

An interaction with a speech interface or agent is surprisingly free of problems as long as the agent is not connected to any outside entity. We can think of the agent as a local device. If nobody else has access to that device, then all the information remains in the user's direct control. Even if the agent exist in a remote cloud-service, if information remains strictly within the desired service, there are very little problems to consider.

```
<IPython.core.display.SVG object>
```

An exception is analysis, which the agent can perform, which is not related to the desired service, which can take abusive forms. For example, suppose the agent analyses the user's voice for health problems and identifies that the user has Alzheimer's disease. What should the agent then do with that information? Not doing anthing seems unethical - getting early access to medical services could greatly improve life quality. Informing the user, on the other hand, involves risks. How will the user react to the information? Is the user sufficiently psychologically stable to handle it? What if the analysis is incorrect and the agent thus causes suffering? It is also easy to think of further problematic scenarios.

### Multi-user interaction with a speech interface

An agent can be involved in an interaction with multiple users at the same time.

```
<IPython.core.display.SVG object>
```

This scenario differs from the single-user case in particular in the way the device can store information. To which extent should different combinations of users be permitted to have access to information from prior interactions? Quite obviously Alice should not have default, unrestricted access to Bob's prior interactions without Bob's permission. Where Alice and Bob have engaged in a joint discussion, the question of access becomes more complicated. It would seem natural that both can have access to information about their prior discussions. However, if Bob is in a discussion with Eve, then access to prior discussion between Bob and Alice should again be restricted. The rules governing access will thus be complicated, often non-obvious and they will have many exceptions.

#### Interaction with a speech interface in the presence of others

In the early days of mobile phones, a common *faux pas* was to speak loudly on the phone in public places such as on a bus or subway. It seems that it causes an uncomfortable feeling to people when they overhear private discussions. It can also be hard to ignore speech when you hear it. Obviously, the reverse is also true, participants of a private discussion often feel uncomfortable if they fear that outsiders can hear their discussion.

```
<IPython.core.display.SVG object>
```

The same applies to speech operated devices. Such interactions can be private, but even when they are not, the fact that they can be overheard is often uncomfortable to all parties.

This is a problem when designing user interfaces to services. Speech interaction is often a natural way to user a service or device, but it is not practical in locations where other people can overhear private information and where it the sound is annoying to other people present.

#### Interaction with a speech interface connected to other services

When interacting with a speech interface, users typically do it with a specific objective in mind. For example, suppose Alice wants to turn off the lights in the bedroom and says "*Computer, lights off*". To which extent is it permissible that that information is relayed to a cloud-service? If the local device is unable to or uncapable of deciphering the command, it can transmit it to the cloud. The cloud-service then obtains information from a very private part of Alice's life.

```
<IPython.core.display.SVG object>
```

Information obtained this way can be very useful for example, to advertisers. By analyzing the habits of users, they can serve more meaningful advertisements. Arguably, by better targeting of users, advertisement can more effective, which could *potentially* reduce the need for advertisement. It is however questionable whether advertisers ever would have incentives to *reduce* the amount of advertisement. Still, some people are creeped out by "*overly fitting*" advertisement.

There are however plenty of other scenarios which are more potent sources of danger. What if insurance agencies analyze users life patterns and increase payments for at-risk users such as substance abusers? Some smart devices already today can call the emergency services if they recognize cries of help or other obvious signs of distress. What are the moral dilemmas of that?

### 13.1.3 Multi-user and multi-device scenarios

Things get even more complicated when multipled users and/or users co-exist in the same space. Consider, for example, an open office with two users simulatenously engaged in independent video conferences.

```
<IPython.core.display.SVG object>
```

## 13.2 Information contained in speech signals

Speech signals contain a wide variety of information which are often potentially private or even sensitive and it is difficult or impossible to list all categories of potential information. However, information which speech at least contains includes for example;

- The linguistic (text) content of a speech signal can contain almost anything
  - If you reveal private information about yourself in a conversation, then clearly that information is contained in a transcription of the conversation.

  - Word-choices and manners of speaking can reveal things about the speaker, without the speaker realizing it himself or herself.

- Para-linguistic (i.e. non-linguistic viz. non-text) content has a wide variety of information:
  - Speaker identity
  - Physical traits, including gender, body size and age
  - Emotional state (happy, sad, angry, excited etc.)
  - Speaking style (public, intimate, theatrical etc.)
  - State of health, such as flu, mental health and diseases like Alzheimer's, but also including tiredness and intoxication.
  - Association and affiliation with reference groups, including groups of gender-identity, ethnicity, culture background, geographic background, nationality, political and religious affiliations etc.
- In interaction with other speakers:
  - Proof that you have met with the other speaker
  - Level of familiarity, intimacy and trust.
  - Power structures (leader/follower/partner)
  - Family, romantic and other relationships

In other words, speech contains or can contain just about all types of private and sensitive information you could imagine. As speech is a tool for communication, this is not surprising; anything we can communicate about, can be spoken. Conversely, if we find that (and we do find that) privacy is important, then speech is among the most important signals to protect.

## 13.3 Types of privacy

In a more general scope than just speech, privacy can be categorized into seven types: [Finn *et al.*, 2013]

- **Privacy of the person**, which refers to the privacy with respect to our physical body as well as any information about our body such as fingerprints, voice characteristics and medical history.
- **Privacy of behaviour and action**, refers to privacy with respect to what we do; for example, nobody needs to know what I do within the confine of my home.
- **Privacy of communication**, is particularly important in speech communication and refers to privacy of the content and meta-content of communication. That is, it is not only about the literal content of a communication, but also about the style of communication and also about the fact that communication has happened.
- **Privacy of data, including images and sound**, which extends the privacy of possessions to immaterial things. In particular, this addresses privacy with regard to sharing information about a third person.
- **Privacy of thoughts and feelings**, is paraphrasing, the privacy of thinking. We have the right to share our thoughts and emotions with whom we like, or to choose not to share our feelings with anyone. This does not mean that people would have to listen to you, but that your are allowed to offer them your thoughts.
- **Privacy of location and space**, has become increasingly important, since so many of our mobile devices has the ability to track your location. However, this type of privacy covers both your location and location history (as in location tracking).
- **Privacy of association**, refers to the privacy of whether you belong or to which extent you otherwise associate yourself to a particular group (religious, political, ethnic, gender identity, professional, interest groups etc.).

Note that this list does not make any claims with respect to *rights* to these types of privacy, but that privacy-issues can be often be split into these sub-topics. Whether someone has a right to privacy is a society-level decision and political choice, where psychological and cultural aspects play a big role.

## 13.4 Threat and attack scenarios

Threats to privacy in speech communication can almost always be defined as covert extraction of information as well as storage, processing and usage of that information in ways of which the speaker is not aware, and/or with which the speaker does not agree. Variability in scenarios is then almost entirely due to the type of information involved as well as the stakeholders. In particular,

- Companies can extract information from private users for unethical advantage
  - Insurance policies and mortgages can be denied on based on covertly extracted information
  - The price of services can be increased for vulnerable groups
  - Access to information (search results) can be restricted and information can be targeted (advertisement) to covertly influence users for unethical advantage. Such behaviour is often advertised under the pretence of customizing services to the users preferences, but without giving the user any tools for choosing how services are customized.
- State operators can use surveillance and access restrictions on their own citizens
  - Authoritative regimes can track and eavesdrop the political opposition, dissent and other groups such as religious, ethnic and sexual orientations.
  - Also legitimate uses by police for surveillance in criminal investigations
- State operators can use surveillance on foreign citizens
  - Spies can use speech technology for (remote) eavesdropping and information extraction
- Criminals can steal information and use it for their advantage
  - Identity theft
  - Paparazzi's can steal private information of famous people
  - Private information can potentially be used for extortion
  - Explicit content could be sold as entertainment
- Private persons can covertly use speech technology for eavesdropping and extracting information of other persons
  - For example, the command history of smart speakers can give access to past commands of all users, also when speech commands have been made in private.

## 13.5 Privacy and security scandals

Most of the threats and attack scenarios are not familiar to the common public and some of them might be too abstract to be relevant to average users. Typically, we can hypothesize that scenarios which do not touch directly on the life of an individual, probably do not get much attention in the media. Topics in privacy and security which however have received attention in the public media include:

- Amazon workers are listening to what you tell Alexa (Bloomberg, 2019). Later it was revealed that Google, Apple and others are doing the same.
- Amazon Sends 1,700 Alexa Voice Recordings to a Random Person (Threatpost, 2018).

- Amazon's Alexa recorded private conversation and sent it to random contact (The Guardian, 2018).
- Fraudsters Used AI to Mimic CEO's Voice in Unusual Cybercrime Case (The Wall Street Journal, 2019).

Note that the fact that many of the above examples are related to Amazon/Alexa is probably more coincidence than an indication that Alexa would treat privacy differently than its competitors.

## 13.6 Approaches for safeguarding privacy in and improving usability of speech technology

### 13.6.1 Basic design concepts

At least from the European perspective, the following design concepts are seen as basis of good design for privacy. In fact, they are mandated by the General Data Protection Regulations of the European Union.

- *Privacy by default*; systems should always default to the least invasive configuration. Additional services, which are more invasive, can be chosen (opt-in) if the user so chooses.
- *Privacy by design*; privacy should not be an after-thought but systems should be designed for privacy. The overall systems structure should be chosen such that it supports privacy.
- *Data minimization*; data can be extracted from users only the extent explicitly required by the system. For example, if you order a pizza through a web-portal, you need to tell which pizza you want and where it should be delivered. Otherwise you cannot receive the service. In other words, all services require some level of information transfer, but the services cannot ask about information which is irrelevant to the service. For example, the pizza service cannot ask you about your gender-identity. Data minimization cab also be interpreted as *data ecology*, which would underline the fact that "more data" usually means also a higher consumption of energy and other resources. In fact, data could (or should) be treated as a natural resource itself. This line of argumentation connects privacy with the UN sustainability goals.An important aspect of data minimization is that information should stored only as long as it is necessary for the service.
- *Informed and meaningful consent*; when the user chooses a service, he should receive information about its implications to privacy in an understandable and accessible way, and the service provider should receive the users consent without any form of coercion.

Typically it is reasonable that service providers have access to aggregated data such as ensemble averages, but not to information about individuals. For example, in a hypothetical case, a smart speaker operator could receive the information that 55% of users are male, but would not get the gender of any individual user.

### 13.6.2 Definitions

A central problematic in privacy with speech signals is the concept of "uniquely identifiable". Legal frameworks such as the GDPR state that private information is such data where individual users are "uniquely identifiable", but there is no accurate definition of what it really means. If your partner recognizes your "Hello" on the phone, it means that for her, your "Hello" is uniquely identifiable. However, if you give 10.000 speech samples to your partner, one of which is your "Hello", then there's a significant likelihood that your partner would not find your "Hello" from the pile. An unanswered question is thus, "What is the size of the group where a user should be uniquely identifiable?".

A more detailed aspect is that of significance. The speaker recognition approach is to find the most likely speaker, out of the reference group of size $N$, whereas speaker verification tries to determine whether we, within some confidence intervals, can be sure that you are who you claim you are. In engineering terms, this means that we want to find the speaker with the highest likelihood, but with a sufficient margin to all other speakers. In the opposite direction, we can also use a lower threshold; we could say that statistically significant correlation already exposes the users privacy. For

example, if we find that the speaker is either you or your father/mother, then we have a significant statistical correlation, but your are not uniquely identified.

A further consideration is that of adjoining data; Suppose there is a recording of a speaker A, and that you happen to know a speaker A very well. Then it will be easy for you to recognize the voice of A in that recording. That is, you have a lot of experience (stored data) about how A sounds, therefore it is easy for you to identify A. Does that mean that A is uniquely identifiable in that recording? After all, A would not be identifiable if you did not know A (= if you would not have prior, stored data about A).A slight variation of the above case is a recording of a speaker B, where B is relatively famous public person, such that there are readily available sound samples of his voice on-line. Does that make the recording of B uniquely identifiable? Or if there is a recording of a currently non-famous person C, who later becomes famous. Does that change the status of the recording of C to uniquely identifiable?

Today, this question remains unanswered and we have no commonly agreed interpretation of what "uniquely identifiable" really means. What level of statistical confidence is assumed? What level of adjoining data is assumed (in terms of GDPR probably: any and all data which exists)? Can it change over time if new information becomes public (probably: yes)? Can it change over time if new technologies are developed (probably: yes)?

### 13.6.3 Local/edge processing

Privacy is an issue only if some other party has access to data about you. Data which resides on a device which is in your control is therefore relatively safe, assuming that no outsider has access to that device. If data is sent to a cloud server then there are more entities which could potentially have access to your data. Therefore all storage and processing which can be done on your local device is usually by design more private than any cloud server.

Observe that this does not protect you from other local users. For example, if multiple persons are using one smart speaker at home, then the other users could have access to information about you through that device and any connected other devices.

Central limitations of edge processing are

- Many services require outside access; say if you ask your phone "What's the weather tomorrow?", the phone cannot know that by itself, but has to retrieve the information from a cloud server. The essential content of your speech is therefore relayed to the cloud and you don't have much benefit from edge processing.
- Improving voice operated services requires a lot of data. Moreover, data which reflects features of actual users is much better than any simulations. Service providers thus argue that they need to collect data from users to provide high-quality services, and local processing could prevent the services providers from getting that data.
- Edge devices would use their full capacity only when they pick up speech in their microphone, which means that most of the time, edge devices would lie dormant, waiting for speech commands. This is a wasteful use of resources; a cloud server can better balance the load because with a large number of users, the resource requirements would likely be more stable.

### 13.6.4 Differential privacy

Even when operating with aggregate data, like the mean user age, it is still possible to extract private information in some scenarios. For example, if we know the mean user age and the number of users at a time $t$, and we also know that the age of user $X$ was added to the mean at time $t + 1$, as well as the mean user age at $t + 1$, then we can deduce the age of user $X$ with basic algebra. As a safeguard against such differential attacks, to provide differential privacy, it is possible to add noise to any data transfers. Individual data points are then obfuscated and cannot be exactly recovered. However, the ensemble average can still be deduced if the distribution of the added noise is known.

The required compromise here is that the level of privacy corresponds to amount of noise, which is inversely proportional to the accuracy of the ensemble mean. That is, if the amount of noise is large, then we need a huge number of users to determine an accurate ensemble average. On the other hand, if the amount of noise is small, then we can get a fair guess of an individual data point, but also the ensemble average is accurate.

### 13.6.5 Federated learning

To enable machine learning in the cloud without the need to provide access to private data, we can use federated learning, where private data remains on the local device, but only model updates are sent to the cloud. Clearly this is approach has better privacy than one where all private data is sent to the cloud. However, currently we do not yet have clear understanding of the extent of privacy with this type of methods; some data is sent to the cloud, but can some private data still be traced back to the user?

### 13.6.6 Homomorphic encryption

Suppose a service provider has a proprietary model, say an analysis method for Alzheimer's disease from the voice, and your doctor would like to analyse your voice with that method. Naturally your voice is also private, so you do not want to send your voice to the third-party service provider, but also the service provider does not want to send the model to you. Homomorphic encryption provides a method for applying the secret model on *encrypted* data, such that you have to only send your data in an encrypted form to the service provider. Your doctor would then receive only the final diagnosis, but not the model nor your speech data. The concept is in principle beautiful, it solves the problem of mutual distrust very nicely. However, the compromise is that currently available homomorphic encryption methods require that all processing functions can be written as polynomial functions. In theory, we can transform any function to a corresponding polynomial, but the increase in complexity is often dramatic. Consequently, privacy-preserving methods based on homomorphic encryption typically have a prohibitively high computational complexity.

### 13.6.7 myData

In addition to privacy-preserving algorithms, we can also design privacy-preserving architectures. The myData paradigm is based on a three-tier design, where the user can choose where all his/her data is stored and where the user can give access for service providers to his/her data when required. The idea is to separate service providers from data storage, such that users have better control over his/her data. To transform existing services to adhere with the myData concept requires that new storage services for private data are created and that APIs between storage and processing services are specified.

Note that, if a user chooses to store private data on a cloud-server, then it is still susceptible for abuse by the storage-service-provider, unless appropriate encryption methods are used. However, the user could in principle choose to store private data on a edge device, such that the storage-service-provider is cut out of the loop.

A further risk is that in the myData concept, we usually assume that data is stored at a single central location, which becomes a central point of weakness. Should someone gain illegitimate access to the storage, then all your data would be compromised. Distributing data to several different storage locations might therefore be reasonable.

## 13.7 Design goals, human computer interfaces and user experience

A common prejudice is that privacy and security requirements cause problems for developers and make systems more difficult for users to use. Such prejudice are unfortunate and patently misguided. The problem is that many privacy problems are not visible to the casual observer and their effects become apparent only when it already is too late. Another argument is "*privacy is not my concern because I haven't seen any privacy problems*", which is like saying that "*rape is not my concern because I haven't seen any rapes*". This is thus an absurd argument. Privacy safeguards are meant to protect users and developers from very bad consequences. These problems are real. *You* cannot ignore them.

However, designing for privacy is also *not only* about protection of users. It is also very much about designing technology which is *easy to use* and where the user experience feels intuitive and natural. For example, speaker recognition can be used to grant access to voice technology such that the user does not have to be bothered with passwords, PIN-codes or other cumbersome authentication methods. Overall speech technology promises to give access to services without the

need scroll through menus on your washing machine to find that one mode which is optimized for white curtains made out of cotton.

The overall design goal could be that people should be able to *trust* the system. In particular, a trustworthy system will be

- *consistent*; It does what it says, and it says what it does.
- *competent*; It is able to do what it should do and what it says it does.
- *benevolent*; It cares for the user and it shows that in both *what* is says and *how* it says it.

These goals are best illustrated by examples;

- We should avoid cognitive dissonance; if the computer has the intellectual capacity corresponding to a 3-year-old child, but if it then speaks with the authoritative voice of a middle-aged person, it is giving mixed signals which can confuse users. Similarly, if a computer leaks out all your information to the world, while speaking with an intimate voice, it is also giving the wrong impression to the user. Conversely, the intuitive impression which a device gives should be consistent with its true nature.
- Consistency is extremely important and one mistake can be very costly; If your friend Greg once tells about your intimate health-incident to your friends, it casts a shadow of doubt over all future and all of the past 10 years. Did he already earlier tell my secrets to them? Who else has he told my secrets to? It takes a very long time to patch such breaches of trust.

## 13.8 Ethical dilemmas

The following is a list of hypothetical questions which can (and do) arise in the design of speech operated systems:

- If a device hears cries of help, should it call the police automatically, even when it breaches privacy and the trust of the user?
- If a device hears indications of domestic abuse, but has no direct evidence, should it call the police? What level of evidence is required?
- If a device or service recognizes that you have Alzheimer's or some other serious illness, should it tell it to you? Even if your doctor would not yet have noticed it? Even if the diagnosis might be incorrect? Even if you might choose not to want to hear it? Even if you'd have a history of depression and such bad news might trigger suicidal thoughts?
- Suppose you have been dreaming of buying a new bicycle, but haven't told anyone. Your smart TV, though, knows about it because you have searched for information about that bicycle. Suppose that your spouse is simultaneously trying to come up with a nice surprise birthday present. Should your smart TV suggest to your spouse that she buys the bicycle?
- Suppose your local cafeteria has automatic speech operated ordering of drinks. On this day, last year, you bought a birthday-surprise drink from this cafeteria. Should the computer remember that and congratulate you today about your birthday?
- The better your services know you, the better they could serve you. That's undoubtedly a fact. (The fact that current services are not optimal is not a contradiction.) However, do you want to have privacy from devices in the same way you have privacy from your friends. For example, your friends do not follow you to the toilet or the bedroom; is it ok if your device does that?

## 13.9 Security and privacy in speech research

Scientific research is based on arguments supported by evidence, where evidence, in the speech sciences, is recordings of speech. Access to speech data is therefore a mandatory part of research in the speech sciences. To obtain trustworthy results, independent researchers have to be able to verify each others results, which means that they have to have access to the same or practically identical data sources. Shared data is the gold standard for reproducible research. However, the sharing of speech data can be problematic with respect to speaker privacy.

The concept of "uniquely identifiable" is here the key. If an individual is *not* uniquely identifiable in a data set, then you are allowed to share that data. Conversely, if you remove all identifying data, then you can share data relatively freely. However, in perspective of the discussions above, it should be clear that it is not clear what constitutes identifying data nor is it clear what makes that data "uniquely" identifying.

A second important consideration is *consent*. The persons whose voices are recorded must be allowed to choose freely whether they want to participate and that choice has to be explicit; you need to ask them clearly whether they want to participate in a recording. The research needs to be able to prove that consent has been given, and therefore that consent must be documented carefully. Consent must also be given freely such that there are no explicit or hidden penalties of rejecting consent. Furthermore, if any uniquely identifying data of a participant is stored, then the participant must be allowed to withdraw consent afterwards. There are however some important exemptions to this rule; the right to withdraw consent can be rejected, for example, if that would corrupt the integrity of the data set, such as

- If withdrawal of a participant could bias the results, then that *could* be grounds for denying withdrawal. For example, if a dataset is constructed in a way that it represents a balanced subset of the population (the amount of say, males and females, different age, cultural and education backgrounds are chosen to match the general population), then we cannot remove any participants without corrupting the distribution. Moreover, people more educated in questions of privacy could hypothetically be more prone to withdrawing their consent, such that the population becomes biased.
- If withdrawal of a participant could jeopardize the reproducibility of results, then that *could* be grounds for denying withdrawal. For example, if a machine learning algorithm is used on a dataset, then we can recreate that algorithm only if we have *exactly* the same data available. This is especially problematic if the dataset is relatively small, where small changes in the dataset can have big consequences on the output.

To allow plausible grounds for denying the right to withdraw consent, datasets can then be designed to be either balanced or relatively small. Collecting balanced datasets is good practice in any case, such that this is not a limitation but can actually improve quality. Conversely, good data is balanced and that should be our goal; A consequence is that we *might be forced* to deny the right to withdraw consent. Avoiding the collection of excessively large data sets is also good from the perspective of *data minimization* and data ecology.

In a request for consent, the data collector should state the purpose of the dataset (i.e. *purpose binding*). For instance, a dataset could be collected for development of wake-word detection methods and consent is received for that purpose. Then *it is not permissible* to use the same data set for speaker detection experiments or medical analysis of the voice. Period. It is therefore good practice to ask for consent in a sufficiently wide way, such that researchers have some flexibility in using the data. Blanket consent to all research purposes is however not good practice. In particular, it is recommend that processing of sensitive information such as health, ethnic, political information is excluded if it is not the express purpose of the dataset (cf. data minimization).

If a dataset by nature does include uniquely identifiable data, then the researchers need to apply stronger layers of safeguards. In particular, typically researchers have to keep track of who has access to the data, to ensure purpose binding and to allow withdrawal of consent. This could also require that any researcher who downloads the data signs a contract with the data provider, where the terms of usage are defined. Such a contract can be required in any case, not only with uniquely identifiable data.

Data such as medical information, data about children or other exposed groups, political, religious and gender-identity affiliations etc. are particularly *sensitive*. If your dataset contains *any* such information, then you have to apply stronger safeguards. To begin with, access to such data has to be, in practice, always limited to only persons who are included in a legally binding contract specifying access rights and allowable uses, processing and storage.

As an overall principle, note that the principal investigator (research group leader) is legally responsible for the use of the data that is collected, stored and processed. In particular, if a third party downloads the data and misuses it, for example by analysing health information even if no consent has been acquired for that purpose, then it is the principal investigator who is responsible. However, the principal investigator is only required to apply *reasonable safeguards* to ensure that data is not misused. What level of safeguards are sufficient has however not yet been agreed. It is likely that there will never be rules which specify exactly a sufficient level of safeguards.

In the above discussion it has become clear that the nature of *unique identifiability* can change over time, when new information is published and new technologies emerge. This means that datasets which previously were adequately protected, over time become exposed to privacy problems. It is therefore important that researchers monitor their published datasets over time such that if new threats emerge, they can take appropriate action. For example, they could withdraw an dataset entirely. Reasonable ways for implementing this could be:

- *Expiry date*; All datasets should have a clearly stated shelf-life and use of the dataset after the expiry date should be prohibited. The manager of the dataset could update the expiry date if no new threats are discovered. Academic publications based on expired datasets should not be accepted.
- *Controlled access to datasets*; To enforce purpose binding and to enable withdrawal of datasets, the data manager can require that all users are registered and sign a formal contract which specifies accepted uses.

As a last resort, when data is so sensitive and private that it cannot be publicly released, it is possible to require on-site processing of data. For example, you can design a computing architecture, where data resides on a secure server, to which researcher have access through a secure API. Data never leaves the server such that privacy is always preserved. For an even higher level of security, data can be stored on an air-gapped computer system, which means that access to the data requires that researchers physically come to the computer (no network access). This level of security is usually the domain of military-grade systems.

## 13.10 References


- ISCA Special Interest Group “Security and Privacy in Speech Communication”, https://www.spsc-sig.org/

CHAPTER

# FOURTEEN

# REFERENCES

# BIBLIOGRAPHY


[Nol75] Peter Noll. A comparative study of various quantization schemes for speech encoding. *Bell System Technical Journal*, 54(9):1597–1614, 1975. URL: https://doi.org/10.1002/j.1538-7305.1975.tb02053.x.

[BackstromLF+17] Tom Bäckström, Jérémie Lecomte, Guillaume Fuchs, Sascha Disch, and Christian Uhle. *Speech coding: with code-excited linear prediction*. Springer, 2017. URL: https://doi.org/10.1007/978-3-319-50204-5.

[ZSLC17] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. Hello edge: keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017. URL: https://doi.org/10.48550/arXiv.1711.07128.

[Zha19] Xiaohui Zhang. *Strategies for Handling Out-of-Vocabulary Words in Automatic Speech Recognition*. PhD thesis, Johns Hopkins University, 2019. URL: http://jhir.library.jhu.edu/handle/1774.2/62275.

[BZMA20] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. Wav2vec 2.0: a framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020. URL: https://doi.org/10.48550/arXiv.2006.11477.

[BGV92] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, 144–152. 1992. URL: https://doi.org/10.1145/130385.130401.

[CHTG19] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James Glass. An unsupervised autoregressive model for speech representation learning. *arXiv preprint arXiv:1904.03240*, 2019. URL: https://doi.org/10.48550/arXiv.1904.03240.

[EWollmerS10] Florian Eyben, Martin Wöllmer, and Björn Schuller. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, 1459–1462. 2010. URL: https://doi.org/10.1145/1873951.1874246.

[PRasanenK15] Jouni Pohjalainen, Okko Räsänen, and Serdar Kadioglu. Feature selection methods and their combinations in high-dimensional classification of speaker likability, intelligibility and personality traits. *Computer Speech & Language*, 29(1):145–171, 2015. URL: https://doi.org/10.1016/j.csl.2013.11.004.

[SB13] Björn Schuller and Anton Batliner. *Computational paralinguistics: emotion, affect and personality in speech and language processing*. John Wiley & Sons, 2013. URL: https://www.wiley.com/en-us/9781118706626.

[WVBWK15] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 4460–4464. IEEE, 2015. URL: https://doi.org/10.1109/ICASSP.2015.7178814.

[HB96] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, 373–376. IEEE, 1996. URL: https://doi.org/10.1109/ICASSP.1996.541110.

[RS07] Lawrence R Rabiner and Ronald W Schafer. Introduction to digital speech processing. *Foundations and Trends in Signal Processing*, 1(1):1–194, 2007. URL: https://doi.org/10.1561/2000000001.

[BackstromLF+17] Tom Bäckström, Jérémie Lecomte, Guillaume Fuchs, Sascha Disch, and Christian Uhle. *Speech coding: with code-excited linear prediction*. Springer, 2017. URL: https://doi.org/10.1007/978-3-319-50204-5.

[BackstromLF+17] Tom Bäckström, Jérémie Lecomte, Guillaume Fuchs, Sascha Disch, and Christian Uhle. *Speech coding: with code-excited linear prediction*. Springer, 2017. URL: https://doi.org/10.1007/978-3-319-50204-5.

[BSH+08] Jacob Benesty, M Mohan Sondhi, Yiteng Huang, and others. *Springer handbook of speech processing*. Volume 1. Springer, 2008. URL: https://doi.org/10.1007/978-3-540-49127-9.

[Bol79] Steven Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on acoustics, speech, and signal processing*, 27(2):113–120, 1979. URL: https://doi.org/10.1109/TASSP.1979.1163209.

[Mar01] Rainer Martin. Noise power spectral density estimation based on optimal smoothing and minimum statistics. *IEEE Transactions on speech and audio processing*, 9(5):504–512, 2001. URL: https://doi.org/10.1109/89.928915.

[Bir05] Peter Birkholz. *3D-Artikulatorische Sprachsynthese*. PhD thesis, der Universität Rostock, 2005. URL: https://www.vocaltractlab.de/publications/birkholz-2005-dissertation.pdf.

[BMW+15] Peter Birkholz, Lucia Martin, Klaus Willmes, Bernd J Kröger, and Christiane Neuschaefer-Rube. The contribution of phonation type to the perception of vocal emotions in german: an articulatory synthesis study. *The Journal of the Acoustical Society of America*, 137(3):1503–1512, 2015. URL: https://doi.org/10.1121/1.4906836.

[Dup18] Emmanuel Dupoux. Cognitive science in the era of artificial intelligence: a roadmap for reverse-engineering the infant language-learner. *Cognition*, 173:43–59, 2018. URL: https://doi.org/10.1016/j.cognition.2017.11.008.

[HBR17] William Havard, Laurent Besacier, and Olivier Rosec. Speech-coco: 600k visually grounded spoken captions aligned to mscoco data set. *arXiv preprint arXiv:1707.08435*, 2017. URL: https://doi.org/10.21437/GLU.2017-9.

[HM14] Ian S Howard and Piers Messum. Learning to pronounce first words in three languages: an investigation of caregiver and infant behavior using a computational model of an infant. *PLoS One*, 9(10):e110334, 2014. URL: https://doi.org/10.1371/journal.pone.0110334.

[KRasanen16] Sofoklis Kakouros and Okko Räsänen. 3pro–an unsupervised method for the automatic detection of sentence prominence in speech. *Speech Communication*, 82:67–84, 2016. URL: https://doi.org/10.1016/j.specom.2016.06.004.

[KJG17] Herman Kamper, Aren Jansen, and Sharon Goldwater. A segmental framework for fully-unsupervised large-vocabulary speech recognition. *Computer Speech & Language*, 46:154–174, 2017. URL: https://doi.org/10.1016/j.csl.2017.04.008.

[Kir02] Simon Kirby. Natural language from artificial life. *Artificial life*, 8(2):185–215, 2002. URL: https://doi.org/10.1162/106454602320184248.

[Mae88] Shinji Maeda. Improved articulatory models. *The Journal of the Acoustical Society of America*, 84(S1):S146–S146, 1988. URL: https://doi.org/10.1121/1.2025845.

[MYL+20] James S Magnuson, Heejo You, Sahil Luthra, Monica Li, Hosung Nam, Monty Escabi, Kevin Brown, Paul D Allopenna, Rachel M Theodore, Nicholas Monto, and others. Earshot: a minimal neural network model of incremental human speech recognition. *Cognitive science*, 44(4):e12823, 2020. URL: https://doi.org/10.1111/cogs.12823.

[Mar82] David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. W.H. Freeman and Company, 1982.

[MWG02] Jessica Maye, Janet F Werker, and LouAnn Gerken. Infant sensitivity to distributional information can affect phonetic discrimination. *Cognition*, 82(3):B101–B111, 2002. URL: https://doi.org/10.1016/S0010-0277(01)00157-3.

[ME86] James L McClelland and Jeffrey L Elman. The trace model of speech perception. *Cognitive psychology*, 18(1):1–86, 1986. URL: https://doi.org/10.1016/0010-0285(86)90015-0.

[NSM15] Tasha Nagamine, Michael L Seltzer, and Nima Mesgarani. Exploring how deep neural networks form phonemic categories. In *Sixteenth Annual Conference of the International Speech Communication Association*. 2015. URL: https://www.isca-speech.org/archive_v0/interspeech_2015/papers/i15_1912.pdf.

[Nor94] Dennis Norris. Shortlist: a connectionist model of continuous speech recognition. *Cognition*, 52(3):189–234, 1994. URL: https://doi.org/10.1016/0010-0277(94)90043-4.

[OKS19] Pierre-Yves Oudeyer, George Kachergis, and William Schueller. Computational and robotic models of early language development: a review. In J.S. Horst and J. von Koss Torkildsen, editors, *International handbook of language acquisition*. Routledge/Taylor & Francis Group, 2019. URL: https://psycnet.apa.org/doi/10.4324/9781315110622-5.

[RRasanen17] Heikki Rasilo and Okko Räsänen. An online model for vowel imitation learning. *Speech Communication*, 86:1–23, 2017. URL: https://doi.org/10.1016/j.specom.2016.10.010.

[Rasanen12] Okko Räsänen. Computational modeling of phonetic and lexical learning in early language acquisition: existing models and future directions. *Speech Communication*, 54(9):975–997, 2012. URL: https://doi.org/10.1016/j.specom.2012.05.001.

[RasanenDF18] Okko Räsänen, Gabriel Doyle, and Michael C Frank. Pre-linguistic segmentation of speech into syllable-like units. *Cognition*, 171:130–150, 2018. URL: https://doi.org/10.1016/j.cognition.2017.11.003.

[RasanenR15] Okko Räsänen and Heikki Rasilo. A joint model of word segmentation and meaning acquisition through cross-situational learning. *Psychological review*, 122(4):792, 2015. URL: https://psycnet.apa.org/doi/10.1037/a0039702.

[SAN96] Jenny R Saffran, Richard N Aslin, and Elissa L Newport. Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–1928, 1996. URL: https://doi.org/10.1126/science.274.5294.1926.

[SK18] Jenny R Saffran and Natasha Z Kirkham. Infant statistical learning. *Annual review of psychology*, 69:181–203, 2018. URL: https://doi.org/10.1146/annurev-psych-122216-011805.

[Ste97] Luc Steels. The synthetic modeling of language origins. *Evolution of communication*, 1(1):1–34, 1997. URL: https://doi.org/10.1075/eoc.1.1.02ste.

[TG11] Jason A Tourville and Frank H Guenther. The diva model: a neural theory of speech acquisition and production. *Language and cognitive processes*, 26(7):952–981, 2011. URL: https://doi.org/10.1080/01690960903498424.

[WS12] Andrea Weber and Odette Scharenborg. Models of spoken-word recognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 3(3):387–401, 2012. doi:10.1002/wcs.1178.

[WT84] Janet F Werker and Richard C Tees. Cross-language speech perception: evidence for perceptual reorganization during the first year of life. *Infant behavior and development*, 7(1):49–63, 1984. URL: https://doi.org/10.1016/S0163-6383(84)80022-3.

[FWF13] Rachel L Finn, David Wright, and Michael Friedewald. Seven types of privacy. In *European data protection: coming of age*, pages 3–32. Springer, 2013. URL: https://doi.org/10.1007/978-94-007-5170-5_1.

[Gas18] Jessica Gasiorek. *Message processing: The science of creating understanding*. UH Mānoa Outreach College, 2018. URL: http://pressbooks-dev.oer.hawaii.edu/messageprocessing/.

[Lar19] Xabier Lareo. Smart speakers and virtual assistants. *TechDispatch #1:*, 2019. URL: https://data.europa.eu/doi/10.2804/004275.

[NJK+19] Andreas Nautsch, Catherine Jasserand, Els Kindt, Massimiliano Todisco, Isabel Trancoso, and Nicholas Evans. The gdpr & speech data: reflections of legal and technology communities, first steps towards a common understanding. *arXiv preprint arXiv:1907.03458*, 2019. URL: https://doi.org/10.21437/Interspeech.2019-2647.

[1] Peter Noll. A comparative study of various quantization schemes for speech encoding. *Bell System Technical Journal*, 54(9):1597–1614, 1975. URL: https://doi.org/10.1002/j.1538-7305.1975.tb02053.x.

[2] Tom Bäckström, Jérémie Lecomte, Guillaume Fuchs, Sascha Disch, and Christian Uhle. *Speech coding: with code-excited linear prediction*. Springer, 2017. URL: https://doi.org/10.1007/978-3-319-50204-5.

[3] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. Hello edge: keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017. URL: https://doi.org/10.48550/arXiv.1711.07128.

[4] Xiaohui Zhang. *Strategies for Handling Out-of-Vocabulary Words in Automatic Speech Recognition*. PhD thesis, Johns Hopkins University, 2019. URL: http://jhir.library.jhu.edu/handle/1774.2/62275.

[5] Björn Schuller and Anton Batliner. *Computational paralinguistics: emotion, affect and personality in speech and language processing*. John Wiley & Sons, 2013. URL: https://www.wiley.com/en-us/9781118706626.

[6] Jouni Pohjalainen, Okko Räsänen, and Serdar Kadioglu. Feature selection methods and their combinations in high-dimensional classification of speaker likability, intelligibility and personality traits. *Computer Speech & Language*, 29(1):145–171, 2015. URL: https://doi.org/10.1016/j.csl.2013.11.004.

[7] Florian Eyben, Martin Wöllmer, and Björn Schuller. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, 1459–1462. 2010. URL: https://doi.org/10.1145/1873951.1874246.

[8] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. Wav2vec 2.0: a framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020. URL: https://doi.org/10.48550/arXiv.2006.11477.

[9] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James Glass. An unsupervised autoregressive model for speech representation learning. *arXiv preprint arXiv:1904.03240*, 2019. URL: https://doi.org/10.48550/arXiv.1904.03240.

[10] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, 144–152. 1992. URL: https://doi.org/10.1145/130385.130401.

[11] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 4460–4464. IEEE, 2015. URL: https://doi.org/10.1109/ICASSP.2015.7178814.

[12] Lawrence R Rabiner and Ronald W Schafer. Introduction to digital speech processing. *Foundations and Trends in Signal Processing*, 1(1):1–194, 2007. URL: https://doi.org/10.1561/2000000001.

[13] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, 373–376. IEEE, 1996. URL: https://doi.org/10.1109/ICASSP.1996.541110.

[14] Jacob Benesty, M Mohan Sondhi, Yiteng Huang, and others. *Springer handbook of speech processing*. Volume 1. Springer, 2008. URL: https://doi.org/10.1007/978-3-540-49127-9.

[15] Rainer Martin. Noise power spectral density estimation based on optimal smoothing and minimum statistics. *IEEE Transactions on speech and audio processing*, 9(5):504–512, 2001. URL: https://doi.org/10.1109/89.928915.

[16] Steven Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on acoustics, speech, and signal processing*, 27(2):113–120, 1979. URL: https://doi.org/10.1109/TASSP.1979.1163209.

[17] William Havard, Laurent Besacier, and Olivier Rosec. Speech-coco: 600k visually grounded spoken captions aligned to mscoco data set. *arXiv preprint arXiv:1707.08435*, 2017. URL: https://doi.org/10.21437/GLU.2017-9.

[18] James L McClelland and Jeffrey L Elman. The trace model of speech perception. *Cognitive psychology*, 18(1):1–86, 1986. URL: https://doi.org/10.1016/0010-0285(86)90015-0.

[19] Okko Räsänen. Computational modeling of phonetic and lexical learning in early language acquisition: existing models and future directions. *Speech Communication*, 54(9):975–997, 2012. URL: https://doi.org/10.1016/j.specom.2012.05.001.

[20] Emmanuel Dupoux. Cognitive science in the era of artificial intelligence: a roadmap for reverse-engineering the infant language-learner. *Cognition*, 173:43–59, 2018. URL: https://doi.org/10.1016/j.cognition.2017.11.008.

[21] Luc Steels. The synthetic modeling of language origins. *Evolution of communication*, 1(1):1–34, 1997. URL: https://doi.org/10.1075/eoc.1.1.02ste.

[22] Simon Kirby. Natural language from artificial life. *Artificial life*, 8(2):185–215, 2002. URL: https://doi.org/10.1162/106454602320184248.

[23] David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. W.H. Freeman and Company, 1982.

[24] Andrea Weber and Odette Scharenborg. Models of spoken-word recognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 3(3):387–401, 2012. doi:10.1002/wcs.1178.

[25] James S Magnuson, Heejo You, Sahil Luthra, Monica Li, Hosung Nam, Monty Escabi, Kevin Brown, Paul D Allopenna, Rachel M Theodore, Nicholas Monto, and others. Earshot: a minimal neural network model of incremental human speech recognition. *Cognitive science*, 44(4):e12823, 2020. URL: https://doi.org/10.1111/cogs.12823.

[26] Janet F Werker and Richard C Tees. Cross-language speech perception: evidence for perceptual reorganization during the first year of life. *Infant behavior and development*, 7(1):49–63, 1984. URL: https://doi.org/10.1016/S0163-6383(84)80022-3.

[27] Jenny R Saffran, Richard N Aslin, and Elissa L Newport. Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–1928, 1996. URL: https://doi.org/10.1126/science.274.5294.1926.

[28] Jessica Maye, Janet F Werker, and LouAnn Gerken. Infant sensitivity to distributional information can affect phonetic discrimination. *Cognition*, 82(3):B101–B111, 2002. URL: https://doi.org/10.1016/S0010-0277(01)00157-3.

[29] Jenny R Saffran and Natasha Z Kirkham. Infant statistical learning. *Annual review of psychology*, 69:181–203, 2018. URL: https://doi.org/10.1146/annurev-psych-122216-011805.

[30] Tasha Nagamine, Michael L Seltzer, and Nima Mesgarani. Exploring how deep neural networks form phonemic categories. In *Sixteenth Annual Conference of the International Speech Communication Association*. 2015. URL: https://www.isca-speech.org/archive_v0/interspeech_2015/papers/i15_1912.pdf.

[31] Okko Räsänen and Heikki Rasilo. A joint model of word segmentation and meaning acquisition through cross-situational learning. *Psychological review*, 122(4):792, 2015. URL: https://psycnet.apa.org/doi/10.1037/a0039702.

[32] Sofoklis Kakouros and Okko Räsänen. 3pro–an unsupervised method for the automatic detection of sentence prominence in speech. *Speech Communication*, 82:67–84, 2016. URL: https://doi.org/10.1016/j.specom.2016.06.004.

[33] Herman Kamper, Aren Jansen, and Sharon Goldwater. A segmental framework for fully-unsupervised large-vocabulary speech recognition. *Computer Speech & Language*, 46:154–174, 2017. URL: https://doi.org/10.1016/j.csl.2017.04.008.

[34] Okko Räsänen, Gabriel Doyle, and Michael C Frank. Pre-linguistic segmentation of speech into syllable-like units. *Cognition*, 171:130–150, 2018. URL: https://doi.org/10.1016/j.cognition.2017.11.003.

[35] Dennis Norris. Shortlist: a connectionist model of continuous speech recognition. *Cognition*, 52(3):189–234, 1994. URL: https://doi.org/10.1016/0010-0277(94)90043-4.

[36] Shinji Maeda. Improved articulatory models. *The Journal of the Acoustical Society of America*, 84(S1):S146–S146, 1988. URL: https://doi.org/10.1121/1.2025845.

[37] Peter Birkholz. *3D-Artikulatorische Sprachsynthese*. PhD thesis, der Universität Rostock, 2005. URL: https://www.vocaltractlab.de/publications/birkholz-2005-dissertation.pdf.

[38] Peter Birkholz, Lucia Martin, Klaus Willmes, Bernd J Kröger, and Christiane Neuschaefer-Rube. The contribution of phonation type to the perception of vocal emotions in german: an articulatory synthesis study. *The Journal of the Acoustical Society of America*, 137(3):1503–1512, 2015. URL: https://doi.org/10.1121/1.4906836.

[39] Jason A Tourville and Frank H Guenther. The diva model: a neural theory of speech acquisition and production. *Language and cognitive processes*, 26(7):952–981, 2011. URL: https://doi.org/10.1080/01690960903498424.

[40] Ian S Howard and Piers Messum. Learning to pronounce first words in three languages: an investigation of caregiver and infant behavior using a computational model of an infant. *PLoS One*, 9(10):e110334, 2014. URL: https://doi.org/10.1371/journal.pone.0110334.

[41] Heikki Rasilo and Okko Räsänen. An online model for vowel imitation learning. *Speech Communication*, 86:1–23, 2017. URL: https://doi.org/10.1016/j.specom.2016.10.010.

[42] Pierre-Yves Oudeyer, George Kachergis, and William Schueller. Computational and robotic models of early language development: a review. In J.S. Horst and J. von Koss Torkildsen, editors, *International handbook of language acquisition*. Routledge/Taylor & Francis Group, 2019. URL: https://psycnet.apa.org/doi/10.4324/9781315110622-5.

[43] Xabier Lareo. Smart speakers and virtual assistants. *TechDispatch #1:*, 2019. URL: https://data.europa.eu/doi/10.2804/004275.

[44] Andreas Nautsch, Catherine Jasserand, Els Kindt, Massimiliano Todisco, Isabel Trancoso, and Nicholas Evans. The gdpr & speech data: reflections of legal and technology communities, first steps towards a common understanding. *arXiv preprint arXiv:1907.03458*, 2019. URL: https://doi.org/10.21437/Interspeech.2019-2647.

[45] Jessica Gasiorek. *Message processing: The science of creating understanding*. UH Mānoa Outreach College, 2018. URL: http://pressbooks-dev.oer.hawaii.edu/messageprocessing/.

[46] Rachel L Finn, David Wright, and Michael Friedewald. Seven types of privacy. In *European data protection: coming of age*, pages 3–32. Springer, 2013. URL: https://doi.org/10.1007/978-94-007-5170-5_1.